AIX Operating System for the PS/2 and System/370 Commands Reference

Volume 1 and Volume 2

Document Number SC23-2184-1

AIX Operating System for the PS/2 and System/370

Commands Reference

Volume 1 and Volume 2

Document Number SC23-2184-1

Edition Notice

Third Edition (March 1991)

This edition applies to Version 1.2.1 of the IBM Advanced Interactive Executive for the System/370 (AIX/370), Program Number 5713-AFL, and for Version 1.2.1 of the IBM Advanced Interactive Executive for the Personal System/2, Program Number 5713-AEQ, and to all subsequent releases until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

A form for reader's comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, Department 52QA MS 911
Neighborhood Road
Kingston, NY 12401
U.S.A.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Notices

Subtopics

Trademarks and Acknowledgments

Table of Contents

Tables

1.0 Volume 1

Subtopics

1.1 Chapter 1. Commands

The first section of this chapter explains how to use the syntax diagrams in this book and gives other general information about AIX commands. The remainder of the chapter is composed of reference material for commands beginning with the letters **a** through **m**. Commands beginning with the letters **n** through **z** are covered in Volume 2 of the *AIX Commands Reference*.

Subtopics

1.1.1 Reading the Syntax Diagrams

The syntax diagrams provide you with information about how to enter a command on the command line. A syntax diagram can tell you:

- Which flags can be entered on the command lin
- Which flags must take parameter
- Which flags have optional parameter
- What the default values of the flags and parameters ar
- Which flags can and cannot be entered togethe
- Where you must enter flags or parameters and where you have a choic
- Where you can repeat flag and parameter sequences

Syntax diagrams are read from left to right. The first item in the diagram is the name of the command you want to enter. After the command name, the path may branch into two paths. You can follow either path until you encounter the end mark. This mark means that nothing more can be entered with the command.

Sometimes a diagram is too long to fit on one line. In this case, you see an arrow which indicates that you should go to the next line of the diagram and continue entering the command.

Subtopics

1.1.1.1 Notational Conventions and Symbols

The following notational conventions and symbols are used in the syntax diagrams.

bold	Items that you must enter exactly as shown appear in **bold**. These items include command names, flags, and literal characters.
italics	Items for which you must specify a value appear in **italics**. These items include parameters that follow flags and parameters that the command reads, such as files and directories.
	When you follow a path that takes you to a box with this repeat arrow around it, you must choose at least one item from the box. You may continue forward along the path or follow the repeat arrow to another part of the diagram.
	If a diagram is too long to fit on one line, this arrow tells you to go to the next line of the diagram to continue entering your command line.

1.1.1.2 Types of Syntax Diagrams

This section contains descriptions and illustrations of the various types of syntax diagrams appearing in this book.

Subtopics

1.1.1.2.1 Command Only

The simplest syntax diagram contains a single command that is entered on the command line. For example:

```
logname ---¦
```

The command name is in bold which means that **logname** should be entered literally. As you follow the line away from the command name to the right, you reach the end mark and must stop. To enter this command, you would enter:

```
  logname
```

1.1.1.2.2 Commands with Required Parameters

Many diagrams have parameters that represent specific values that you must enter on the command line. When you encounter a parameter, read the command description to determine what to enter in place of the parameter. The following diagram contains a required parameter:

```
dirname -- path --¦
```

The command name is in bold which means that **dirname** should be entered literally. As you follow the line away from the command name to the right, the next item that you encounter is the parameter **path**. Since there is only one line and it goes through this parameter, you must supply a path. As you move further to the right, you reach the end mark and must stop. To enter this command, you might enter:

```
  dirname /u/tom/program.c
```

Suppose you wanted to use the **dirname** command for two paths, **/u/tom/program.c** and **/u/fran/a.out**. Since this command accepts only one path, you would have to enter the command twice:

```
  dirname /u/tom/program.c
  dirname /u/fran/a.out
```

Some diagrams contain a **repeat arrow**, an arrow that provides a path back to an earlier part of the diagram. The following diagram shows a **repeat arrow**:

```
sact -- file --¦
             ¦
      +------+
```

The command name is in bold, indicating **sact** should be entered literally. As you follow the line away from the command name to the right, the next item that you encounter is the parameter **file**. Since there is only one path and it goes through this parameter, you must enter a file name. As you move further to the right, you reach a repeat arrow. Here you may choose to remain on the main path and proceed to the end mark or follow the repeat arrow around to the point between the command name and the parameter. Following the repeat arrow allows you to select the parameter again. If there is a maximum number of parameters that you can enter, the diagram tells you that number. If no maximum is specified (as in this diagram), you can choose to follow the repeat arrow again and again until you reach the limit of the length of a command line. Here are some examples:

```
  sact  s.letter
  sact  s.letter  s.memo  s.report
```

1.1.1.2.3 Commands with an Optional Flag or Parameter

Many commands have optional flags or parameters. If something is optional, you have a choice of paths in the diagram. One path takes you around the item, and the other path takes you through it.

```
       +-----+
del ---¦     +-- file --¦
       +- - -+        ¦
               +------+
```

In this diagram, as you move to the right from the command name, you reach a branch. You can either take the upper branch (where you enter nothing) or you can take the lower branch (where you enter the - flag). The line above the flag is the default line. After the branch, you encounter a parameter. Since the only path goes through the parameter, you must enter it. After the parameter is entered once, you can choose to proceed to the end mark or use the repeat arrow to select the parameter again. For example:

```
  del  file1
  del  file1  file2  file3
  del  -  file1  file2
```

As the command syntax becomes more complicated, the features of the diagrams are combined to help you enter commands properly. The next diagram shows a command that accepts an optional flag and an optional parameter that can be repeated.

```
      +------+   +-------------------+
df ---+- -I -+---¦                  ¦+---¦
      +- -L -¦   +--- filesystem ---+
      +- -M -¦                     ¦
      +- -g -¦     +--------------+
      +- -i -¦
      +- -p -¦
      +- -s -¦
      +- -v -+
```

¦ The default action is to provide information for each file system on each site in the current cluster partition for which the **/etc/filesystems** entry has the attribute free=true.

In this diagram, as you move to the right from the command name, you reach a branch. You can either take the upper branch (where you enter nothing) or the lower branch (where you enter one of the flags). Next you encounter another branch. Here you can either take the upper branch (where you enter nothing) or the lower branch (where you enter the parameter). If you choose to enter the parameter, you can enter the parameter once and proceed to the end mark or you can follow the repeat arrow and select the parameter again. For example:

```
  df
  df  -s
  df  system1
  df  -s  system1
```

```
df  system1  system2
df  -s  system1  system2  system3
```

1.1.1.2.4 Commands That Take More Than One Flag

With many commands, you can enter as many items from a group of flags or parameters as you want as long as you do not exceed the limits of the length of the command line. If this is the case, the items are in a box that has a repeat arrow around it. Follow the arrow around and through the box as many times as necessary to select all of the items you want to use. Once you choose an item from the box, you should not choose it again unless a footnote indicates a flag can be used more than once.

```
       +------------+   +------------+
cat ---¦   +----+   +---¦            +---¦
       +---¦ -b +---+   +--- file ---+
           ¦ -e ¦                  ¦
         ¦ ¦ -n ¦ ¦       +--------+
         ¦ ¦ -q ¦ ¦
         ¦ ¦ -s ¦ ¦
         ¦ ¦ -t ¦ ¦
         ¦ ¦ -u ¦ ¦
         ¦ ¦ -v ¦ ¦
         ¦ +----+ ¦
         +--------+
```

With this command, you can enter only the command name by following the default line over the box, or you can enter one flag and then continue to the end mark, or you can follow the arrow around and choose more than one flag. Following are examples of valid uses of this command:

```
  cat
  cat  -u
  cat  -s
  cat  -u  -s
  cat  -u  -s  -b  notes
  cat  notes
```

1.1.1.2.5 Commands with an Exclusive Flag or Parameter

Many commands have flags or parameters that should not be entered together on the command line. Mutually exclusive items are enclosed in a box with the words "one of" above it. You can choose only one item from this type of box. The following diagram contains such a box.

```
        +--------+
mesg ---¦ one of +---¦
        ¦ +---+  ¦
        +-¦ n +--+
          ¦ y ¦
          +---+
```

Valid ways to enter this command are:

```
  mesg
  mesg  n
  mesg  y
```

1.1.1.2.6 Commands That Can Repeat Part of a Sequence

Some commands allow you to choose flags for each parameter that they read. When this is the case, more than one repeat arrow allows you to go back to earlier parts of the diagram.

```
         +--------+          +-----+
fuser ---¦ +----+ +-- file --¦     +---¦
         +-¦ -k +-+        ¦ +- - -+ ¦
       ¦   ¦ -  ¦¦  +------+         ¦
       ¦  ¦+----+¦                   ¦
       ¦  +------+                   ¦
       +-----------------------------+
```

In this diagram, there are three repeat arrows. The first allows you to choose one or both flags. The second allows you to have **fuser** read more than one file. The third allows you to repeat the complete sequence from the beginning of the diagram to the end. The following are correct ways to enter **fuser**:

```
  fuser  memo
  fuser  memo  -
  fuser  -k  memo
  fuser  -k  -u  memo
  fuser  -k  -u  memo  letter
  fuser  -k  -u  memo  -
  fuser  -k  memo  -  -u  -k  letter  -u  report  -
```

The third arrow allows you to enter the same flag more than once but only **after** at least one file name has been entered. If you follow the diagram, you cannot repeat a flag without entering at least one file name after it.

1.1.1.2.7 Commands with Default Values

The default line can show more than just an alternate path around flags and parameters. Sometimes a flag is set automatically (by default), and sometimes a parameter has a default value. When this is the case, the default value is shown in the normal font on the default line.

```
           +---------- -p -------------------------+
gettext ---¦           maximum three each          +-- outfile --¦
           ¦ +------+   +--------------+           ¦
           +-¦      +---¦ -h helpnum   +--- infile -+
             +- -p -+   ¦ -m mesgnum   ¦ ¦
                      ¦ ¦ -t insertnum ¦ ¦
                      ¦ +--------------+ ¦
                      +------------------+
```

If you do not enter any flags with **gettext**, the **-p** flag is set by default. If you choose the path that contains the **-h**, **-m**, and **-t** flags, you must choose whether or not to use the **-p** flag also. The following command lines are equivalent ways of entering **gettext**:

```
  gettext  -p  report
  gettext  report
```

The following is a valid use of this command in which the default flag for **gettext** is not used:

```
  gettext  -m3  memo  report
```

To select both the **-p** and **-m** flag, you must explicitly enter the **-p** flag. For example:

```
  gettext -p -m3 memo report
```

You can also have default parameter values.

```
         +-------------+   +- /etc/trcprofile -+
trace ---¦ +---------+ +---¦                   +---¦
         +-¦ -o name +-+   +----- profile -----+
           +---------+¦
          +-----------+
```

In this case, you can choose to specify a value for **profile** or you can use the default value **/etc/trcprofile**. If a value for **profile** is not supplied, **trace** reads the file **/etc/trcprofile**.

The following are equivalent uses of this command:

```
  trace
  trace  /etc/trcprofile
```

1.1.1.2.8 Diagrams That Are Continued on the Next Line

Some of the more complex diagrams cannot fit on one line. They are marked with a continuation arrow where they break, and they continue with the arrowhead on the next line.

```
      +- -l66 -o0 +1 -+   +-- -1 --------------------------------+   +-------
pr ---¦  +---------+  +---¦ +- -m -------------+   +-----------+ +---¦      or
      +--¦ -d      +--+   +-¦         +------+ +---¦           +-+   ¦ +------
         ¦ -       ¦¦       +- -num --¦      +-+   +- -s char -+     +-¦ -t
        ¦¦ -l num  ¦¦                 +- -a -+                         ¦ -h "s
        ¦¦ -o num  ¦¦                                                  +------
        ¦¦ -p      ¦¦
        ¦¦ -r      ¦¦
        ¦¦ -w num  ¦¦
        ¦¦ + num   ¦¦
        ¦+---------+¦
        +-----------+

    +------------------------------+   +---------------------------------+   +
 ---¦       +--------+   +-- 5 --+ +---¦ one of                          +---¦
    +- -n --¦        +---¦       +-+   ¦ +----+   +--------+   +-- 8 --+ ¦   +
            +- char -+   +- num -+     +-¦ -e +---¦¦       +---¦¦      +-+
                                         ¦ -i ¦   +- char -+   +- num -+¦
                                        ¦+----+                         ¦
                                        +-------------------------------+
```

```
-----------------
¦ Do not put a blank between these items.
```

Follow the diagram making choices until you reach the continuation arrow. Then go down to the arrowhead on the next line. Continue until you reach the end mark.

Following the diagram will seem to impose a specific order to the flags; however, you do not need to strictly follow that order when entering the command. If strict order is important, it is stated under "Description" in the commands discussion.

This diagram has a footnote. Footnotes are used to show information that cannot be diagramed. In this case, the footnote tells you that you cannot put a space between the **-e** or **-i** flags and their parameters.

Following are some of the ways you can enter the **pr** command:

```
  pr
  pr  -d
  pr  -o4  -r  -m  -sX  memo  letter
  pr  -r  -m  -t  -n4  -iX3  memo  letter  report
  pr  -m  -n4  -r  -iX3  -t  memo  report  letter
  pr  -l30  5  -3  -a  -nX  -iX3  -eY  memo  report
```

1.1.1.2.9 Commands with More Than One Diagram

Some commands require more than one diagram to indicate the different ways a command can be entered.

```
           +----------------+   +--------------+          +-------+
install ---¦                +---¦ +----------+ +-- file --¦       +---¦
           +- -n directory -+   +-¦ -i       +-+          +- dir -+
                                  ¦ -        ¦¦                  ¦
                                 ¦¦ -o       ¦¦           +-----+
                                 ¦¦ -s       ¦¦
                                 ¦¦ -M mode  ¦¦
                                 ¦¦ -O owner ¦¦
                                 ¦¦ -G group ¦¦
                                 ¦+----------+¦
                                 +------------+

           +- -c directory ----------+   +----------+
install ---¦                +------+ +---¦ -m       +-- file --¦
           +- -f directory -¦      +-+   ¦ -s       ¦
                            +- -o -+     ¦ -M mode  ¦
                                         ¦ -O owner ¦
                                         ¦ -G group ¦
                                         +----------+
```

Because some flags and parameters cannot be combined with others, two diagrams are required to indicate the ways the command can be entered. For example, the following are ways you can enter this command:

```
  install  -o fixit  /etc  /games
  install  -n  /usr/bin  fixit
  install  -c  /usr/bin  fixit
  install  -f  /usr/bin  -o  -s  fixit
```

1.1.1.2.10 File Input and Output in Diagrams

Some commands read a file as their input, while others read standard input, and still others read both. The syntax diagrams help you to determine which case applies to each particular command. If a command reads a file as its input, the diagram shows a path through a parameter representing the file and the "Description" section tells you that this file is an input file. The path in the diagram does not have an empty default line above it. The following is an example of a command that reads an input file:

```
                                 +------------+
sccsdiff -- -r SID1 -- -r SID2 --¦ +--------+ +-- file --¦
                                 +-¦ -p     +-+         ¦
                                   ¦ -s nu  ¦¦  +------+
                                  ¦+--------+¦
                                  +----------+
```

A command that can read either input files or standard input has no default line above the file parameter. If the command can write its output to either an output file or to standard output, it is also shown with no default line above the output file parameter. If a command can read only from standard input, an input file is not shown in the diagram, and standard input is assumed. If a command writes only to standard output, an output file is not shown in the diagram, and standard output is assumed. When you must supply a file name for input or output, the file parameter is included in the diagram without an empty default line above it.

1.1.2 Command Input and Output

Many commands take their input from **standard input** and write their output to **standard output**. By default, standard input comes from the keyboard, and standard output goes to the display. It is important to remember this distinction as you read the command descriptions since they describe the default action. In this context, the verb **display** means "write to the standard output". Any command that reads standard input and writes to standard output can have its input or output redirected to a file and can be used in a **pipeline**, where the standard output of a previous command is directed to the standard input of the next command. For more information on pipelines, see "sh, Rsh" in topic 1.1.420.

1.1.3 File Name Substitution

When **file** is supplied as an argument to either a command or a flag, you can automatically produce a list of file name arguments by specifying a pattern for the shell to match with file names in a directory. Most characters in such a pattern match themselves, but you can also use some special **pattern-matching characters** in your pattern. These special characters are:

*	Matches any string, including the null string.
?	Matches any one character.
[...]	Matches any one of the characters enclosed in square brackets.
[!...]	Matches any character **other than** one of the characters that follow the exclamation mark within square brackets.

Inside square brackets, a pair of characters separated by a - (hyphen) specifies a set of all characters lexically within the inclusive range of that pair, so that **[a-dy]** is equivalent to **[abcdy]**.

Using pattern-matching characters in file names on the command line has some restrictions. If the first character of a file name is a . (dot), it can be matched only by a pattern that begins with a dot. For example, ***file** matches the file names **myfile** and **yourfile**, but not **.myfile** or **.yourfile**. Use the pattern **.*file** to match these file names.

If a pattern does not match any file names, the pattern itself is returned as the result of the match.

Note: File and directory names should not contain the characters *, ?, [, or] because attempts to name such files may instead name additional files in a directory.

1.1.4 Commands (a - m)

The next section contains reference information for the AIX commands. Items included for each command are:

- The purpose of a comman
- One or more syntax diagram
- A description of how a command work
- Descriptions of command flags and subcommand
- A list of related files and cross-references to related information

1.1.5 ac

Purpose
Produces a printout of user logins.

Syntax

```
          +-----------------+
/etc/ac --¦   +---------+   +--¦
          +---¦ -w wtmp +---+
              ¦ -p      ¦ ¦
            ¦ ¦ -d      ¦ ¦
            ¦ ¦ people  ¦ ¦
            ¦ +---------+ ¦
           +--------------+
```

Description

The **ac** command produces a printout giving connect time for each user who has logged in during the life of the current **wtmp** file. The command also produces a total.

The accounting file **/usr/adm/wtmp** is maintained by the **init** and **login** commands. Neither of these commands creates the file, so if it does not exist, no connect-time accounting is done. To start accounting, the file should be created with length 0. If the file is left undisturbed, it grows without bound, so periodically information should be collected and the file should be truncated.

Flags

-d	Produces a printout for each midnight-to-midnight period.
-p	Prints individual totals. Without this option, only totals are printed.
people	Requests that the printout be limited to only login names of the specified **people**.
-w	Specifies an alternate **wtmp** file. If no **wtmp** file is given, **/usr/adm/wtmp** is used.

Files
/usr/adm/wtmp

Related Information

See the following commands: "init, telinit" in topic 1.1.208 and "login" in topic 1.1.241.

*1.1.6 acct/**

Purpose
Provides accounting shell procedures.

Syntax

```
/usr/lib/acct/chargefee --- user --- number ---¦

                         +--- 500 -----+
/usr/lib/acct/ckpacct ---¦             +---¦
                         +- numblocks -+

                        +------+   +--------------+
/usr/lib/acct/dodisk ---¦      +---¦              +---¦
                        +- -o -+   +- filesystem -+
                                                 ¦
                                    +------------+

/usr/lib/acct/lastlogin ---¦

                         +----------+
/usr/lib/acct/monacct ---¦          ¦+---¦
                         +- number -+

/usr/lib/acct/nulladm --- file ---¦
                                ¦
                       +--------+

/usr/lib/acct/prctmp ---¦

                           +------+   +--------+
                         +-¦      +---¦       ²+-+
/usr/lib/acct/prdaily ---¦ +- -l -+   +- mmdd -+ +---¦
                         +------- -c ------------+
```

¦ The default **number** is the current month.
² The default **mmdd** is the current day.

```
                         +--------------+        +--------------+
/usr/lib/acct/prtacct ---¦ -f fieldspec +- file -¦              +---¦
                         ¦ -v           ¦        +- 'heading' -+
                         +--------------+

/usr/lib/acct/remove ---¦

                          +------------+
/usr/lib/acct/shutacct ---¦            +---¦
                          +- 'reason' -+

/usr/lib/acct/startup ---¦

                            one of
                          +--------+
                          ¦ on     ¦
/usr/lib/acct/turnacct ---¦ off    +---¦
                          ¦ switch ¦
```

+--------+

Description

Subtopics

1.1.6.1 chargefee

The **chargefee** command charges the specified **number** of units to the specified **user**. The **number** variable can have an integer or decimal value. The command writes a record to the **/usr/adm/fee** file to be merged with other accounting records by the **runacct** command.

1.1.6.2 ckpacct

The **ckpacct** command checks the size of **/usr/adm/pacct**. If the size exceeds the number specified in the **numblocks** parameter, the **ckpacct** command invokes the **turnacct switch** command. The default value for **numblocks** is 500.

If the number of free disk blocks in the **/usr** file system falls below 500, the **ckpacct** command automatically turns off the collection of process accounting records by invoking the **turnacct off** command. When 500 blocks are available again, accounting is reactivated. This feature is sensitive to how frequently **ckpacct** is run (usually by the **cron** command).

1.1.6.3 dodisk

The **dodisk** command performs the disk-usage accounting functions. The **cron** command normally runs **dodisk** periodically. By default, the **dodisk** command does disk accounting on the special files whose stanzas in **/etc/filesystems** contain the attribute **account=true**. If you specify the **-o** flag, **dodisk** does a slower version of disk accounting by login directory.

The **filesystem** parameter specifies the one or more file systems where disk accounting is to be done. If you specify any file names, disk accounting is done on only these file systems. If you do not use the **-o** flag, file system names should be the special file names of mountable file systems. If you use both the **-o** flag and **filesystem** parameter, the files should be mount points of mounted file systems.

1.1.6.4 lastlogin

The **lastlogin** command updates the file **/usr/adm/acct/sum/loginlog** to show the last date each user logged in. The **runacct** command normally calls **lastlogin**.

1.1.6.5 monacct

The **monacct** command performs monthly or periodic accounting. The **cron** command should run this command once each month or accounting period. The **number** parameter indicates the month or period to process. The default **number** is the current month. This default is useful if **monacct** is run by **cron** on the first day of each month. The **monacct** command creates summary files in **/usr/adm/acct/fiscal** and restarts summary files in **/usr/adm/acct/sum**.

Daily reports are deleted and thus inaccessible each time the **monacct** command runs.

1.1.6.6 nulladm

The **nulladm** command creates a file, assigns it permission code 664, and ensures that its owner and group are **adm**. (See "chmod" in topic 1.1.67 for an explanation of file permissions.) Various accounting shell procedures call the **nulladm** command.

1.1.6.7 prctmp

The **prctmp** command displays the session record file (normally the **/usr/adm/acct/nite/ctmp** file) created by the **acctcon1** command.

1.1.6.8 prdaily

The **prdaily** command formats a report of the day's accounting data. Use the **mmdd** parameter to specify a date other than the current day. The report resides in **/usr/adm/acct/sum/rprtmmdd**, where **mmdd** specifies the month and day of the report. The **runacct** command invokes this command to format a report of the previous day's accounting data.

The **-c** flag reports exceptional resource usage by command and also can be used on the current day's accounting data only. The **-l** flag reports exceptional usage by login ID for the specified date.

1.1.6.9 prtacct

The **prtacct** command formats and displays total accounting (**tacct**) files. You can specify a **heading** for the report by enclosing it in quotation marks.

The **-f fieldspec** parameter selects fields to be displayed, using the field selection mechanism of the **acctmerg** command. The **-v** flag produces verbose output in which more precise notation is used for floating point numbers.

1.1.6.10 remove

The **remove** command deletes all **/usr/adm/acct/sum/wtmp***, **/usr/adm/acct/sum/pacct***, and **/usr/adm/acct/nite/lock*** files.

1.1.6.11 shutacct

The **shutacct** command turns process accounting off and adds a "reason" record to the **/usr/adm/wtmp** file. It is usually invoked during a system shutdown.

1.1.6.12 startup

The **startup** command turns on the accounting functions when the system is started up. It is usually called by the **/etc/rc** command.

1.1.6.13 turnacct

The **turnacct** command provides an interface to the **accton** command for turning on or off process accounting.

The **switch** flag turns accounting off, moves the current **/usr/adm/pacct** file to the next free name in the **/usr/adm/pacctincr** file, where **incr** is a number starting at 1 and increased by one for each additional **pacct** file. After moving the **pacct** file, **turnacct** turns accounting back on.

The **turnacct** command is usually called by the **ckpacct** command, which in turn is called by the **cron** command, keeping the **pacct** file to a manageable size.

Files

/usr/adm/fee	Accumulator for fees charged to login names
/usr/adm/pacct	Current file for process accounting
/usr/adm/pacct*	File used if **pacct** gets large and during running of the daily accounting procedures
/usr/adm/wtmp	Login/logout history file
/usr/lib/acct/ptelus.awk	Shell procedure that calculates the limits for exceptional usage by login ID
/usr/lib/acct/ptecms.awk	Shell procedure that calculates the limits of exceptional usage by command name
/usr/adm/acct/nite	Working directory
/usr/lib/acct	Holds all accounting commands
/usr/adm/acct/sum	Summary directory

Related Information

See the following commands: "acctcms" in topic 1.1.7, "acctcom" in topic 1.1.8, "acctcon1, acctcon2" in topic 1.1.9, "acctmerg" in topic 1.1.11, "chmod" in topic 1.1.67, "cron" in topic 1.1.97, and "runacct" in topic 1.1.398.

See the **acct** system call and the **acct**, **utmp**, and **filesystems** files in *AIX Operating System Technical Reference*.

See "Running System Accounting" in *Managing the AIX Operating System*.

1.1.7 acctcms

Purpose
Produces command usage summaries from accounting records.

Syntax

```
                        +-------------------------+   +------------+   +-----
/usr/lib/acct/acctcms---¦ +- -t ----------------+ +---¦   +----+   +---¦ one c
                        +-¦      +------------+ +-+   +---¦ -j +---+   ¦ +---
                          +- -a -¦   +----+   +-+         ¦ -s ¦       +-¦ -c
                                 +---¦ -o +---+         ¦ +----+ ¦       ¦ -n
                                     ¦ -  ¦ ¦           +--------+       +----
                                   ¦ +----+ ¦
                                   +--------+
```

Description
The **acctcms** command reads the specified **file**s. It adds all records for identically named processes, sorts them, and writes them to standard output in a binary format. Files are usually in the **acct** file format described in *AIX Operating System Technical Reference*.

When you use the **-o** and **-p** flags together, **acctcms** produces a report that combines prime- and non-prime time. All the output summaries are for total usage except those given for number of times run, CPU minutes, and real minutes, which are split into prime and non-prime minutes.

A typical sequence for performing daily command accounting and for maintaining a running total is:

```
  acctcms  file...  > today
  cp  total  previoustotal
  acctcms  -s  today  previoustotal  > total
  acctcms  -a  -s  today
```

Flags

-a Displays output in ASCII summary format rather than binary summary format. Each output line contains the command name, the number of times the command was run, its total **kcore-time**, its total **CPU time**, its total **real time**, its mean memory size (in K bytes), its mean CPU time per invocation of the command, and its **CPU usage factor**. The listed times are all in minutes. **acctcms** normally sorts its output by total kcore-minutes. The unit kcore-minutes measures the amount of storage used (in K-bytes) multiplied by the amount of time it was in use.

-c Sorts by total CPU time rather than total kcore-minutes.

-j Combines all commands called only once under the heading *****other**.

-n Sorts by the number of times the commands were called.

-o Displays a command summary of nonprime-time commands only. You can use this flag only with the **-a** flag.

-p Displays a command summary of prime-time commands only. You can use this flag only with the **-a** flag.

-s	Assumes that any named files that follow this flag are already in binary format.
-t	Processes all records as total accounting records. The default binary format splits each field into prime and non-prime time sections.

Related Information

See the following commands: "acct/*" in topic 1.1.6, "acctcom" in topic 1.1.8, "acctcon1, acctcon2" in topic 1.1.9, "acctmerg" in topic 1.1.11 and "runacct" in topic 1.1.398.

See the **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

See "Running System Accounting" in *Managing the AIX Operating System*.

1.1.8 acctcom

Purpose
Displays selected process accounting record summaries.

Syntax

```
                    +------------------------+
                    ¦        one of          ¦
/usr/bin/acctcom ---¦      +---------+       +---
                    ¦      ¦ -q      ¦       ¦
                    ¦ +----¦ -o file +-----+ ¦
                    +-¦    +---------+     +-+
                      ¦   +------------+   ¦
                      +---¦ -a  -i  -t +---+
                          ¦ -b  -k  -v ¦ ¦
                        ¦ ¦ -f  -m     ¦ ¦
                        ¦ ¦ -h  -r     ¦ ¦
                        ¦ +------------+ ¦
                        +----------------+

    +------------------------------+   +-------------+
 ---¦   +----------------------+   +---¦             +---¦
    +---¦ -C seconds   -e time +---+   +--- file ---+
        ¦ -g group     -E time ¦ ¦                ¦
      ¦ ¦ -H factor    -s time ¦ ¦       +--------+
      ¦ ¦ -I num       -S time ¦ ¦
      ¦ ¦ -l line              ¦ ¦
      ¦ ¦ -n pattern           ¦ ¦
      ¦ ¦ -O seconds           ¦ ¦
      ¦ ¦ -u user              ¦ ¦
      ¦ +----------------------+ ¦
      +--------------------------+
```

Description
The **acctcom** command reads from specified files, standard input, or the **/usr/adm/pacct** file and writes records (selected by flags) to standard output. The input file format is described under **acct** in *AIX Operating System Technical Reference*.

If you do not specify any **file** arguments and standard input is assigned to a work station or **/dev/null** file (as it is when a process runs in the background), **acctcom** reads the **/usr/adm/pacct** file instead of standard input.

By default, if you specify any **file** arguments, the **acctcom** command reads each file chronologically by process completion time. Usually **/usr/adm/pacct** is the current file that you want **acctcom** to examine. Because the **ckpacct** command keeps this file from growing too large, a busy system can have several **pacct** files. All but the current file have the path name **/usr/adm/pacct?**, where **?** is an integer incremented each time a new file is created.

Each record represents one completed process. The default display consists of the command name, user name, **tty** name, start time, end time, real seconds, CPU seconds, and mean memory size in kilobytes. These default items have the following headings in the output:

```
  COMMAND                START   END   REAL   CPU     MEAN
  NAME     USER  TTYNAME TIME   TIME (SECS) (SECS)  SIZE(K)
```

By using the appropriate flags, you also can display the following system statistics:

Flag	**Statistic**
F	Fork/exec
STAT	System exit value
HOG FACTOR	Ratio of total CPU time to elapsed time
KCORE MIN	Product of memory used and elapsed time
CPU FACTOR	Ratio of user time to total (system and user) time
CHARS TRNSFD	Number of characters transferred in input/output operations
BLOCKS READ	Total number of blocks read or written.

If a process ran with superuser authority, its name is prefixed with a # (number sign). If a process is not assigned to a known work station (for example, when the **cron** command runs it), a ? (question mark) appears in the **TTYNAME** field.

Notes:

1. The **acctcom** command reports only on processes that have finished. Use the **ps** command to examine active processes.

2. If a specified time is later than the current time, it is interpreted as occurring on the previous day.

Flags

-a	Shows some average statistics about the processes selected. The statistics are displayed after the output records.
-b	Reads backwards, showing the most recent commands first. This flag has no effect when **acctcom** reads standard input.
-C seconds	Shows only processes whose total CPU time (system time plus user time) exceeds the number specified in the **seconds** variable.
-e time	Selects processes existing at or before the specified time. The order of hours, minutes, and seconds is displayed in the current locale format. The default order is **hh:mm:ss**.
-E time	Selects processes ending at or before the specified time. The order of hours, minutes, and seconds is displayed in the current locale format. The default order is **hh:mm:ss**. If you specify the same time for both the **-E** and **-S** flags, **acctcom** displays the process that existed at the specified time.
-f	Displays the **fork/exec** flag and the system exit value columns

in the output.

-g group	Selects processes belonging to the specified **group**. You can specify either the group ID or the group name.
-h	Instead of mean memory size, shows the fraction of total available CPU time consumed by the process while it ran (hog factor). This factor is computed as (total CPU time)/(elapsed time).
-H factor	Shows only processes that exceed the specified **factor**. See the **-h** flag for a discussion of how this factor is calculated.
-i	Displays columns showing the number of characters transferred in read or write operations (I/O counts).
-I num	Shows only processes transferring more characters than specified by the **num** variable.
-k	Instead of mean memory size, shows total k-core minutes.
-l line	Shows only processes belonging to work station specified by the **/dev/line** file.
-m	Shows mean memory size. This flag is on by default. Specifying the **-h** or **-k** flags turns off **-m**.
-n pattern	Shows only commands matching a **pattern**, where the **pattern** variable is a regular expression like those in the **ed** command (see page 1.1.147). However, in this case you can use a + (plus sign) as a special symbol for one or more occurrences of the preceding character.
-o file	Copies selected process records to the specified **file**, keeping the input data format. This flag suppresses writing to standard output.
-O seconds	Shows only processes with CPU system time exceeding the specified number of **seconds**.
-q	Displays the average statistics that are displayed with the **-a** flag but does not display output records.
-r	Shows CPU factor. This factor is computed as (user-time) / (system-time + user-time).
-s time	Shows only those processes that existed on or after the specified time. The order of hours, minutes, and seconds is displayed in the current locale format. The default order is **hh:mm:ss**.
-S time	Shows only those processes starting at or after the specified time. The order of hours, minutes, and seconds is displayed in the current locale format. The default order is **hh:mm:ss**.
-t	Shows separate system and user CPU times.
-u user	Shows only processes belonging to **user**. The **user** variable can be a user ID, login name that is converted to a user ID,

# to select processes run with superuser authority, or **?** to select processes associated with unknown user IDs.

-v	Eliminates column headings from the output.

Files

/usr/adm/pacct	Current process accounting file.
/etc/passwd	User names and user IDs.
/etc/group	Group names and group IDs.

Related Information

See the following commands: "acctdisk, acctdusg" in topic 1.1.10, "acctcms" in topic 1.1.7, "acctcon1, acctcon2" in topic 1.1.9, "acctmerg" in topic 1.1.11, "acct/*" in topic 1.1.6, "ps" in topic 1.1.337, and "runacct" in topic 1.1.398.

See the **acct** system call, the **acct** and **utmp** files, and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See "Running System Accounting" and "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.9 acctcon1, acctcon2

Purpose
Performs connect-time accounting.

Syntax

```
                                +-------------+
/usr/lib/acct/acctcon1 ---¦ +---------+ +---¦
                                +-¦ -l file +-+
                                  ¦ -o file ¦
                                  ¦ -p      ¦
                                  ¦ -t      ¦
                                  +---------+

/usr/lib/acct/acctcon2 ---¦
```

Description
acctcon1

The **acctcon1** command converts a sequence of login and logout records (read from standard input) to a sequence of login session records (written to standard output). Its input should normally be redirected from **/usr/adm/wtmp**.

The **acctcon1** command displays, in ASCII format, the login device, user ID, login name, ***prime connect time*** (seconds), ***nonprime connect time*** (seconds), session starting time (numeric), and starting date and time (in date/time format). It also maintains a list of ports on which users are logged in. When it reaches the end of its input, it writes a session record for each port that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress (see the **-t** flag on page 1.1.9).

Flags

-l file
Writes to **file** a line-usage summary showing the line name, the number of minutes used, the percentage of total elapsed time used, the number of sessions charged, the number of logins, and the number of logouts. This file helps track line usage and identify bad lines. All hang-ups, terminations of **login**, and terminations of the login shell cause the system to write logout records, so the number of logouts is often much higher than the number of sessions.

-o file
Writes to **file** an overall record for the accounting period, giving starting time, ending time, number of restarts, and number of date changes.

-p Displays input only, showing line name, login name, and time in both numeric and date/time formats.

-t Uses the last time found in the input as the ending time for any current processes instead of the current time. This is necessary in order to have reasonable and repeatable values for noncurrent files.

acctcon2

The **acctcon2** command converts a sequence of login session records, produced by the **acctcon1** command, into total accounting records.

Files

/usr/adm/wtmp	Login/logout history file.

Related Information

See the following commands: "acctdisk, acctdusg" in topic 1.1.10, "acctcms" in topic 1.1.7, "acctcom" in topic 1.1.8, "acctmerg" in topic 1.1.11, "acctprc1, acctprc2, accton" in topic 1.1.12, "acct/*" in topic 1.1.6, "fwtmp, wtmpfix, acctwtmp" in topic 1.1.183, "init, telinit" in topic 1.1.208, "login" in topic 1.1.241, and "runacct" in topic 1.1.398.

See the **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

See "Running System Accounting" in *Managing the AIX Operating System*.

1.1.10 acctdisk, acctdusg

Purpose
Performs disk-usage accounting.

Syntax

```
/usr/lib/acct/acctdisk ---¦

                          +-----------+   +- -p /etc/passwd -+
/usr/lib/acct/acctdusg ---¦           +---¦                  +---¦
                          +- -u file -+   +---- -p file -----+
```

Description

Subtopics
1.1.10.1 acctdisk
1.1.10.2 acctdusg

1.1.10.1 acctdisk

The **acctdisk** command reads lines from standard input that contain a user ID, the user's login name, and the number of disk blocks occupied by his files. It converts these lines to total accounting records that can be merged with other accounting records and writes those records to standard output.

1.1.10.2 acctdusg

The **acctdusg** command produces a report on disk usage in users' home directories. The command takes a list of file names from standard input (usually from a **find / -print** command) and writes a report to standard output. The report indicates the disk resource usage (including indirect blocks) for each user based on files in each user's home directory or subdirectories thereof. Files named on standard input which are not present in any user's home directory are not charged to any user.

Flags

-p file	Searches **file** for login names and numbers, instead of searching **/etc/passwd**.
-u file	Places in **file** records of file names for which it does not charge.

Files

/etc/passwd	Converts login names to user IDs.
/usr/lib/acct	Holds all accounting commands.

Related Information

See the following commands: "acct/*" in topic 1.1.6, "acctcms" in topic 1.1.7, "acctcom" in topic 1.1.8, "acctcon1, acctcon2" in topic 1.1.9, "acctmerg" in topic 1.1.11, "runacct" in topic 1.1.398, and "quot" in topic 1.1.349.

See the **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

1.1.11 acctmerg

Purpose
Merges total accounting files.

Syntax

```
                            +-----------------------------+   +--------+   +----
/usr/lib/acct/acctmerg ---¦ +-------+   +-------------+ +---¦ +----+ +---¦
                          +-¦ -a -i +---¦²            +-+   +-¦ -t +-+   +- f
                            ¦ -h -p ¦   +- fieldspec -+¦      ¦ -u ¦¦
                           ¦¦ -v    ¦                  ¦     ¦+----+¦     +---
                           ¦+-------+                  ¦     +------+     9 ma
                            +--------------------------+
```

¦ **acctmerg** always reads standard input in addition to any named **files**.
² Do not put a blank between these items.

Description
The **acctmerg** command reads records from standard input and up to nine additional **file**s, all in the **tacct** binary format or the **tacct** ASCII format. It merges these by adding records with keys (normally user ID and name) that are identical, and expects the input records to be sorted by those key fields. It writes these merged records to standard output.

The optional **fieldspec**s allow you to select input or output fields. A field specification is a comma-separated list of fields or field ranges. Field numbers are in the order specified in the **tacct** file in *AIX Operating System Technical Reference*, with array sizes, (except for the **ta_name** characters), taken into account. For example, **-h2-3,7,15-13,2** displays the login name, prime CPU and connect times, fee, queueing system, disk usage data, and the login name again, in that order, with column headings. The default specification is "all fields" (**1-18** or **1-**), which produces very wide output lines containing all the available accounting data.

Queueing system, disk usage, or fee data can be converted into **tacct** records using the **-ifieldspec** argument. For example, disk accounting records, produced by **acctdisk**, consist of lines containing the user ID, login name, number of blocks, and number of disk samples (always 1). The **dacct** file contains these records and can be merged into an existing total accounting file, **tacct**, with:

```
  acctmerg  -i1-2,13,18  <dacct | acctmerg  tacct  >output
```

Flags

-a[fieldspec]	Produces output in the form of ASCII records.
-h[fieldspec]	Displays column headings. This flag implies **-a** but is effective with **-p** or **-v**.
-i[fieldspec]	Expects input files composed of ASCII records.
-p[fieldspec]	Displays input without processing.
-t	Produces a single record that contains the totals of all input.

-u	Summarizes by user ID rather than by user name.
-v[fieldspec]	Produces output in ASCII format, with more precise notation for floating-point numbers.

Example
The following sequence is useful for making repairs to any file in **tacct** format:

acctmerg -v <file1 >file2
edit **file2** as desired...
acctmerg -a <file2 >file1

Related Information
See the following commands: "acct/*" in topic 1.1.6, "acctcms" in topic 1.1.7, "acctcom" in topic 1.1.8, "acctcon1, acctcon2" in topic 1.1.9, "acctdisk, acctdusg" in topic 1.1.10 and "runacct" in topic 1.1.398.

See the **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

See "Running System Accounting" in *Managing the AIX Operating System*.

1.1.12 acctprc1, acctprc2, accton

Purpose
Performs process accounting.

Syntax

```
                            +--------+
/usr/lib/acct/acctprc1 ---¦        +---¦
                            +- file -+

/usr/lib/acct/acctprc2 ---¦

                          +--------+
/usr/lib/acct/accton ---¦        +---¦
                          +- file -+
```

Description

Subtopics

1.1.12.1 acctprc1

The **acctprc1** command reads records from standard input that are in the **acct** format (described in *AIX Operating System Technical Reference*), adds the login names that correspond to user IDs, and then writes an ASCII record to standard output. This record contains the user ID, login name, prime CPU time, nonprime CPU time, the total number of characters transferred (in 512-byte units), the total number of blocks read and written, and mean memory size (in 64-byte units) for each process.

If specified, **file** contains a list of login sessions in **ctmp** format (described in *AIX Operating System Technical Reference*), sorted by user ID and login name. By default, **acctprc1** gets login names from the password file, **/etc/passwd**. The information in **file** helps distinguish among different login names that share the same user ID.

1.1.12.2 acctprc2

The **acctprc2** command reads (from standard input) the records written by **acctprc1**, summarizes them by user ID and name, and writes the sorted summaries to standard output as total accounting records.

1.1.12.3 accton

The **accton** command without arguments turns process accounting off. If you specify **file** (the name of an existing file), the kernel adds process accounting records to it (**/usr/adm/pacct** by convention).

Files

/etc/passwd	Password file; contains user IDs.
/usr/adm/pacct	Contains process accounting records.

Related Information

See the following commands: "acct/*" in topic 1.1.6, "acctdisk, acctdusg" in topic 1.1.10, "acctcms" in topic 1.1.7, "acctcom" in topic 1.1.8, "acctcon1, acctcon2" in topic 1.1.9, "acctmerg" in topic 1.1.11, "fwtmp, wtmpfix, acctwtmp" in topic 1.1.183, and "runacct" in topic 1.1.398.

See the **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

See "Running System Accounting" in *Managing the AIX Operating System*.

1.1.13 actmngr

Purpose
Lets you interact with multiple virtual terminals.

Syntax

actmngr ---¦

Description
The **actmngr** command is the ***activity manager*** for the AIX Operating System. It is normally run by the login command in the same manner as any program listed in the **/etc/passwd** file. Once started by **login**, **actmngr** creates the initial shell (**/bin/sh**) and monitors the number of open virtual terminals until all have been closed. It then exits to the AIX **init** process. If you try to end the initial shell when other virtual terminals are still open, **actmngr** restarts the initial shell.

To take advantage of the multiple virtual terminal capability, use the **open** command to execute another shell in a separate virtual terminal.

Note: The **actmngr** command is for PS/2 only.

Notes:

1. You must log out of each existing shell to end your login session.

2. You do not need an activity manager if you do not have virtual terminal capabilities. Thus if you do not log in from the local console, **actmngr** overlays itself with the initial shell.

Related Information

See the following command: "open" in topic 1.1.307.

See "Using Display Station Features" in *Using the AIX Operating System*.

1.1.14 addbib

Purpose
Creates a bibliography.

Syntax

```
          +-------------------+
addbib ---¦ +---------------+ +--- database ---¦
          +-¦ -p promptfile +-+
            ¦ -             ¦¦
           ¦+---------------+¦
           +-----------------+
```

Note: This command does not have MBCS support.

Description

When this program starts up, answering "y" to the initial "Instructions?" prompt yields directions; typing "n" or <Enter> skips them. The **addbib** command then prompts for various bibliographic fields, reads responses from the terminal, and sends output records to a **database**. A null response (just <Enter>) means to leave out that field. A minus sign (-) means to go back to the previous field. A trailing backslash allows a field to be continued on the next line. The repeating "Continue?" prompt allows the user either to resume by typing "y" or <Enter>, to quit the current session by typing "n" or "q", or to edit the **database** with any system editor (**vi, ex, edit, ed**).

Flags

-a	Suppresses prompting for an abstract. Asking for an abstract is the default. Abstracts are ended with a **Ctrl-D**.
-p promptfile	Causes **addbib** to use a new prompting skeleton, defined in **promptfile**. This file should contain prompt strings, a tab, and the key-letters to be written to the **database**.

The most common key-letters and their meanings are given below. The **addbib** command insulates you from these key-letters, since it gives you prompts in English, but if you edit the bibliography file later on, you will need to know this information.

%A	Author's name
%B	Book containing article referenced
%C	City (place of publication)
%D	Date of publication
%E	Editor of book containing article referenced
%F	Footnote number or label (supplied by **refer**)
%G	Government order number
%H	Header commentary, printed before reference

%I	Issuer (publisher)
%J	Journal containing article
%K	Keywords to use in locating reference
%L	Label field used by **-k** option of **refer**
%M	Bell Labs Memorandum (undefined)
%N	Number within volume
%O	Other commentary, printed at end of reference
%P	Page number(s)
%Q	Corporate or Foreign Author (not reversed)
%R	Report, paper, or thesis (unpublished)
%S	Series title
%T	Title of article or book
%V	Volume number
%X	Abstract--used by **roffbib**, not by **refer**
%Y,Z	ignored by **refer**

Except for %A, each field should be given just once. Only relevant fields should be supplied.

Examples

An example is:

```
    %A    Bill Tuthill
    %T    Refer-- A Bibliography System
    %I    Computing Services
    %C    Berkeley
    %D    1982
    %O    UNX 4.3.5.
```

Files

promptfile Optional file to define prompting.

Related Information
See the following commands: "refer" in topic 1.1.365, "sortbib" in topic 1.1.433, "roffbib" in topic 1.1.382, "lookbib, indxbib" in topic 1.1.244.

1.1.15 adduser, users

Purpose
Adds, deletes and changes user and group information.

Syntax

```
                +- /usr/adm/user.cfile -+
/etc/adduser ---¦                       +---¦
/etc/users      +----- configfile ------+
```

Note: This command does not have MBCS support.

Description
The **adduser** command lets you add, change, or delete user and group information in the **/etc/passwd** and **/etc/group** files. To use the **adduser** command, you must be a member of the system group or have superuser authority (see "su" in topic 1.1.449).

The **adduser** command does all of its work in temporary files. When you enter the **quit** subcommand, the temporary files become the permanent files. The old versions of **/etc/passwd** and **/etc/group** are renamed **/etc/opasswd** and **/etc/ogroup**. If **adduser** is ended by an INTERRUPT (**Ctrl-C**), it removes the temporary files, and the system files remain as they were before the session. However any directories created still exist, so it may be necessary to remove directories after sending an INTERRUPT.

For configuration, **adduser** uses the file **/usr/adm/user.cfile**, the file specified with **configfile**, or the default parameters that follow:

Table 1-1. Configuration File Parameters

Paramete	**Default Value**	**Description**
udir	/u/	Prefix of user home directory names.
program	null	The name of the user login program.
minage	null	Minimum number of weeks that a password must be in effect before the password can be changed.
maxage	null	Maximum number of weeks that a password can be in effect before the password must changed.
siteinfo	null	Any site-specific information.
filesize	null	Size, in blocks, of the largest file that a user can make.
gname	staff	Name of the group that a user is initially assigned.
minid	200	Minimum number that can be assigned as a user or group ID.

maxid	60000	Maximum number that can be assigned as a user or group ID.
pfile	/etc/passwd	Name of the password file.
sitegrou	null	In a TCF cluster, the sites which a user may logon to.
gfile	/etc/group	Name of the group file.
owner	bin	Name of the owner of password and group files.
invalid	/usr/lib/sorr	Program for invalid accounts.

For information on how to use the **adduser** command, see *Managing the AIX Operating System*.

Notes:

1. Each group has a limit of 500 users with 8 character IDs. Shorter IDs may allow more users per group.

2. It is possible to delete a user who still owns files or to delete a group that still has members. However if you do this, it may cause problems later if the user name or group name is reused.

3. The **/etc/users** command is equivalent to **/etc/adduser**, and is included for compatibility with other UNIX systems. This is a separate command from **/usr/ucb/users**.

4. User names must be in 7-bit ASCII characters.

5. The unit of file size limit is 512-byte blocks.

Subcommands

add	Adds a new user or group.
change	Changes data for an existing user or group.
delete	Deletes an existing user or group.
help	Displays a summary of available commands. Entering a question mark (?) also works for help.
invalidate	Changes a user's shell to a do-nothing program.
quit	Updates files and exit.
show	Shows information about a user or group.

The initial letter of each subcommand is recognized as the subcommand name.

Examples
The following is a sample **/usr/adm/user.cfile**:

```
  pfile      /etc/passwd
  gfile      /etc/group
  owner      root
  minid      200
  maxid      1000
  udir       /u/
  program    /bin/sh
  gname      staff
  invalid    /usr/lib/sorry
```

Files

/usr/adm/user.cfile	Default configuration file.
/usr/adm/newuser.sys	Initialization shell file for added users.
/usr/adm/newuser.usr	Initialization shell file for added users.
/etc/passwd	Password file that identifies all known users.
/etc/group	Group file that identifies all known groups.
/etc/opasswd	Saved previous version of the password file.
/etc/ogroup	Saved previous version of the group file.

Related Information

See the **group** and **passwd** files in *AIX Operating System Technical Reference*.

See the discussions of **users**, **newuser.sys**, and **newuser.usr** in *Managing the AIX Operating System*.

1.1.16 admin

Purpose
Creates and initializes Source Code Control System (SCCS) files.

Syntax

```
                               +---------------------------+
         +- -n ------------+   ¦            +-- a --+      ¦
admin ---¦      +--------+ +---¦      +- l -¦¦      +----+ +---
         +- -i -¦¦       +-+   ¦      ¦¦    +- num -+    ¦ ¦
                +- name -+     +- -f -¦            ¦     +-+
                                      ¦      +--,--+     ¦
                                      ¦      one of      ¦
                                      ¦ +--------------+ ¦
                                      +-¦ b    j       +-+
                                        ¦ cnum mmodule ¦
                                        ¦ dSID n       ¦
                                        ¦ fnum qtext   ¦
                                        ¦ i    ttype   ¦
                                        +--------------+

    +-----------------------------------+   +----------+   +--- -r1.1 ---+
 ---¦ +---- -fv -----+   +---- -m ----+ +---¦         ²+---¦             +---
    +-¦              +---¦            +-+   +- -auser -+   +- -rnum.num -+
      +- -fvprogram -+   +- -mmrlist -+               ¦
                                             +--------+

    +----------+   +-----------------+
 ---¦          +---¦ +---- -y -----+ +--- s-file ---¦
    +- -tfile -+   +-¦             +-+            ¦
                     +- -ycomment -+   +----------+
```

¦ Do not put a blank between these items.
[2] If **-a** is never used to specify **users**,
any user can run **get -e** on the file.

1.1.16.1 To Change Existing SCCS Files:

```
                          +----------------------------+
         +------------+   ¦            +-- a --+       ¦
admin ---¦ +--------+ +---¦      +- l -¦¦      +----+ +---
         +-¦ -auser +-+   ¦      ¦     +- num -+    ¦ ¦
           ¦ -euser ¦¦    +- -f -¦¦          ¦     +-+
          ¦+--------+¦           ¦      +--,--+     ¦¦
          +----------+ ¦           ¦                 ¦¦
                       ¦     ¦ +--------------+ ¦¦
                       ¦     +-¦ b    j        +-+¦
                       ¦       ¦ cnum mmodule ¦   ¦
                       ¦       ¦ dSID n       ¦   ¦
                       ¦       ¦ fnum qtext   ¦   ¦
                       ¦       ¦ i    ttype   ¦   ¦
                       ¦       +--------------+   ¦
                       +--------------------------+

    +-------------------------+
    ¦            +-- a --+    ¦   +------------------+   +--------------+
 ---¦      +- l -¦¦      +-+ +---¦ +---- -fv -----+ +---¦ +--- -t ---+ +-- fi
    ¦      ¦     +- num -+ ¦ ¦   +-+- -fvprogram -+-+   +-¦          +-+
    +- -d -¦¦           ¦  +-+     +---- -dv -----+       +- -tfile -+   +----
     ¦     ¦      +--,--+   ¦¦
     ¦     ¦     one of     ¦¦
     ¦     ¦    +------+    ¦¦
     ¦     +----¦ b  j +---+¦
     ¦          ¦ c  m ¦    ¦
     ¦          ¦ d  n ¦    ¦
     ¦          ¦ f  q ¦    ¦
     ¦          ¦ i  t ¦    ¦
     ¦          +------+    ¦
     +----------------------+
```

1.1.16.2 To Check and Correct Damaged SCCS Files:

```
         one of
         +----+
admin ---¦ -z +-- file --¦
         ¦ -h ¦         ¦
         +----+ +------+


-----------------
¦ Do not put a blank between these items.
```

Description

The **admin** command creates new Source Code Control System (**SCCS**) files or changes specified parameters in existing SCCS files. These parameters control how the **get** command builds the files that you can edit. They also provide information about who can access the file, who can make changes, and when changes were made.

If the named **file** exists, **admin** modifies its parameters as specified by the flags. If it does not exist and you supply the **-i** or the **-n** flag, **admin** creates the new file and provides default values for unspecified flags. If you specify a directory name for **file**, **admin** performs the requested actions on all SCCS files in that directory (all files with the **s.** prefix). If you specify a **-** (minus) as a **file** name, **admin** reads standard input and interprets each line as the name of an SCCS file. An end-of-file character (**Ctrl-D**) ends input.

The **admin** command is most often used to create new SCCS files without setting parameters. See "Examples" in topic 1.1.16.3 for the syntax used to create an SCCS file with no parameters set in the new file.

If you are not familiar with the delta numbering system, see *AIX Operating System Programming Tools and Interfaces* for more information.

1.1.16.3 SCCS File Conventions

All SCCS file names must have the form **s.name**. New SCCS files are created with read-only permission. You must have write permission in the directory to create a file (see "chmod" in topic 1.1.67 for an explanation of file permissions). **admin** writes to a temporary x-file, which it calls **x.name**. The x-file has the same permissions as the original SCCS file if it already exists, and it is read-only if **admin** creates a new file. After successful completion of **admin**, the x-file is moved to the name of the SCCS file. This ensures that changes are made to the SCCS file only if **admin** does not detect any errors while it is running.

Directories containing SCCS files should be created with permission code 755 (read, write, and execute permissions for owner, read and execute permissions for group members and others). SCCS files themselves should be created as read-only files (444). With these permissions, only the owner can use non-SCCS commands to modify SCCS files. If a group can access and modify the SCCS files, the directories should include group write permission.

The **admin** command also uses a temporary lock file (called **z.name**), to prevent simultaneous updates to the SCCS file by different users. See "SCCS Files" in topic 1.1.186.1 for additional information on the **z.name** file.

The following table contains the header flags that can be set with the **-f** flag and unset with the **-d** flags (see page 1.1.16.3). The header flags control the format of the g-file created with the **get** command (see "SCCS Files" in topic 1.1.186.1 for details on the g-file).

Table 1-2. SCCS Header Flags

Header Flag	**Header Flag Purpose**
b	Lets you use the **-b** flag of a **get** command to create branch deltas.
c**num**	Makes **num** the highest release number that a **get -e** can use. The value of **num** must be less than or equal to 9999. (Its default value is 9999.)
f**num**	Makes **num** the lowest release number that a **get -e** can retrieve. **num** must be greater than 0 and less than 9999. (Its default value is 1.)
d**SID**	Makes **SID** the default delta supplied to a **get** command.
i	Treats the **No ID keywords (ge6)** message issued by the **get** or **delta** command as an error (see "Identification Keywords" in topic 1.1.186.2).
j	Permits concurrent **get** commands for editing the same SID of an SCCS file. This allows multiple concurrent updates to the same version of the SCCS file.
l**num**[,**num**]...	Locks the releases specified by **num**... against

	editing, so that a **get -e** against one of these releases fails. You can lock all releases against editing by specifying **-fla** and unlock specific releases with the **-d** flag.
n	Causes **delta** to create a null delta in any releases that are skipped when a delta is made in a new release. For example, if you make delta 5.1 after delta 2.7, releases 3 and 4 will be null. The resulting null deltas can serve as points from which to build branch deltas. Without this flag, skipped releases do not appear in the SCCS file.
q**text**	Substitutes **text** for all occurrences of the %Q% keyword in an SCCS text file retrieved by a **get** command. (See "Identification Keywords" in topic 1.1.186.2 for more information on keywords.)
m**module**	Substitutes **module** for all occurrences of the %M% keyword in an SCCS text file retrieved by a **get** command. The default **module** is the name of the SCCS file without the **s.** prefix.
t**type**	Substitutes **type** for all %Y% keywords in a g-file retrieved by a **get**.
v[**program**]	Makes **delta** prompt for Modification Request (MR) numbers as the reason for creating a delta. **program** specifies the name of an MR number validity checking program (see "delta" in topic 1.1.117). If **v** is set in the SCCS file, the **admin -m** flag must also be used, even if its value is null.

Flags

You can enter the flags and input file names in any order. All flags apply to all the files.

-auser Adds the specified **user** to the list of users who can make sets of changes, or **deltas**, to the SCCS file. **user** can be either a user name, a group name, or a group ID. Specifying a group name or number is the same as specifying the names of all users in that group. You can specify more than one **-a** flag on a single **admin** command line. If an SCCS file contains an empty user list, anyone can add deltas.

If a file has a user list, the creator of the file must be included in the list in order for the creator to make deltas to the file.

-dhdrflag Removes the specified header flag from the SCCS file. You can specify this flag only with existing SCCS files. You can also specify more than one **-d** flag in a single **admin** command. See Table 1-2 for the header flags that **admin** recognizes.

-euser Removes the specified **user** from the list of users allowed to make deltas to the SCCS file. Specifying a group ID is equivalent to specifying all **user** names common to that group. You can specify several **-e** flags on a single **admin** command line.

-fhdrflag[value] Places the specified header flag and value in the SCCS file. You can specify more than one header flag in a single **admin** command. See Table 1-2 for the header flags that **admin** recognizes.

-h Checks the structure of the SCCS file and compares a newly computed checksum with the checksum that is stored in the first line of the SCCS file. When the checksum value is not correct, the file has been improperly modified or has been damaged. This flag helps you detect both accidental damage and damage caused by the improper use of non-SCCS commands to modify SCCS files. The **-h** flag prevents writing to the file, so it cancels the effect of any other flags supplied. If an error message is returned indicating the file is damaged, use the **-z** flag to recompute the checksum. Then test to see if the file is corrected by using the **-h** flag again.

-i[name] Gets the text for a new SCCS file from **name**. This text is the first delta of the file. If you specify the **-i** flag but you omit the file name, **admin** reads the text from standard input until it reaches END OF FILE (**Ctrl-D**). If you do not specify the **-i** flag, but you do specify the **-n** flag, **admin** creates an empty SCCS file. **admin** can only create one file containing text at a time. If you are creating two or more SCCS files with one call to **admin**, you must use the **-n** flag. The SCCS files created are empty.

-m[mrlist] Specifies a list of Modification Request (MR) numbers to be inserted into the SCCS file as the reason for creating the initial delta. The **v** flag must be set. The MR numbers are validated if the **v** flag has a value (the name of an MR number validation program). **admin** reports an error if the **v** flag is not set or if MR validation fails.

-n Creates a new, empty SCCS file. Do not specify this flag when you use the **-i** flag.

-rnum.num Inserts the initial delta into **num.num**, the release and version, respectively. You can specify **-r** only if you also specify the **-i** or **-n** flag. If you do not specify this flag, the initial delta becomes release 1, version 1. Use this flag only when creating an SCCS file.

-t[file] Takes descriptive text for the SCCS file from **file**. If you use **-t** when creating a new SCCS file, you must supply a file name. In the case of existing SCCS files:

> Without a file name, **-t** causes removal of the descriptive text (if any) currently in the SCCS file. With a file name, **-t** causes text in the named file to replace the descriptive text (if any) currently in the SCCS file.

-y[comment] Inserts **comment** text into the initial delta in a manner identical to that of the **delta** command. Use this flag only when you create an SCCS file. If you do not specify a comment, **admin** inserts a line of the following form:

```
date and time created YY/MM/DD HH:MM:SS by login
```

-z Recomputes the SCCS file checksum and stores it in the first line of the SCCS file (see the **-h** flag on page 1.1.16.3).

Warning: Using **admin** with this flag on a damaged file can prevent future detection of the damage. This flag should only be used if the SCCS file is changed using non-SCCS commands because of a serious error.

Examples

1. To create an empty SCCS file named **s.prog.c**:

   ```
   admin  -n  s.prog.c
   ```

2. To convert an existing text file into an SCCS file:

   ```
   admin  -iprogram.c  s.prog.c
   ```

   This converts the text file **program.c** into the SCCS file **s.prog.c**. The original file remains intact, but it is no longer needed. You must rename or delete it before you can use the **get** command on **s.prog.c**.

Related Information

See the following commands: "delta" in topic 1.1.117, "ed, red" in topic 1.1.147, "get" in topic 1.1.186, "sccshelp" in topic 1.1.411, "prs" in topic 1.1.336 and "what" in topic 1.1.531.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See "Maintaining Different Versions of a Program" in *AIX Operating System Programming Tools and Interfaces*.

1.1.17 ali

Purpose
Lists mail aliases and their addresses.

Syntax

```
       +--- -alias /usr/lib/mh/MailAliases ----+   +-- -nolist --+
ali ---¦                                       +---¦ +---------+ +---
       +---------- -alias     file -----------+   +-¦ -list   +-+
                                     ¦              ¦ -nolist ¦
                +--------------------+              +---------+

    +-- -nonormalize --+   +-- -nouser --+
 ---¦      one of      +---¦   one of    +-- alias --¦
    ¦ +--------------+ ¦   ¦ +---------+ ¦           ¦
    +-¦ -normalize   +-+   +-¦ -user   +-+ +---------+
      ¦ -nonormalize ¦       ¦ -nouser ¦
      +--------------+       +---------+

ali -- -help --¦
```

Note: This command does not have MBCS support.

Description

The **ali** command is used to list mail aliases and the addresses that the aliases represent. **ali** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **ali** command searches the specified mail alias files for each given **alias**, and writes to standard output the address of each **alias**. If you specify the **-user** flag, **ali** interprets the **alias** arguments as actual addresses, searches the alias files for the addresses, and writes to standard output the aliases that contain definitions of the addresses. Thus, if you want to find the address of an alias, use the default **-nouser** flag. If you want to find the aliases that represent an address, use the **-user** flag.

Flags

-alias *file*	Specifies that *file* is a mail alias file to be searched for each given **alias**. The default alias file is **/usr/lib/mh/MailAliases**.
-help	Displays help information for the command.
-list	Displays each address on a separate line.
-nolist	Displays addresses separated by commas on as few lines as possible. This flag is the default.
-nonormalize	Does not attempt to convert local nicknames of hosts to their official host names. This flag is the default.
-normalize	Attempts to convert local nicknames of hosts to their official host names.
-nouser	Lists the addresses that the specified aliases represent.

	This flag is the default.
-user	Lists the aliases that contain the specified addresses. When the **-user** and **-nonormalize** flags are used together, the result may be a partial list of aliases that contain the specified addresses.

Files

/usr/lib/mh/MailAliases	The default mail alias file.
$HOME/.mh_profile	The MH user profile.
/etc/passwd	List of users.
/etc/group	List of groups.

Related Information

See other MH commands: "comp" in topic 1.1.85, "dist" in topic 1.1.131, "forw" in topic 1.1.174, "repl" in topic 1.1.369, "send" in topic 1.1.416 and "whom" in topic 1.1.539.

See the **mh-alias** and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.18 anet

Purpose
Runs a command on all sites in the cluster.

Syntax

```
        +--------------------+
anet ---¦                    +--- command ---¦
        +- -p processortype -+
```

Description
The **anet** command executes **command** on each site in the cluster, and then waits for each occurrence of **command** to finish execution. **command** is run concurrently at all sites. In a heterogeneous environment **command** will be executed on all sites with an available load module. On each site the **command** is re-evaluated to select a load module. If **command** begins with a '/' then a load module will be available if a load module of the appropriate type can be selected by using **command** as the file name of the load module. If **command** does not begin with a '/' then the environment is examined for the 'PATH' variable and a shell-like expansion is done on **command** using the **PATH** environment variable. If this results in a load module file of the appropriate type for the specified machine, then the command will be run on the selected machine.

When the **command** is executed the <LOCAL> alias is set to the local alias of the selected site. Use the **-p** option to run the command only on sites of the specified processor type.

Processor type is one of the following:

i386 - selects all AIX PS/2 sites.

i370 - selects all AIX/370 sites.

xa370 - selects all xa AIX/370 sites.

s370 - selects all non-xa370 AIX/370 sites.

Related Information
See the following commands: "loads" in topic 1.1.236, "site, sitechar, sitelocal, sitename, sitenum, ptn" in topic 1.1.426, "fast, fastsite" in topic 1.1.161, "onsite, on" in topic 1.1.306 and "printlocal" in topic 1.1.327.

See the discussion of **site** and **runvp** in the *AIX Operating System Technical Reference.*

Diagnostics
The **anet** command silently ignores sites which are down. The exit status of **anet** is the number of up sites for which **command** could not be successfully executed.

1.1.19 anno

Purpose
Annotates messages.

Syntax

```
        +-----------+   +------- cur ----------------------------------+
anno ---¦           +---¦ +---------- all -------------------------+ +---
        +- +folder -+   +-¦   +----------- sequence ------------+   +-+
                          +---¦  one of    +------------------+ +---+
                              ¦ +-------+  ¦       one of      ¦ ¦ ¦
                            ¦ +-¦ num   +--¦1+--------------+ +-+ ¦
                            ¦   ¦ first ¦  +-¦ :num   -prev +-+   ¦
                            ¦   ¦ prev  ¦    ¦ :+num  -cur  ¦     ¦
                            ¦   ¦ cur   ¦    ¦ :-num  -.    ¦     ¦
                            ¦   ¦ .     ¦    ¦ -num   -next ¦     ¦
                            ¦   ¦ next  ¦    ¦ -first -last ¦     ¦
                            ¦   ¦ last  ¦    +--------------+     ¦
                            ¦   +-------+                         ¦
                            +-------------------------------------+

    +--------------------+   +-- -noinplace --+   +----------------+
 ---¦                    +---¦     one of     +---¦                +---¦
    +- -component field -+   ¦ +------------+ ¦   +- -text string -+
                             +-¦ -inplace   +-+
                               ¦ -noinplace ¦
                               +------------+

anno --- -help ---¦


----------------
¦ Do not put a blank between these items.
```

Note: This command does not have MBCS support.

Description

The **anno** command is used to annotate messages with specified text and dates. The **anno** command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **anno** command annotates messages with the lines:

```
  field:date
  field:body
```

Although **dist**, **forw**, and **repl** enable you to perform annotations, their annotations are limited to adding distribution information to messages. The **anno** command enables you to perform arbitrary annotations. The annotation fields must contain alphanumeric characters and dashes only.

Flags

-component *field*   Specifies the field name for the annotation text. The field name must be a valid message field name, consisting of alphanumeric characters and dashes only. If you do not specify this flag, **anno** prompts you for the name of the field.

+folder msgs — Specifies the messages that you want to annotate. *msgs* can be several messages, a range of messages, or a single message. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default message is the current message in the current folder. If several messages are specified, the first message annotated becomes the current message. If you specify a folder, that folder becomes the current folder.

-help — Displays help information for the command.

-inplace — Forces annotation to be done in place in order to preserve links to the annotated messages.

-noinplace — Does not perform annotation in place. This flag is the default.

-text *string* — Specifies the text to be annotated to the messages.

Profile Entries

Current-Folder: — Sets your default current folder.

Path: — Specifies your **user_mh_directory**.

Files

$HOME/.mh_profile — The MH user profile.

Related Information

See other MH commands: "dist" in topic 1.1.131, "forw" in topic 1.1.174 and "repl" in topic 1.1.369.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See the "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.20 ap

Purpose
Parses and reformats addresses.

Syntax

```
      +--------------------+   +--- -normalize ---+   +--------------+
ap ---¦       one of       +---¦      one of      +---¦              +--- usei
      ¦ +----------------+ ¦   ¦ +--------------+ ¦   +- -width num -+
      +-¦ -form file     +-+   +-¦ -normalize   +-+                  +-----
        ¦ -format string ¦       ¦ -nonormalize ¦
        +----------------+       +--------------+

ap --- -help ---¦
```

Description

The **ap** command is used to parse and reformat addresses. **ap** is not designed to be run directly by the user; it is designed to be called by other programs. The **ap** command is typically called by its full path name, and is part of the Message Handling (MH) package.

The **ap** command parses each string specified as an address and attempts to reformat the string. The default output format for **ap** is the ARPA RFC822 standard. When the default format is used, **ap** displays an error message for each string it is unable to parse.

Flags

-form *file*	Reformats the given addresses into the alternate format described in *file*.
-format *string*	Reformats the given addresses into the alternate format specified by *string*. The default format string is: %<{error}%{error}:%{**address**}%|%(putstr(proper{**address**}))%
-help	Displays help information for the command.
-nonormalize	Does not attempt to convert local nicknames of hosts to their official host names.
-normalize	Attempts to convert local nicknames of hosts to their official host names. This flag is the default.
-width *num*	Sets the maximum number of columns that **ap** uses to display dates and error messages. The default is the width of the display.

Files

$HOME/.mh_profile	The MH user profile.
/usr/lib/mh/mtstailor	The MH tailor file.

Related Information
See other MH commands: "ali" in topic 1.1.17, "dp" in topic 1.1.138 and "scan" in topic 1.1.409.

See the **mh-alias**, **mh-format**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.21 apply

Purpose
Runs a named **command** on each argument, **arg** in turn.

Syntax

```
        +-------+   +------+
apply ---¦       +---¦      +--- command --- args ---¦
        +- -ac -+   +- -n -+                         ¦
                                         +--------+
```

Description

The **apply** command runs the named **command** on each argument **arg** in turn. Normally arguments are chosen singly; the optional number **n** specifies the number of arguments to be passed to **command**. If **n** is zero, **command** is run without arguments once for each **arg**. Character sequences of the form %**d** in **command**, where **d** is a digit from 1 to 9, are replaced by the **d**'th following unused **arg**. If any such sequences occur, **n** is ignored, and the number of arguments passed to **command** is the maximum value of **d** in **command**. The character '%' may be changed by the **-a** option.

Flags

-ac	Identifies the character **c** used instead of the '%'.
-n	Specifies the number of arguments to be passed to **command**.

Examples

1. A use similar to **ls**:

   ```
   apply echo *
   ```

2. To compare the 'a' files to the 'b' files:

   ```
   apply -2 cmp a1 b1 a2 b2 . . .
   ```

3. To run **who** 5 times:

   ```
   apply -0 who 1 2 3 4 5
   ```

4. To link all files in the current directory to the directory /usr/joe:

   ```
   apply 'ln %1 /usr/joe' *
   ```

Related Information
See the following commands: "sh, Rsh" in topic 1.1.420 and "xargs" in topic 1.1.544.

1.1.22 apropos

Purpose
Shows which manual sections contain keywords.

Syntax

```
apropos --- keyword ---¦
                     ¦
         +-----------+
```

Note: This command does not have MBCS support.

Description
The **apropos** command shows which manual sections contain instances of any of the given keywords in their title. Each word is considered separately. The case of letters is ignored. Words which are part of other words are considered; thus, when searching for **compile**, **apropos** also finds all instances of **compiler**.

The **apropos** command behaves identically to the **man** command with the **-k** flag.

Examples

1. The following command locates all manual pages that contain **password** in the title.

```
apropos password
```

2. The following example lists all manual titles containing **format**. Once the desired manual page has been listed, the user can view the documentation by entering the **man** command with the proper section (3s) and subroutine (**printf**).

```
apropos format
man 3s printf
```

Files

/usr/man/whatis	Data base that **apropos** searches.

Related Information
See the following commands: "man" in topic 1.1.258, "whatis" in topic 1.1.532 and "catman" in topic 1.1.50.

1.1.23 ar

Purpose
Maintains portable libraries used by the linkage editor.

Syntax

```
                    +----- m -----+   +------------------+
                  +-¦             +---¦¦ one of          +-+
                  ¦ ¦     +-----+ ¦   ¦  +---+           ¦ ¦
      +-------+   ¦ +- r -¦¦    +-+   +--¦ a +- posname -+ ¦
ar ---¦¦+---+ +---¦       +- u -+        ¦ b ¦             +-- library -- name
      +-¦ c +-+   ¦       one of         ¦ i ¦             ¦
        ¦ s ¦¦    ¦       +-----+        +---+             ¦           +-----
       ¦¦ l ¦¦    +-------¦ d q +--------------------------+
       ¦¦ v ¦¦            ¦ p t ¦
       ¦+---+¦            ¦ x h ¦
       +-----+            +-----+

ar -- w -- library --¦
```

¦ Do not put a blank between these items.

Description
The **ar** command combines one or more named files into a single **library** file written in **ar** archive format. When **ar** creates a library, it creates headers in a transportable format; when it creates or updates a library, it rebuilds the symbol table that the linkage editor (the **ld** command) uses to make efficient multiple passes over object file libraries. See the **ar** file entry in *AIX Operating System Technical Reference* for information on the format and structure of portable archives and symbol tables.

Flags

In an **ar** command, you must list all selected flags together on the command line without blanks between them. You must specify one from the set **dhmpqrtxw**. You can also specify any number of optional flags from the set **abcilsuv**. If you select a positioning flag (**a**, **b**, or **i**), you must also specify the name of a file within **library** (**posname**), immediately following the flag list and separated from it with a blank.

a posname	Positions the named files after the existing file identified by **posname**.
b posname	Positions the named files before the existing file identified by **posname**.
c	Suppresses the normal message that is produced when **library** is created.
d	Deletes the named files from the library.
h	Sets the modification times in the member headers of the named files to the current date and time. If you do not specify any file names, **ar** sets the time stamps of all member headers. The environment variables **LANG** and **LC_TIME** control the format of the archive date and time.

i posname	Positions the named files before the existing file identified by **posname** (same as **b**).
l	Places temporary files in the current (local) directory instead of directory **/tmp**.
m	Moves the named files to some other position in the library. By default, it moves the named files to the end of the library. Use a positioning flag (**abi**) to specify some other position.
p	Writes to the standard output the contents of the named **file**s or all files in a **library** if you do not specify any files.
q	Adds the named files to the end of the library. Positioning flags, if present, do not have any effect. This process does not check to see if the named files are already in the library. In addition, if you name the same file twice, it may be put in the library twice.
r	Replaces a named file if it already appears in the library. Since the named files occupy the same position in the library as the files they replace, a positioning flag does not have any additional effect. When used with the **u** flag (update), **r** replaces only files modified since they were last added to the library file. If a named file does not already appear in the library, **ar** adds it. In this case, positioning flags do affect placement. If you do not specify a position, new files are placed at the end of the library. If you name the same file twice, it may be put in the library twice.
s	Forces the regeneration of the library symbol table whether or not **ar** modifies the library contents. Use this flag to restore the library symbol table after using the **strip** command on the library.
t	Writes to the standard output a table of contents for the library. If you specify file names, only those files appear. If you do not specify any files, **t** lists all files in the library.
u	Copies only files that have been changed since they were last copied (see the **r** flag discussed previously).
v	Writes to standard output a verbose file-by-file description of the making of the new library. When used with the **t** flag, it gives a long listing similar to that of the **ls -l** command, described under "ls, lf, lr" in topic 1.1.252. When used with the **x** flag, it precedes each file with a name. When used with the **h** flag, it lists the member name and the updated modification times. The environment variables **LANG** and **LC_TIME** control the format of the archive date and time.
w	Displays the archive symbol table. Each symbol is listed

with the name of the file in which the symbol is defined.

x Extracts the named files by copying them into the current directory. These copies have the same name as the original files, which remain in the library. If you do not specify any files, **x** copies all files out of the library. This process does not alter the library.

Examples

1. To create a library:

   ```
   ar  vq  lib.a  strlen.o  strcpy.o
   ```

   If **lib.a** does not exist, this creates it and enters into it copies of the files **strlen.o** and **strcpy.o**. If **lib.a** does exist, this adds the new members to the end without checking for duplicate members. The **v** flag sets verbose mode, in which **ar** displays progress reports as it proceeds.

2. To list the table of contents of a library:

   ```
   ar  vt  lib.a
   ```

   This lists the table of contents of **lib.a**, displaying a long listing similar to **ls -l**. To list only the member file names, omit the **v** flag.

3. To replace or add new members to a library:

   ```
   ar  vr  lib.a  strlen.o  strcat.o
   ```

   This replaces the members **strlen.o** and **strcat.o**. If **lib.a** was created as shown in Example 1, the **strlen.o** member is replaced. A member named **strcat.o** does not already exist, so it is added to the end of the library.

4. To specify where to insert a new member:

   ```
   ar vrb strlen.o lib.a strcmp.o
   ```

   This adds **strcmp.o**, placing the new member before **strlen.o**.

5. To update a member if it has been changed:

   ```
   ar  vru  lib.a  strcpy.o
   ```

   This replaces the existing **strcpy.o** member, but only if the file **strcpy.o** has been modified since it was last added to the library.

6. To change the order of the library members:

   ```
   ar  vma  strcmp.o  lib.a  strcat.o  strcpy.o
   ```

   This moves the members **strcat.o** and **strcpy.o** to positions immediately after **strcmp.o**. The relative order of **strcat.o** and **strcpy.o** is preserved. In other words, if **strcpy.o** preceded **strcat.o** before the move, it still does.

7. To extract library members:

```
ar  vx  lib.a  strcat.o  strcpy.o
```

This copies the members **strcat.o** and **strcpy.o** into individual files named **strcat.o** and **strcpy.o**, respectively.

8. To extract and rename a member:

```
ar  p  lib.a  strcpy.o  >stringcopy.o
```

This copies the member **strcpy.o** to a file named **stringcopy.o**.

9. To delete a member:

```
ar  vd  lib.a  strlen.o
```

This deletes the member **strlen.o** from the library **lib.a**.

Files

/tmp/ar*	Temporary files.

Related Information

See the following commands: "backup" in topic 1.1.32, "ld" in topic 1.1.226, "lorder" in topic 1.1.245, "make" in topic 1.1.254, "nm" in topic 1.1.298, "size" in topic 1.1.427, and "strip" in topic 1.1.445.

See the **a.out** and **ar** files and **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.24 arithmetic

Purpose

Tests arithmetic skills.

Syntax

```
                            +- + - -+   +-- 10 ---+
/usr/games/arithmetic ---¦¦+---+ +---¦         +---¦
                            +-¦ + +-+   +- range -+
                              ¦ - ¦
                             ¦¦ x ¦¦
                             ¦¦ / ¦¦
                             ¦+---+¦
                             +-----+
```

```
-----------------
¦ Do not put a blank between these items.
```

Description

The **arithmetic** command displays simple arithmetic problems and waits for you to enter an answer. If your answer is correct, the program displays "Right!" and presents a new problem. If your answer is wrong, it displays "What?" and waits for another answer. Every 20 problems, **arithmetic** displays the number of correct and incorrect responses and the time required to answer.

The **arithmetic** command does not give the correct answers to the problems it displays. It provides practice rather than instruction in performing arithmetic calculations.

The **range** is a decimal number specifying the permissible range of all numbers (except answers). The default range is 10. At the start, all numbers within this range are equally likely to appear. If you make a mistake, the numbers in the problem you missed become more likely to reappear.

To quit the game, press INTERRUPT (**refer to keyboard definition**). The **arithmetic** command displays the final game statistics and exits.

Flags

+ Specifies addition problems.

- Specifies subtraction problems.

x Specifies multiplication problems.

/ Specifies division problems.

If you do not select any flags, **arithmetic** selects addition and subtraction problems. If you give more than one problem specifier (**(+)-x/**), the program mixes the specified types of problems in random order.

Examples

1. To drill on addition and subtraction of integers from 0 to 10:

   ```
   /usr/games/arithmetic
   ```

2. To drill on addition, multiplication, and division of integers from 0 to 50:

   ```
   /usr/games/arithmetic (+)x/  50
   ```

1.1.25 as

Purpose
Assembles a source file.

Syntax
For AIX/370:

```
      +----------------+   +--------+
as ---¦ +------------+ +---¦        +---¦
      +-¦ -C         +-+   +- file -+
        ¦ -D         ¦
        ¦ -ibuck     ¦
        ¦ -llistfile ¦
        ¦ -ofile     ¦
        ¦ -sn        ¦
        ¦ -t         ¦
        ¦ -T addr    ¦
        ¦ -V         ¦
        ¦ -xa        ¦
        +------------+
```

For AIX PS/2:

```
      +----------------+   +------------+
as ---¦ +------------+ +---¦ +--------+ +---¦
      +-¦ -a         +-+   +-¦        +-+
        ¦ -dl        ¦       +- file -+
        ¦ -iint      ¦
        ¦ -llistfile ¦
        ¦ -nbuck     ¦
        ¦ -R         ¦
        ¦ -s0        ¦
        ¦ -s1        ¦
        ¦ -s2        ¦
        +------------+
```

Description

Note: AIX assemblers are intended to support the compilers and may not have the full functionality of other assemblers specifically used for assembler language programming. See *AIX Programming Tools and Interfaces* for AIX PS/2 and AIX/370 restrictions.

The **as** command reads and assembles the named **file** (conventionally this file ends with a **.s** suffix). If you do not specify a **file**, **as** reads and assembles standard input. It creates an object file ending with a **.o** suffix.

Flags
The following flags are common to both AIX/370 and PS/2:

-llistfile	Generates a source listing. If the optional file is specified, the source listing is written to that file; otherwise it is written to standard output. Do not leave a space between **-l** and the **listfile**.
-ofile	Sets the name of the output module to file.

-T addr	Sets the origin address to addr.

The following flags are used only with AIX/370:

-C	Fixed format inspection (for example, card format).
-D	Use different version of the line and bcall built-in macros.
-ibuck	Sets the number of buckets in the symbol table to buck.
-sn	Tab size used only with **-C**.
-t	Time the assembler. Also counts and prints the number of lines processed and the number of lines expanded from macros.
-V	Prints the version number of the assembler on standard output.
-xa	Assemble module with XA/370 instruction.

The following flags are used only with PS/2:

-a	Does not automatically import any symbols that are referenced in the source code but are otherwise undefined. Issues an error message for this case.
-dl	Remove line number entries from the symbol table.
-iint	Sets error interval to the specified integer.
-nbuck	Sets the number of buckets in the symbol table to buck.
-R	Suppresses any **data** directives; all code is assembled in the text segment.
-s0	Generates the long form for all forward references and the short form, where possible, for backward references.
-s1	Runs one extra pass, in which most of the forward references are reduced to the shortest possible form.
-s2	Runs as many passes as are necessary to generate the short form for all qualifying forward references.

Files

file.s	Assembler source file.
file.o	Object file.

Related Information

The following commands: "cc" in topic 1.1.52, "ld" in topic 1.1.226.

The a.out file in *AIX Operating System Technical Reference*.

The discussion of **as** in the *AIX Operating System Programming Tools and Interfaces*.

1.1.26 at, batch

Purpose
Runs commands at a later time.

Syntax

```
      +-----------+            +- today -+   +-------------+   +--------+
at ---¦ +-------+ +--- time ---¦         +---¦             +---¦        +-+
      +-¦ -m -k +-+            +- date --+   +- increment -+   +- file -+ ¦
       ¦¦ -c -s ¦                                                       +---
       ¦+-------+                                                       ¦
       ¦+--- -l ---+                                                    ¦
      +¦           +----------------------------------------------------+
       +- -r job -+

           +-----------+   +--------+
         +-¦ +-------+ +---¦        +-+
         ¦ +-¦ -m -k +-+   +- file -+ ¦
batch ---¦   ¦ -c -s ¦                +---¦
         ¦   +-------+                ¦
         +------ -l ------------------¦
         +------ -r job --------------+
```

Description
The **at** and **batch** commands read the names of commands to be run at a later time from *file* or standard input:

at allows you to specify when the commands should be run.
batch runs jobs when the system load level permits.

Both **at** and **batch** mail you all output from standard output and standard error for the scheduled commands, unless you redirect that output. They also write the job number and the scheduled time to standard error.

Variables in the shell environment, the current directory, **umask**, and **ulimit** are retained when the commands are run. Open file descriptors, traps, and priority are lost.

You can use **at** if your name appears in the file **/usr/adm/cron/at.allow**. If that file does not exist, **at** checks the file **/usr/adm/cron/at.deny** to determine if you should be denied access to **at**. If neither file exists, only the superuser can submit a job. The allow/deny files contain one user name per line. If **at.allow** does exist, the superuser's login name must be included in it for the superuser to be able to use the command.

The required **time** parameter can be one of the following:

1. A number followed by an optional suffix. The **at** command interprets one- and two-digit numbers as hours. It interprets four digits as hours and minutes. The default order is the hour followed by the minute. You can also separate hours and minutes with a : (colon). The default order is **hour:minute**.

   In addition, you may specify a suffix of **am**, **pm**, or **zulu**. (**a.m.** can be abbreviated **a** and **p.m.** can be abbreviated **p**.) If you do not specify **am** or **pm**, **at** uses a 24-hour clock. The suffix **zulu** indicates that the time is **GMT** (Greenwich Mean Time).

2. The **at** command also recognizes the following keywords as special **time**s: **noon**, **midnight**, and **now**. (**noon** can be abbreviated **N** and **midnight** can be abbreviated **M**.) You can use the special word **now** only if you also specify a **date** or an **increment**. Otherwise, **at** tells you: **too late**.

The default format and English words recognized by the **at** and **batch** commands are replaced by their locale-specific equivalents found in the **environment** file. For more information, see the **setlocale** subroutine in the *AIX Technical Reference*.

The **at** command recognizes two special days, **today** and **tomorrow**, by default. The **today** is the default **date** if the specified time is later than the current hour; **tomorrow** is the default if the time is earlier than the current hour.

If the specified month is less than the current month (and a year is not given), next year is the default year.

The optional **increment** can be one of the following:

A + (plus sign) followed by a number and one of the following words **minute[s]**, **hour[s**], **day[s**], **week[s**], **month[s**], **year[s**] (or their non-English equivalents).

The special word **next** followed by one of the following words: **minute[s]**, **hour[s]**, **day[s]**, **week[s]**, **month[s]**, **year[s]** (or their non-English equivalents).

The **LANG** environment variable specifies the non-English equivalents of the English defaults.

The optional **file** is as follows:

Only shell scripts can be specified on the command line. If th program is a real program, it must come from standard input (or placed into a shell file).

Flags

-r job...	Removes **job**s previously scheduled by **at** or **batch**, where **job** is the number assigned by **at** or **batch**. If you do not have superuser authority (see "su" in topic 1.1.449), you can remove only your own jobs.
-c	Runs commands using **csh** as the shell.
-k	Runs commands using **ksh** as the shell, if **ksh** is present.
-l	Reports your scheduled jobs, including the job number.
-m	Mails a message to the user about the successful execution of the command.
-s	Run the commands using the Bourne shell.

Examples

1. To schedule the command **uucleanup** (which deletes old spool files) from the terminal, use a command similar to one of the following:

```
at  5 pm  tomorrow
/usr/lib/uucp/uucleanup
Ctrl-D
```

```
at  now  +  2  days
/usr/lib/uucp/uucleanup
Ctrl-D
```

These examples run **uucp** to delete the old **uucp** files at 5:00 PM tomorrow, or two days in the future, respectively.

2. To run **uucleanup** to delete the old **uucp** spool files at 3:00 in the afternoon on the 24th of January, use any one of the following commands:

```
echo  /usr/lib/uucp/uucleanup |  at  3:00  pm  January  24
echo  /usr/lib/uucp/uucleanup |  at  3pm  Jan  24
echo  /usr/lib/uucp/uucleanup |  at  1500  Jan  24
```

3. To run a job when the system load permits:

```
batch  <<!
longjob  2>&1  >outfile  |  mail  myID
!
```

This example shows the use of a ***here document*** to send standard input to **at** (see "Inline Input Documents" in topic 1.1.420.19). The order of redirections is important here, so that only error messages are sent into the pipe to the **mail** command. If you reverse the order, both standard error and standard output are sent to **outfile** (see the discussion of "Input and Output Redirection Using File Descriptors" in topic 1.1.420.20 for details).

4. To have a job reschedule itself, invoke **at** from within the shell procedure by including code similar to the following within the shell file:

```
echo  "sh  shellfile"  |  at  now  tomorrow
```

5. To list the jobs you have sent to be run later:

```
at  -l
```

6. To cancel jobs:

```
at  -r  103  227
```

This cancels jobs **103** and **227**. Use **at -l** to list the job numbers assigned to your jobs.

Files

/usr/adm/cron	Main cron directory.
/usr/adm/cron/at.allow	List of allowed users.
/usr/adm/cron/at.deny	List of denied users.
/usr/spool/cron/atjobs	Spool area.

Related Information
See the following commands: "cron" in topic 1.1.97, "kill" in topic 1.1.221, "mail, Mail" in topic 1.1.253, "nice" in topic 1.1.296, "ps" in topic 1.1.337 and "sh, Rsh" in topic 1.1.420.

See the **environment** special facility in *AIX Operating System Technical Reference*.

See "Running Commands at Pre-set Times" and "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.27 atq

Purpose
Prints the queue of jobs that are waiting to be run.

Syntax

```
       +--------+   +--------+
atq ---¦ +----+ +---¦        +---¦
       +-¦ -c +-+   +- name -+
         ¦ -  ¦¦
        ¦+----+¦
        +------+
```

Description

The **atq** command prints the queue of jobs that are waiting to be run at a later date. These jobs were created with the **at** command. With no flags, the queue is sorted in the order that the jobs will be executed.

If one or more names is provided, only files belonging to those users are displayed.

Flags

-c Sorts the queue by the time that the **at** command was issued.

-n Prints only the number of files currently in the queue.

Files

/usr/spool/cron/atjobs Spool area.

Related Information
See the following commands: "at, batch" in topic 1.1.26, "atrm" in topic 1.1.28 and "cron" in topic 1.1.97.

1.1.28 atrm

Purpose
Removes jobs that were created with the **at** command.

Syntax

```
        +--------+   +--------------+
atrm ---¦ +----+ +---¦ +----------+ +---¦
        +-¦ -f +-+   +-¦ jobno    +-+
          ¦ -i ¦       ¦ username ¦¦
          +----+      ¦+----------+¦
                       +------------+
```

Description

The **atrm** command removes jobs that were created with the **at** command. If one or more job numbers is specified, **atrm** attempts to remove only those jobs.

If one or more user names is specified, all jobs belonging to those users are removed. This form of invoking **atrm** is useful only to the superuser.

Flags

- Removes all jobs belonging to the person invoking **atrm**.

-f Suppresses all information about the jobs being removed.

-i Prompts before a job is removed; a 'y' response causes the job to be removed.

Files

/usr/spool/cron/atjobs	Spool area.

Related Information
See the following commands: "at, batch" in topic 1.1.26, "atq" in topic 1.1.27 and "cron" in topic 1.1.97.

1.1.29 awk, nawk, oawk

Purpose
Finds lines in files matching specified patterns and performs specified actions on them.

Syntax

```
                                 +-----------+   +----------+
      +----------+   +-'-¦          ²+---¦         ¦+-'-+   +---------------
awk ---¦          +---¦   +- pattern -    +- action -+¦  +---¦
      ¦          ¦¦   ¦  +---------------------------+  ¦   ¦
      +- -Fchar -+   +----------- -fprogfile -----------+   +- variable=valu

                                                             +--------------
```

```
-----------------
¦ The default char is a tab.
² The default pattern is every line.
¦ The default action is to print the line.
```

Description
The **awk** command is a more powerful pattern matching command than the **grep** command. It can perform limited processing on the input lines, instead of simply displaying lines that match. Some of the features of **awk** are:

    It performs convenient numeric processing
    It allows variables within actions
    It allows general selection of patterns
    It allows control flow in the actions
    It does not require any compiling of programs

Version 1.2.1 provides an enhanced version of the **nawk** (for "new awk"). This new version is similar to that provided by AIX Version 3.1 and recent releases of AT&T UNIX System V. The AIX 1.2 **awk** command is being provided as **oawk** (for "old awk"). The **awk** command is linked to **oawk** for backwards compatability with AIX 1.2.

The **nawk** command provides enhanced handling of files and pipes and better error messages for debugging. The **nawk** command also supports the Japanese language, which uses the multi-byte character set (MBCS) facilities of AIX 1.2.1. The **oawk** (**awk**) command has not been enhanced to support MBCS pattern matching. If this functionality is required, use **nawk**.

Differences between **oawk** and **nawk** are noted in the text where applicable. Except where noted, **awk** implies **oawk** and **nawk** in this document.

For a detailed discussion of **awk**, see *AIX Operating System Programming Tools and Interfaces*.

The **awk** command, reads **file**s in the order stated on the command line. If you specify a file name as **-** (minus) or do not specify a file name, **awk** reads standard input.

The **awk** command searches its input line by line for **pattern**s. When it finds a match, it performs the associated **action** and writes the result to standard output. In **nawk**, the **pattern** can contain Japanese characters. Enclose **pattern-action** statements on the command line in single quotation marks to protect them from interpretation by the shell.

The **awk** command first reads all pattern-action statements, then it reads a line of input and compares it to each pattern, performing the associated actions on each match. When it has compared all patterns to the input line, it reads the next line.

The **awk** command treats input lines as fields separated by spaces, tabs, or a field separator you set with the **FS** variable. Fields are referenced as **$1**, **$2**, and so on. **$0** refers to the entire line.

On the **awk** command line, you can assign **values** to variables as follows:

variable=value

Subtopics

1.1.29.1 Pattern-Matching Statements

Pattern-matching statements follow the form:

pattern { **action** }

If a **pattern** lacks a corresponding **action**, **awk** writes the entire line that contains the pattern to standard output. If an **action** lacks a corresponding **pattern**, it matches every line.

Actions

An action is a sequence of statements that follow C Language syntax. These statements can include:

statement	**format**
if	**if** (**conditional**) **statement** [**else statement**]
while	**while** (**conditional**) **statement**
for	**for** (**expression** ; **conditional** ; **expression**) **statement**
for	**for** (**variable** in **array**) **statement**¦
break	**break**
continue	**continue**
close (**filename**), **close** (**command**)	break connection between **print** and **filename** or **command** (**nawk** only)²
(assignment)	**variable** = **expression**
print	**print** [**expression-list**] [>**expression**]
printf	**printf format**[, **expression-list**] [>**expression**] (**oawk**)
printf	**printf format**[, **expression** ¦ 77 **expression** ¦ ¦ **command**] (**nawk**)¦
next	**next**
exit	**exit** [**expression**]4
(compound statement)	{**statement**...}

¦ **variable** may contain Japanese characters in **nawk** only.
² **filename** and **command** may contain Japanese characters in **nawk** only.
¦ **format**, **expression-list**, and **command** may contain Japanese characters in **nawk** only.
4 **expression** may contain Japanese characters in **nawk** only.

Statements can end with a semicolon, a new-line character , or the right brace enclosing the action.

If you do not supply an action, **awk** displays the whole line. Expressions can have string or numeric values and are built using the operators **+**, **-**, *****, **/**, **%**, a blank for string concatenation, and the C operators **++**, **--**, **+=**, **-=**, ***=**, **/=**, and **%=**.

In statements, variables may be scalars, array elements (denoted x[i]), or fields. Variable names can contain uppercase and lowercase alphabetic letters, underscores, and digits (0-9). **nawk** variable names may contain Japanese characters. Variable names cannot begin with a digit. Variables are initialized to the null string. Array subscripts may be any string; they do not have to be numeric. This allows for a form of associative memory. String constants in expressions should be enclosed in double quotation marks.

There are several variables with special meaning to **awk**. They include:

ARGC	Number of command-line arguments.
ARGV	Array of command-line arguments. **argv** may contain Japanese characters in **nawk** only.
FILENAME	The name of the current input file. In **nawk** only, may contain Japanese characters.
FNR	Record number in current file.
FS	Input field separator (default is a blank). This separator must be an ASCII character, in **oawk**, but may contain Japanese characters in **nawk**.
NF	The number of fields in the current input line (record).
NR	The number of the current input line (record).
OFMT	The output format for numbers (default **%.6g**).
OFS	The output field separator (default is a blank). This separator must be an ASCII character, in **oawk**, but may contain Japanese characters in **nawk**.
ORS	The output record separator (default is a new-line character). This separator must be an ASCII character, in **oawk**, but may contain Japanese characters in **nawk**.
RLENGTH	Length of string matched by match function.
RS	Controls the input record separator (default is **\n**).
RSTART	Start of string matched by match function.
SUBSEP	Subscript separator (default is **\034**).

Since the actions process fields, input blanks or white space are not preserved on the output.

The **printf** statement formats its expression list according to the format of the **printf** subroutine (see *AIX Operating System Technical Reference*), and writes it arguments to standard output, separated by the output field separator and terminated by the output record separator. You can redirect the output using the **print >** "filename" or **printf >** "filename" statements. An empty expression list stands for the whole line.

Example: To redirect the output of a print statement to a file named "myfile":

```
  awk '{print > filename}'
```

or

```
  awk ' {print > filename}'
```

You have two ways to designate a character other than white space to separate fields. You can use the **-Fc** flag on the **awk** command line, or you can start **progfile** with:

```
  BEGIN { FS = c }
```

Either action changes the field separator to **c**.

There are several built-in functions that can be used in **awk** actions.

atan2(y,x)	Takes arctangent of **y/x** in the range **-r** to **r** (**nawk** only).
cos(x)	Takes cosine of **x**, with **x** in radians (**nawk** only).
exp(n)	Takes the exponential of its argument.
getline	Reads the next line of standard input (**oawk**).

	The **nawk** version can also read from a pipe or an input file. An optional **var** parameter will store the input (**nawk** only).
gsub(r,s)	Substitute **s** for **r** globally in **$0**, return number of substitutions made (**nawk** only).
index(s,t)	Return first position of string **t** in **s**, or 0 if **t** is not present.
int(n)	Takes the integer part of its argument.
length	Returns the length of the whole line if there is no argument or the length of its argument taken as a string.
log(n)	Takes the base e logarithm of its argument.
log(x)	Takes natural (base **e**) logarithm of **x** (**nawk** only).
match(s,r)	Test whether **s** contains a substring matched by **r**, return index or 0; sets **RSTART** and **RLENGTH** (**nawk** only).
n=split(s,array,sep)	Splits string **s** into **array** [**1**] ...**array** [**n**] and returns number of elements. If present, **sep** is the field separator; otherwise, the variable **FS** is used.
rand( )	Returns random number **r**, where 0 <**r** <l (**nawk** only).
sin(x)	Takes sine of **x**, with **x** in radians (**nawk** only).
sqrt(n)	Takes the square root of its argument.
srand(x)	(nothing) **x** is new seed for **rand ( )** (**nawk** only).
substr(s,m,n)	Returns the substring of **s** which is **n** characters long, beginning at position **m**.
sprintf(fmt,expr,expr,...)	Formats the expressions according to the **printf** format string **fmt** and returns the resulting string.

Patterns

Patterns are arbitrary Boolean combinations of regular expressions and relational expressions (the **!**, ||, and **&&** operators and parentheses for grouping). You must start and end regular expressions with slashes (/). You can use regular expressions like those allowed by the **egrep** command (see "grep, egrep, fgrep" in topic 1.1.193), including the following special characters:

*****	Zero or more occurrences of the pattern.
+	One or more occurrences of the pattern.
?	Zero or one occurrences of the pattern.
|	Either of two statements.
( )	Grouping of expressions.

Isolated regular expressions in a pattern apply to the entire line. Regular expressions can occur in relational expressions. A pattern may consist of two patterns separated by a comma, in which case the action is performed on all lines between an occurrence of the first pattern and the next occurrence of the second. Regular expressions can contain extended characters with one exception: range constructs in character class specifications using square brackets cannot contain two-byte extended characters. Individual instances of extended characters can appear within square brackets; however, two-byte extended characters are treated as two separate one-byte characters.

There are two types of relational expressions that you can use. One has the form:

expression matchop regular-expression

where **matchop** is either: ~ (for "contains") or !~ (for "does not contain"). The second has the form:

expression relop expression

where **relop** is any of the six C relational operators: **<**, **>**, **<=**, **>=**, **==**, and **!=**. A conditional can be an arithmetic expression, a relational expression, or a Boolean combination of these.

You can use the special patterns **BEGIN** and **END** to capture control before the first and after the last input line is read, respectively. **BEGIN** may only be the first pattern in **profile**, and **END** may only be the last pattern.

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, add **0** to it. To force it to be treated as a string, append a null string (**""**).

1.1.29.2 nawk User-Defined Functions

A **nawk** program can contain user-defined functions. Such a function is defined by a statement of the form

```
  function name(parameter-list)  {
      statements
  }
```

A function definition can occur anywhere a pattern-action statement can. Thus, the general form of a **nawk** program is a sequence of pattern-action statements and function definitions separated by newlines or semicolons.

In a function definition, newlines are optional after the left brace and before the right brace of the function body. The parameter list is a sequence of variable names separated by commas; within the body of the function these variables refer to the arguments with which the function was called.

The body of a function definition may contain a **return** statement that returns control and perhaps a value to the caller. It has the form

```
  return expression
```

The **expression** is optional, and so is the **return** statement itself, but the returned value is undefined if none is provided or if the last statement executed is not a **return**.

For example, a function **"max"** might be called like this:

```
  { print max($1,max($2,$3)) }  # print maximum of $1, $2, $3

  function max(m, n) {
      return m > n ? m : n
  }
```

The variables **m** and **n** belong to the function **max**; they are unrelated to any other variables elsewhere in the program.

1.1.29.3 Updating oawk scripts to work using nawk

Note this example if you plan to convert your AIX 1.2 **oawk** scripts to run with **nawk**:

With AIX 1.2 **awk** or AIX 1.2.1 **oawk** you could write the following **awk** program:

```
  {
     LINE = $0   /* set line equal to input string */
     print $LINE  /* print out input */
  }
```

Note: **$** is only used with indirectly referenced variables.

With AIX 1.2.1 **nawk**, **$** is only used with a number or a variable whose value is a number, so you would need to write:

```
  {
     LINE = $0   /* set line equal to input string */
     print LINE   /* print out input */
  }
```

Notes: Do not use a **$** in front of a variable unless that variable has a value which is a number. For example, if **var="1"** then **$var** is really **$1**, which is the first word on the input line. However, if **var="a"** then **$var** is **$(a)**, which is not a valid field.

Flags

-f progfile	Searches for the patterns and perform the actions found in the file **progfile**.
-Fchar	Uses **char** as the field separator character (by default a blank).

Examples

1. To display the lines of a file that are longer than 72 characters:

   ```
   awk  "length  >72"  chapter1
   ```

   This selects each line of the file **chapter1** that is longer than 72 characters. **awk** then writes these lines to standard output because no **action** is specified.

2. To display all lines between the words **start** and **stop**:

   ```
   awk  "/start/,/stop/"  chapter1
   ```

3. To run an **awk** program (**sum2.awk**) that processes a file (**chapter1**):

   ```
   awk  -f  sum2.awk  chapter1
   ```

   The following **awk** program computes the sum and average of the numbers in the second column of the input file:

   ```
        {
           sum += $2
        }
   ```

```
  END {
        print "Sum: ", sum;
        print "Average:", sum/NR;
      }
```

The first action adds the value of the second field of each line to the variable **sum**. **awk** initializes **sum** (and all variables) to zero before starting. The keyword **END** before the second action causes **awk** to perform that action after all of the input file has been read. The variable **NR**, which is used to calculate the average, is a special variable containing the number of records (lines) that have been read.

4. To print the names of the users who have the C shell as the initial shell:

   ```
   awk  -F:  '/csh/{print  $1}'  /etc/passwd
   ```

5. To send output to the **more** command:

   ```
   awk  print|more  chapter1
   ```

6. To determine the correct number of characters, words, and lines in **chapter1**:

   ```
   awk  print|wc  chapter1
   ```

Related Information

See the following commands: "lex" in topic 1.1.229, "grep, egrep, fgrep" in topic 1.1.193 and "sed" in topic 1.1.415.

See the **printf** subroutine in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

See the discussion of **awk** and **uawk** in *AIX Operating System Programming Tools and Interfaces*.

1.1.30 axeb, ebxa

Purpose

axeb - Provides an ASCII-to-EBCDIC Translation
ebxa - Provides an EBCDIC-to-ASCII Translation

Syntax

```
axeb ----¦

ebxa ----¦
```

Note: This command is for the System/370 only.

Note: This command does not have MBCS support.

Description

The **axeb** command translates ASCII character data from standard input and writes the translated EBCDIC character data to standard output. The ASCII-to-EBCDIC translation is performed using the file named by the value of the **NLOUT** environment variable. If the **NLOUT** environment is not set or is invalid, the translation is performed using the default "round-trip" universal translation.

The **ebxa** command translates EBCDIC character data from standard input and writes the translated ASCII character data to standard output. The EBCDIC-to-ASCII translation is performed using the file named by the value of the **NLIN** environment variable. If **NLIN** environment is not set or is invalid, the translation is performed using the default "round-trip" universal translation.

axeb and **ebxa** do not report errors.

Files

/usr/lib/nls/nlout	Directory containing the ASCII/EBCDIC translation tables.
/usr/lib/nls/nlin	Directory containing the EBCDIC/ASCII translation tables.

Related Information

See the following commands: "genxlt" in topic 1.1.185, "vucp" in topic 1.1.527 and "uvcp" in topic 1.1.517.

1.1.31 back

Purpose

Plays backgammon.

Syntax

/usr/games/back ---¦

Description

The **back** game provides you with a partner for backgammon. You select one of three skill levels: beginner, intermediate, or expert. You may also choose to roll your own dice during your turns, and you are asked if you want to move first.

The points are numbered such that:

0 is the bar for removed white pieces

1 is white's extreme inner table

24 is brown's extreme inner table

25 is the bar for removed brown pieces

For details on how to make your moves, enter "y" when **back** asks "Instructions" at the beginning of the game. When it first asks "Move?", enter "?" to see a list of choices other than entering a numerical move.

When the game is finished, **back** asks you if you want to save game information. A "y" response stores game data in the file **back.log** in your current directory.

The **back** game plays only the forward game, even at the expert level. It will object if you try to make too many moves in a turn, but not if you make too few. Doubling is not implemented.

To quit the game, press INTERRUPT (**refer to keyboard definition**).

Files

/usr/games/lib/backrules	Rules file.
/tmp/b*	Log temp file.
back.log	Log file.

1.1.32 backup

Purpose
Backs up files.

Syntax

```
                                      +--- -9 ---+   +------+   +----- / -----
               +-----------------+  +-¦          +---¦      +---¦
/etc/backup ---¦ +------------+ +---¦ +- -level -+   +- -u -+   +- filesystem
               +-¦ -c         +-+   ¦   +------+
                 ¦ -f device  ¦¦    ¦ +-¦      +--- -i ----------------------
                ¦¦ -l num     ¦¦    +-¦ +- -b -+   +----- / ------+
                ¦¦ -v         ¦¦      +--- -m -----¦              +----------
                ¦¦ -C num     ¦¦                   +- filesystem -+
                ¦¦ -d density ¦¦
                ¦¦ -s length  ¦¦
                ¦¦ -r         ¦¦
                ¦¦ -q         ¦¦
                ¦+------------+¦
                +--------------+
```

Description
The **backup** command copies files in **backup** format to a backup medium, such as a magnetic tape or diskette.

There are three ways to back up data:

-i	Backs up specified files.
-level	Backs up an entire file system.
-m	Backs up an entire minidisk.

To back up specific files, use the **-i** flag. The **backup** command reads standard input for the files to be backed up. You can specify files by using the **find** command to generate a list of path names and pipe the list into the **backup** command.

To back up file system (inode), specify **-level** and **filesystem** to indicate the files you want to back up. You can use the **level** variable to back up all files on the system (a full backup) or only the files that have been modified since a specific full backup (an incremental backup). The possible levels are 0-9. If you do not supply a value for **level**, the default value is 9. If you specify 0, all files on the file system are backed up. If you specify **n**, all files modified since the last (**n**-1) backup are backed up. The levels, in conjunction with the **-u** flag, provide an easy way to maintain a hierarchy of incremental backups for each file system. For a discussion of backup strategy and the use of incremental backups, see *Managing the AIX Operating System*.

If you specify a **filesystem**, the name can be either the physical device name (the block or raw name) or the name of the directory on which the file system is normally mounted. When you specify a directory, the **backup** command reads the **/etc/filesystems** file, for the physical device name. In this case, the command also acquires values for other backup parameters from the **/etc/filesystems** file. If you do not specify a file system, the default is the root file system.

To back up a minidisk, use the **-m** flag. This option copies an exact image of the entire minidisk. You can specify the file system name of the minidisk. The default is the root file system. Because a backup by minidisk backs up an entire minidisk as an exact image, backing up a large minidisk with a small or sparsely used file system may take longer and require more backup medium than backing up a file system or specific files.

When you do not specify a backup device the **backup** command writes files to a default backup device. When you back up specific files, the system writes to the **/dev/rfd0**, unless you specify a device with the **-f** flag. When backing up a file system or minidisk, the **backup** command searches the **/etc/filesystems** file for a stanza matching the file system name you specified. If found, **backup** searches this stanza for a backup entry and writes to the device specified. Otherwise, the system writes to the **/dev/rfd0**, or the device specified with the **-f** flag.

The **backup** command recognizes a special syntax for the names of output files. If the argument is a range of file names, such as **/dev/rfd0** through **/dev/rfd3**, the **backup** command automatically goes from one drive in the range to the next. After exhausting all of the specified drives, the command halts and requests that new volumes be mounted.

Notes:

1. If the file system you are backing up is mounted and is not the root file system, the **backup** command unmounts the file system before it performs an inode backup and then remounts the file system before quitting. If the file systems you are backing up include the root file system, the **backup** command ensures that the other file systems are not in use. If one is, the command warns you of this use and quits.

2. When you are using the **backup** command on an internal tape backup unit, the **-C** option is ignored and the **-s** option cannot be used. Use the **-f** option to specify a PS/2 internal tape backup unit. For example:

   ```
   find . -print | backup -ivf/dev/rst0
   ```

   Use a 200 series minor device (for example, **/dev/rst204**) to restore a tape backed up with a 0 series (best for **backup**) minor device (for example, **dev/rst0**). Using the 0 series minor device number can, under some circumstances, result in a "READ ERROR on sector ####, assuming zeros" message, but should not affect the data being restored.

Warning: Be sure that the flags you specify match the backup medium. If the backup medium is not a disk or diskette, do not specify the **-l** flag. Similarly, if the backup medium is not a tape, do not specify the **-d** or **-s** flags. If you do specify flags that do not go with the medium, the **backup** command displays an appropriate error message and continues the backup.

Flags

-b	Enables users to back up files in ***unattended mode*** (when user input is not permitted). If any user input (such as, **Please insert volume 2**) is required, the command ends in an error. This enables users to set up a shell file that backs up files at night or at other times when the user is unavailable. The **-b** option can only be used with the **-i** backup mode.
-c	Creates a backup format compatible with AIX/RT. This flag

must be used when the **backup** command is used to transport data to AIX/RT and AIX RS/6000. This flag is ignored if the **-m** flag is also specified.

-Cnum	Specifies the number of blocks in 1K bytes to write in a single output operation. If you do not specify **num**, the **backup** command uses a default value appropriate for the physical device selected. Larger values of **num** result in longer physical transfers to tape devices. The value of the **-C** flag is always ignored when the **backup** command writes to diskette. In this case, it always writes in clusters that occupy a complete track.
-ddensity	Specifies the **density** of a tape medium in bytes per inch. The default density is 700 bytes per inch. The density you specify depends on your hardware; consult your tape drive operation manual.
-fdevice	Specifies the output device. Specify **device** as a file name (such as **/dev/rmt0**) to send output to the named device or specify **-** (minus) to send output to the standard output device. The **-** feature enables you to improve performance when backing up to streaming tape by piping the output of the **backup** command to the **dd** command.
-i	Reads standard input for the names of files to back up.
-lnum	Uses **num** as the limit of the total number of blocks to use on a diskette. The default value is the entire diskette (1440 blocks).
-m	Backs up the entire minidisk as an exact image.
-q	Indicates that removable medium is ready to use. When you specify this flag, the **backup** command proceeds without prompting you to prepare the backup medium or waiting for you to press the **Enter** key to continue. Same as the **-r** flag.
-r	Indicates that removable medium is ready to use. When you specify this flag, **backup** proceeds without prompting you to prepare the backup medium or waiting for you to press the **Enter** key to continue. Same as **-q** flag.
-slength	Specifies the **length** in feet of usable space on a tape medium. This length is a combination of the physical length and the number of tracks on the tape.
-u	Updates the time, date, and level of the backup in the **/etc/budate** file. This file provides the information needed for incremental backups.
-v	Reports on each phase of the backup as it is completed and gives regular progress reports during the longest phase.
-level	Specifies the backup **level** (0-9). The default **level** is 9.

You should use the **-u** flag when you do an incremental backup to ensure that information regarding the last date, time, and level of each incremental backup is written to the file **/etc/budate**.

Examples

1. To back up selected files:

   ```
   find  $HOME  -print  |  /etc/backup  -i  -v
   ```

   The **-i** flag causes the system to read from standard input the names of files to be backed up. The **find** command generates a list of files in the user's **$HOME** directory. This list is piped to the **backup** command as standard input. The **-v** flag causes a progress report to be displayed as each file is copied. The files are backed up on the default backup device.

2. To back up an entire non-replicated file system:

   ```
   /etc/backup  -0  -u  /xyz
   ```

   The **-0** level and the **/xyz** cause the system to back up the **/xyz** system. The file system **/xyz** and other mounted file systems are not backed up.

   The file system is backed up to the default device defined in the **backupdev** entry in the **/etc/filesystems** file. Otherwise, the files are backed up to the **/dev/rfd0** file. The **-u** flag causes the system to update the current backup level record in the **/etc/budate** file. Only the root file system is backed up; mounted file systems are not.

3. To back up all files modified since the last level 0 backup:

   ```
   /etc/backup  -1  -u  /xyz
   ```

4. To back up an entire minidisk:

   ```
   /etc/backup  -mf/dev/rmt1  /
   ```

   This command backs up the entire minidisk that contains the root file system. The **-f** flag causes the system to backup the minidisk to the streaming tape on **/dev/rmt1** instead of backing up to the default device. The **-m** flag **must** be used when backing up replicated file systems in order to retain commit counts.

5. To improve performance on the streaming tape, pipe the **backup** command to the **dd** command:

   ```
   backup -if- -C28 | dd of=/dev/rmt0 bs=28k
   ```

   The **backup** command backs up specific files (**-i**), directs the output to the standard output device (**f-**), and specifies an output size as 28 blocks (**-C28**). The output is piped to the **dd** command. The **dd** command copies the files to an output file, which is a streaming tape device (**of=/dev/rmt0**) and specifies a file size of 28 blocks (**bs=28b**). The file size in both commands must be the same. To restore these files, pipe the **dd** command to the **restore** command. The number of blocks specified in the **backup** option must not exceed the number of blocks available on the tape. Otherwise, even though there is no error message, not all of the data is backed up since the **dd** command does not work with multiple tapes.

6. To back up all files modified since the last level 0 backup to

450-foot tapes with a density of 350 bytes per inch:

```
backup -d 350 -s 4050 -l -u /
```

7. To back up all files modified since the last level 0 backup to 600-foot tapes, using the default density of 700 bytes per inch:

   ```
   backup -s 5400 -l -u /
   ```

8. To back up all files modified since the last level 0 backup to a DC 300 XL/P 450-foot, 310 oersted tape:

   ```
   backup -d 800 -s 450 -l -u /
   ```

Files

/etc/filesystems	Read for default parameters.
/etc/budate	Log for most recent backup dates.
/dev/rfd0	Default backup device.
/dev/rroot	Default file system.

Related Information

See the following commands: "find" in topic 1.1.165, "format, fdformat" in topic 1.1.172, "print" in topic 1.1.326, "restore" in topic 1.1.371 and "sh, Rsh" in topic 1.1.420.

See the **backup** and **filesystems** files and the **tape** special file in *AIX Operating System Technical Reference*.

See "Backing Up Files and File systems" in *Managing the AIX Operating System*.

1.1.33 banner (/usr/bin/banner)

Purpose
Writes character strings in large letters to standard output.

Syntax

```
                   +--------+
/usr/bin/banner ---¦        +--- string ---¦
                   +- -w n -+            ¦
                              +----------+
```

Note: This command does not have MBCS support.

Description
The **/usr/bin/banner** command writes the **string** to standard output in large letters. Each line in the output can be up to 10 uppercase or lowercase characters long. On output, all characters appear in uppercase, with the lowercase input characters appearing smaller than the uppercase input characters.

Flag

-w n	Output is width **n**. Default is 80 characters.

Examples

1. To display a banner at the work station:

   ```
   banner SMILE!
   ```

2. To display more than one word on a line, enclose the text in quotation marks:

   ```
   banner "Out to" Lunch
   ```

   This displays **Out to** on one line, and **Lunch** on the next.

3. To print a banner:

   ```
   banner We like Computers | print
   ```

Related Information
See the following commands: "echo" in topic 1.1.146 and "banner (/usr/games/banner)" in topic 1.1.34.

1.1.34 banner (/usr/games/banner)

Purpose
Writes character strings in extra large letters to standard output.

Syntax

```
                     +-------+   +--------------+
/usr/games/banner ---¦       +---¦              +---¦
                     +- -wn -+   +--- string ---+
                                              ¦
                                   +----------+
```

Note: This command does not have MBCS support.

Description
The **/usr/games/banner** command writes the **string** vertically to standard output in extra large letters. If the **string** argument is omitted, you are prompted to enter a string. The output, which should be sent to a printer, is up to 132 columns wide with no breaks between pages.

Flag

-wn Output is width **n**. Default is 132 characters. If the **n** argument is omitted, output is 132 columns wide.

Related Information
See the following commands: "echo" in topic 1.1.146 and "banner (/usr/bin/banner)" in topic 1.1.33.

1.1.35 basename, dirname

Purpose
Returns the ***basename*** of a string parameter.

Syntax

```
                     +----------+
basename -- string --¦          +---¦
                     +- suffix -+

dirname -- path --¦
```

Description
The **basename** command reads the **string** specified on the command line, deletes any prefix that ends with a **/** (slash), as well as any specified **suffix**, if it is present, and writes the remaining base file name to standard output. A **basename** of **/** is null and is considered an error.

The **dirname** command writes to standard output all except the last part of the specified **path** name (all but the last **/** and the part following it).

The **basename** and **dirname** commands are generally used inside command substitutions within a shell procedure to specify an output file name that is some variation of a specified input file name. For more information, see "Command Substitution" in topic 1.1.420.5.

Examples

1. To display the base name of a shell variable:

   ```
   basename  $WORKFILE
   ```

   This command displays the base name of the value assigned to the shell variable **WORKFILE**. If **WORKFILE** is set to **/u/tom/program.c**, then **program.c** is displayed.

2. To construct a file name that is the same as another file name, except for its suffix:

   ```
   OFILE=`basename $1 .c`.o
   ```

   This command assigns to **OFILE** the value of the first positional parameter (**$1**), but with its **.c** suffix changed to a **.o** suffix. If the **$1** parameter is **/u/tom/program.c**, **OFILE** becomes **program.o**. Because **program.o** is only a base file name, it identifies a file in the current directory. The `` (grave accents) perform command substitution.

3. To construct the name of a file located in the same directory as another:

   ```
   AOUTFILE=`dirname $TEXTFILE`/a.out
   ```

   This command sets the shell variable **AOUTFILE** to the name of an **a.out** file that is in the same directory as **TEXTFILE**. If **TEXTFILE** is **/u/fran/prog.c**, the value of **dirname $TEXTFILE** is **/u/fran**, and **AOUTFILE** becomes **/u/fran/a.out**.

Related Information

See "sh, Rsh" in topic 1.1.420.

1.1.36 bc

Purpose
Provides an interpreter for arbitrary-precision arithmetic language.

Syntax

```
      +--------+   +--------+
bc ---¦ one of +---¦        +---¦
      ¦ +----+ ¦   ¦        ¦
      +-¦ -c +-+   +- file -+
        ¦ -l ¦            ¦
        +----+      +-------+
```

Description
The **bc** command is an interactive process that provides unlimited precision arithmetic. It is a preprocessor for the **dc** command. The **bc** command invokes the **dc** command automatically, unless the **-c** (compile only) flag is specified. If the **-c** flag is specified, the output from the **bc** command goes to the standard output.

The **bc** command lets you specify an input and output base in decimal, octal, or hexadecimal (the default is decimal) notation. The command also has a scaling provision for decimal point notation. The syntax for the **bc** command is similar to that of the C language.

The **bc** command takes input first from the specified **file**. When the command reaches the end of the input **file**, it reads standard input.

The following description of syntax for the **bc** command uses the following abbreviations: **L** stands for the letters **a** through **z**; **E** for expressions; and **S** for statements.

When you enter **bc** expressions directly from the keyboard, press the END OF FILE (**Ctrl-D**) key to end the **bc** session and return to the shell command line.

Subtopics

1.1.36.1 Names

```
Simple variables: L
Array elements: L[E]
The words ibase, obase, and scale.
Comments are enclosed in /* and */.
```

1.1.36.2 Other Operands

```
Arbitrarily long numbers with optional sign and decimal point.
( E )
sqrt ( E )
length ( E )       number of significant decimal digits
scale ( E )         number of digits to the right of the decimal point
L ( E,...,E )
```

1.1.36.3 Operators

```
+ - * / % (remainder); ^ (power)
++ -- (prefix and postfix; apply to names)
== <= >= != <>
= = + =-  =*  =/  =%  =^
```

1.1.36.4 Statements

E
{ **S**;...;**S** }
if (**E**) **S**
while (**E**) **S**
for (**E;E;E**) **S**

Note: All **for** statements must have all three E's.

(null statement)
break

Note: The **quit** statement is interpreted when read, not when executed.

quit

1.1.36.5 Function Definitions

```
define L ( L,...,L ) {

     auto L,...,L
     S;...S
     return ( E )

}
```

1.1.36.6 Functions in -l Math Library

s(x)	sine
c(x)	cosine
e(x)	exponential
l(x)	log
a(x)	arctangent
j(n,x)	Bessel function

All function parameters are passed by value.

The value of a statement that is an expression is displayed unless the main operator is an assignment. A semicolon or new-line character separates statements. Assignments to **scale** control the number of decimal places printed on output and maintained during multiplication, division, and exponentiation. Assignments to **ibase** or **obase** set the input and output number radix respectively.

The same letter may refer to an array, a function, and a simple variable simultaneously. All variables are global to the program. "Auto" variables are pushed down during function calls. When you use arrays as function parameters or define them as automatic variables, empty square brackets must follow the array name.

Flags

-c Compiles **file**, but does not invoke the **dc** command.

-l Includes a library of math functions.

Examples

1. To use **bc** as a calculator:

   Enter:

   ```
   bc
   1/4
   ```

   The system responds:

   ```
   0
   ```

   Enter:

   ```
   scale = 1  /* Keep 1 decimal place  */
   1/4
   ```

   The system responds:

   ```
   0.2
   ```

   Enter:

   ```
   scale = 3  /* Keep 3 decimal places */
   1/4
   ```

   The system responds:

   ```
   0.250
   ```

Enter:

```
16+63/5
```

The system responds:

```
28.600
```

Enter:

```
(16+63)/5
```

The system responds:

```
15.800
```

Enter:

```
71/6
```

The system responds:

```
11.833
```

Enter:

```
1/6
```

The system responds:

```
0.166
```

You can type comments, which are enclosed in **/* and */**, but they are provided only for your information. The **bc** command displays the value of each expression, (except for assignments) when you press the **Enter** key.

2. To convert numbers from one base to another:

   Enter:

```
bc
obase = 16    /* Display numbers in Hexadecimal */
ibase = 8     /* Input numbers in Octal         */
12
```

   The system responds:

```
A
```

   Enter:

```
123
```

   The system responds:

```
53
```

   Enter:

```
          123456
```

The system responds:

```
          A72E
```

3. To write and run C-like programs:

   Enter:

```
          bc -l prog.bc
          e(2)    /*  e squared   */
```

   The system responds:

```
          7.38905609893065022723
```

   Enter:

```
          f(5)    /*  5 factorial */
```

   The system responds:

```
          120
```

   Enter:

```
          f(10)   /* 10 factorial */
```

   The system responds:

```
          3628800
```

   This interprets the **bc** program saved in **prog.bc**, then reads more **bc** statements from the work station keyboard. Starting the **bc** command with the **-l** flag makes the math library available. This example uses the **e** (exponential) function from the math library, and **f** is defined in the program file **prog.bc** as:

```
  /* compute the factorial of n */

  define f(n) {
     auto i, r;

     r = 1;
     for (i=2; i<=n; i++) r =* i;
     return (r);
  }
```

   The statement following a **for** or **while** statement must begin on the same line.

4. To convert an infix expression to reverse polish notation (RPN):

   Enter:

```
           bc -c
           (a * b) % (3 + 4 * c)
```

The system responds:

```
lalb* 3 4lc*+%ps.
```

In this example the **bc** infix-notation expression is compiled into one that the **dc** command can interpret. The **dc** command evaluates extended RPN expressions. In the compiled output, the **l** (ell) before each variable name is the **dc** subcommand that loads the value of the variable onto the stack. The **p** displays the value on top of the stack, and the **s.** discards the top value by storing it in register . (dot). You can save the RPN expression in a file for the **dc** command to evaluate later by redirecting the standard output of this command. For more details, see "Redirection of Input and Output" in topic 1.1.420.16.

Files

/usr/lib/lib.b	Mathematical library.
/usr/bin/dc	Desk calculator proper.

Related Information

See "dc" in topic 1.1.112.

1.1.37 bdiff

Purpose
Uses the **diff** command to find differences between very large files.

Syntax

```
                          +- 3500 -+   +------+
bdiff -- file1 -- file2 --¦         +---¦      +---¦
                          +- num --+   +- -s -+
```

Note: This command does not have MBCS support.

Description
The **bdiff** command compares **file1** and **file2** and writes information about lines that differ to standard output. If either file name is a **-** (minus), the **bdiff** command reads standard input. The **bdiff** command is used to find lines that must be changed in two files to make them identical. Its primary purpose is to process files that are too large for the **diff** command to process.

The **bdiff** command ignores lines common to the beginning of both files, splits the remainder of each file into **num**-line segments, and calls the **diff** command to compare the corresponding segments. In some cases, the 3500-line default for **num** is too large for the **diff** command. If this command fails, specify a smaller value for **num** and try again.

The output of the **bdiff** command has the same format as the **diff** command. The **bdiff** command adjusts line numbers to account for the segmenting of the files. Because of the file segmenting, the **bdiff** command does not necessarily find the smallest possible set of file differences.

Flag

-s	Suppresses error messages from the **bdiff** command (The **-s** flag does not suppress error messages from the **diff** command).

Example
To display the differences between **chap1** and **chap1.bak**:

```
  bdiff  chap1  chap1.bak
```

Files

/tmp/bd*	Temporary files.

Related Information
See the following command: "diff" in topic 1.1.124.

1.1.38 bellmail

Purpose
Sends messages to system users and displays messages to system users.

Syntax

```
                    +--------+   +- -f $HOME/mbox -+
                 +-¦ +----+ +---¦                 +-+
                 ¦ +-¦ -q +-+   +- -f file -------+ ¦
                 ¦   ¦ -r ¦¦                        ¦
bellmail ---¦   ¦¦ -p ¦¦                        +---¦
                 ¦   ¦+----+¦                        ¦
                 ¦   +------+                        ¦
                 +---- -e ------------------------+

   one of
+----------+   +------+
¦ bellmail +---¦      +-- user --¦
¦ rmail    ¦   ¦      ¦        ¦
+----------+   +- -t -+ +------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **bellmail** command with no flags writes to all stored mail addressed to your login name, one message at a time, to standard output. Following each message, the **bellmail** command prompts you with a **?** (question mark). Press the **Enter** key to display the next mail message, or enter one of the subcommands that control the disposition of the message (see "Subcommands").

When sending mail, you specify one or more **users**, and the **bellmail** command reads a message from standard input until you press the END OF FILE (**Ctrl-D**) key or enter a line containing only a . (period). The **bellmail** command prefixes the message with the sender's name and the date and time of the message (its postmark) and adds the message to the file **$HOME/.newmail** for each **user** specified on the command line.

The action of **bellmail** can be modified in two ways by manipulating the file **$HOME/.newmail**:

By default, the permissions on system mailbox files are read/write fo the owner only (see the **chmod** command). You may change the permissions on the **$HOME/.newmail** file to allow access by others and **bellmail** will preserve your chosen permissions.

You can edit the file to contain as its first line

```
Forward to person
```

This string causes all messages sent to **user** to be sent to **person** instead. The **Forward to** feature is especially useful for sending all of a person's mail to a particular machine in a network environment.

Note: If **sendmail** is installed, modifying **$HOME/.newmail** to forward messages does not work. In this case, you must add a **$HOME/.forward** file, as described in "mail, Mail" in topic 1.1.253.

To specify a recipient on a remote system which is available for **uucp** file transfer, prefix the system name and add an **!** (exclamation mark) before or after **user**. See "uucp" in topic 1.1.506 for details on addressing remote systems.

Flags

-e	Does not display any messages. This flag causes **bellmail** to return an exit value of 0 if the user has mail or an exit value of 1 if the user has no mail.
-f file	Reads mail from the named **file** instead of from the default system mailbox, **$HOME/.newmail**.
-p	Displays mail, without prompting for a disposition code. This flag does not delete, copy, or forward any messages. (For disposition codes, see "Subcommands.")
-q	Causes the **bellmail** command to exit when you press the INTERRUPT (**Ctrl-C**) key. Normally, pressing this key stops only the message being displayed. (In this case, the next message sometimes does not display until you enter the **p** subcommand.)
-r	Displays mail in first-in, first-out order.
-t	Prefixes each message with the names of **all** recipients of the mail. (Normally, only the individual recipient's name appears as addressee.)

Usually, **user** is a name recognized by the **login** command. It can also be the ASCII synonym that is automatically defined for any name that contains NLS code points.

If the system does not recognize one or more of the specified **user**s or if the **bellmail** command is interrupted during input, the command saves messages in the file **$HOME/dead.letter** to allow for editing and resending, unless the **sendmail** program is installed. In that case, mail to unknown users will be returned. If your login session is cut off while you are entering a message, and before you have completed sending, it will be saved in **$HOME/dead.letter**.

Subcommands
The following subcommands control message disposition:

+	Displays the next mail message (the same result as pressing the **Enter** key).
-	Displays the previous message.
d	Deletes the current message and displays the next message.
p	Displays the current message again.
s [file]	Saves the message in the named **file** instead of in the default mail file, **$HOME/mbox**.
w [file]	Saves the message, without its postmark, in the specified **file** instead of in the default mail file **$HOME/mbox**.

m user	Forwards the message to the named **user**.
q	Writes any mail not yet deleted to the file **$HOME/.newmail** and exits. Pressing the END OF FILE (**Ctrl-D**) key has the same effect.
x	Exits without changing $**HOME/.newmail**. Any changes or deletions you make during this use of **bellmail** will be rescinded.
!AIX-cmd	Runs the specified AIX command.
*****	Displays a subcommand summary.

Examples

1. To display mail:

   ```
   bellmail
   ```

   After the most recent message is displayed, a **?** (question mark) indicates that the **bellmail** command is waiting for you to enter one of the subcommands listed previously (for example, **+**, **-**, **d**, and **p**). Enter **help** or ***** (asterisk) to list the subcommands available.

2. To send mail to other users:

   ```
   bellmail tom rachel
   Don't forget the
   meeting tomorrow at 9:30.
   ```

   Ctrl-D

   In this example the system mails the message **Don't forget the meeting tomorrow at 9:30** to the users **tom** and **rachel**. The **Ctrl-D** indicates the end of the message but is not sent with the text.

3. To send a file to another user:

   ```
   bellmail fran <proposal
   ```

   This command sends the contents of the file **proposal** to **fran**. You can create **proposal** with an editor, which allows you to correct your mistakes before sending the message. You can also use this form of the **bellmail** command to send someone a copy of a text file.

4. To retrieve a file that was sent to you:

   ```
   bellmail
   ```

   This command displays, one at a time, the messages mailed to you. You need to look at them because the file you want was actually added to **$HOME/.newmail** as a message. You may see several other messages before seeing the file that was sent to you. If so, press the **Enter** key after the **?** prompt until the desired file appears. If you go too far, enter the **-** (minus) subcommand to go back a message at a time. After the **?** immediately following the file, enter:

   ```
   w mycopy
   ```

This subcommand saves a copy of the file sent to you in a file named **mycopy**, which resides in the current directory.

Note: You can use the **w** subcommand to save a copy of any message.

Files

/etc/passwd	To identify sender and locate **user**.
/$HOME/.newmail	Incoming mail for **user**.
$HOME/mbox	Saved mail.
$HOME/dead.letter	Non-mailable text.
/tmp/ma*	Temporary file.

Related Information
See the following commands: "login" in topic 1.1.241, "mail, Mail" in topic 1.1.253, "uucp" in topic 1.1.506, "sendmail, mailq, newaliases" in topic 1.1.417 and "write" in topic 1.1.541.

1.1.39 bfs

Purpose
Scans files.

Syntax

```
       +-----+
bfs ---¦     +-- file --¦
       +- - -+
```

Note: This command does not have MBCS support.

Description
The **bfs** command reads a **file** but does not allow any changes to be made to it; that is, it allows you to scan the file but not edit it.

The **bfs** command is basically a read-only version of the **ed** command, except that it can process much larger files and it has some additional subcommands. Input files can be up to 32K lines long, with up to 255 characters per line. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the 255-character limit can be reduced by as much as 50%, depending on the character code set being used.

The **bfs** command is usually more efficient than the **ed** command for scanning a file, because the file is not copied to a buffer. The **bfs** command is most useful for identifying sections of a large file where you can use the **csplit** command to divide it into more manageable parts for editing.

If you enter the **p** subcommand, the **bfs** commands prompts you with an ***** (asterisk). You can turn off prompting by entering a second **p**. The **bfs** command displays error messages when prompting is turned on.

Subtopics

1.1.39.1 Forward and Backward Searches

The **bfs** command supports all the address expressions described under "ed, red" in topic 1.1.147. In addition, you can instruct the **bfs** command to search forward or backward through the file, with or without ***wraparound***. If you specify a forward search with wraparound, the **bfs** command starts its search from the beginning of the file and after it reaches the end of the file it "wrapsaround" and continues its search from the beginning. If you specify a backward search with wraparound, it continues searching backwards from the end of the file after it reaches the beginning. The symbols for specifying the four types of search are as follows:

/pattern/	Searches forward with wraparound for the pattern.
?pattern?	Searches backward with wraparound for the pattern.
>pattern>	Searches forward without wraparound for the pattern.
<pattern<	Searches backward without wraparound for the pattern.

The **pattern** can contain Japanese characters.

The pattern matching routine of the **bfs** command differs somewhat from the one used by the **ed** command and includes additional features (see the **regcmp** subroutine in *AIX Operating System Technical Reference*). There is also a slight difference in mark names: only lowercase letters **a** through **z** may be used, and all 26 marks are remembered.

Flags

- Suppresses the display of the size of the file. Normally, the **bfs** command displays the size (in bytes) of the file being scanned.

Subcommands

The **e**, **g**, **v**, **k**, **n**, **p**, **q**, **w**, **=**, **!** and null subcommands operate as explained under "ed, red" in topic 1.1.147. Subcommands such as --, +++-, +++=, -12, and +4p are accepted. Note that **1,10p** and **1,10** both display the first ten lines. The **f** subcommand displays only the name of the file being scanned; there are no remembered file names. The **w** subcommand is independent of output diversion, truncation, or compression (see the **xo**, **xt**, and **xc** subcommands on page 1.1.39.1). Compressed output has strings of tabs and blanks reduced to one blank and blank lines suppressed.

The following additional subcommands are available:

xf file	Reads **bfs** subcommands from the **file**. When the **bfs** command reaches the end of file or receives an interrupt signal or if an error occurs, the **bfs** command resumes scanning the file that contains the **xf** subcommand. These **xf** subcommands may be nested to a depth of 10.
xo [file]	Sends further output from the **p** and null subcommands to the named **file**, which is created with read and write permission granted to all users. If you do not specify a **file** parameter, the **bfs** command writes to standard output. Each redirection to a file creates the specified file, deleting an existing file, if

necessary.

:label
Positions a **label** in a subcommand file. The **label** is ended with a new-line character. Blanks between the **:** (colon) and the start of the **label** are ignored. This subcommand may be used to insert comments into a subcommand file, since labels need not be referenced.

[addr1[,addr2]]xb/pattern/label
Sets the current line to the line containing **pattern** and jumps to **label** in the current command file if **pattern** is matched within the designated range of lines. The jump fails under any of the following conditions:

Either **addr1** or **addr2** is not between the first and last lines of the file.
addr2 is less than **addr1**.
The pattern does not match at least one line in the specified range, including the first and last lines.

This subcommand is the only one that does not issue an error message on bad addresses, so it may be used to test whether addresses are bad before other subcommands are run. The subcommand:

```
xb/^/label
```

is an unconditional jump.

The **xb** subcommand is allowed only if it is read from some place other than a workstation. If it is read from a pipe, only a downward jump is possible.

xt number
Truncates output from the **p** and null subcommands to **number** characters. The default **number** is 255.

xv[digit] [value]
Assigns the specified **value** to the variable named **digit** (0 through 9). You can put one or more spaces between **digit** and **value**. For example:

```
xv5  100
xv6  1,100p
```

assigns the value **100** to the variable **5** and the value **1,100p** to the variable **6**.

To reference a variable, put a **%** (percent sign) in front of the variable name. Given the preceding assignments for variables **5** and **6**, the following three subcommands:

```
1,%5p
1,%5
%6
```

each display the first 100 lines of a file.

To escape the special meaning of **%**, precede it with a \

(backslash). For example:

```
g/".*\%[cds]/p
```

matches and lists lines containing **printf** variables (**%c**, **%d**, or **%s**).

You can also use the **xv** subcommand to assign the first line of command output as the **value** of a variable. To do this, make the first character of **value** an ! (exclamation point), followed by the command name. For example:

```
xv5 !cat junk
```

stores the first line of the file **junk** in the variable **5**.

To escape the special meaning of ! as the first character of **value**, precede it with a \ (backslash). For example:

```
xv7 \!date
```

stores the value **!date** in the variable **7**.

xbz label	Tests the last saved exit value from a shell command and jumps to **label** in the current command file if the value is zero.
xbn label	Tests the last saved exit value from a shell command and jumps to **label** in the current command file if the value is not zero.
xc [switch]	Turns compressed output mode on or off. (Compressed output mode suppresses blank lines and replaces multiple blanks and tabs with a single space.) If **switch** is **1**, output from the **p** and null subcommands is compressed; if **switch** is **0** output is not suppressed. If you do not specify **switch**, the current value of **switch** reverses. Initially, **switch** is set to 0.

Related Information

See the following commands: "csplit" in topic 1.1.101 and "ed, red" in topic 1.1.147.

See the **regcmp** subroutine in *AIX Operating System Technical Reference*.

1.1.40 bib, listrefs

Purpose
Lists bibliographic reference items.

Syntax

```
                   +-------------------+   +----------+
bib, lisrefs ---¦ +---------------+ +---¦          +---¦
                   +-¦ -aa    -h     +-+   +- -ttype -+
                     ¦ -arnum -nstr  ¦¦
                    ¦¦ -ax    -o     ¦¦
                    ¦¦ -cstr  -pfile ¦¦
                    ¦¦ -ea    -sstr  ¦¦
                    ¦¦ -ex           ¦¦
                    ¦¦ -ernum        ¦¦
                    ¦¦ -f            ¦¦
                    ¦¦ -ifile        ¦¦
                    ¦+---------------+¦
                    +-----------------+
```

Note: This command does not have MBCS support.

Description
The **bib** command is a preprocessor for **nroff** or **troff** that formats citations and bibliographies. The input files (standard input default) are copied to the standard output, except for text between **[.** and **.]** pairs, which are assumed to be keywords for searching a bibliographic database. If a matching reference is found, a citation is generated replacing the text. References are collected, optionally sorted, and written out at a location specified by the user. Citation and reference formats are controlled by the **-t** option.

Reference databases are created using the **invert** utility.

Flags
The following options are available. Note that standard format styles (see the **-t** option) set options automatically. Thus, if a standard format style is used, the user need not indicate any further options for most documents.

-aa	Reduce authors' first names to abbreviations.
-arnum	Reverse the first **num** of the authors' names. If a number is not given, all authors' names are reversed.
-ax	Print authors' last names in Caps-Small Caps style. This style is used by certain Association for Computing Machinery (ACM) publications.
-cstr	Build citations according to the template **str**.
-ea	Reduce editors' first names to abbreviations.
-ex	Print editor' last names in Caps-Small Caps style (see **-ax** option).
-ernum	Reverse the first **num** editors' names. If a number is not given, all editors' names are reversed.

-f	Instead of collecting references, dump each reference immediately following the line on which the citation is placed (used for footnoted references).
-ifile	Process the indicated file, such as a file of definitions.
-h	Replace citations to three or more adjacent reference items with a hyphenated string. That is, 2, 3, 4, 5 becomes 2-5. This option implies the **-o** option.
-nstr	Turn off indicated options. **str** must be composed of the following letters: a f h o s x.
-o	Order contiguous citations according to the reference list before being printed (default).
-pfile	Instead of searching the file INDEX, search the indicated reference files before searching the system file. **files** is a comma-separated list of inverted indices, created using the **invert** utility.
-sstr	Sort references according to the template **str**.
-ttype	Use the standard macros and switch settings for the indicated style to generate citations and references. There are a number of standard styles provided. In addition, users can generate their own style macros.

The **listrefs** command formats an entire reference database file. Options to **listrefs** are the same as for **bib**.

The **bib** command was designed initially for use with **-ms** macros and uses two **-ms** macros (.ip and .lp) in its macro definitions. To use it with **-me** macros, prefix the file being sent to **nroff/troff** with the following macro definitions:

```
  .de IP
  .ip $1 $2
  ..
  .de LP
  .lp
  ..
```

A file named **bibmac.me** containing these macro definitions is in **/usr/lib/bmac**.

Files

INDEX	Inverted index for reference database
/usr/dict/papers/INDEX	Default system index
/usr/lib/bmac/bmac*	Formatting macro packages
/usr/tmp/bibr*	Scratch file for collecting references
/usr/tmp/bibp*	Output of pass one of **bib**

Related Information

See the "invert," "lookup," and "nroff" commands in the *AIX Commands Reference*.

1.1.41 biff

Purpose
Informs the system to notify user when mail arrives.

Syntax

```
        +- y -+
biff ---¦     +---¦
        +- n -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **biff** command informs the system whether you want to be notified when mail arrives during the current terminal session. The command

```
  biff y
```

enables notification; the command

```
  biff n
```

disables it. When mail notification is enabled, the header and first few lines of the message will be printed on your screen whenever mail arrives. A **biff y** command is often included in the file **.login** or **.profile** to be executed at each login.

The **biff** command operates asynchronously. For synchronous notification use the MAIL variable of **sh** or **ksh**, or the **mail** variable of **csh**.

Related Information
See the following commands: "csh" in topic 1.1.100, "sh, Rsh" in topic 1.1.420, "mail, Mail" in topic 1.1.253 and "comsat" in topic 1.1.87.

1.1.42 biod

Purpose

biod -NFS daemons.

Syntax

```
              +------------+
/etc/biod ---¦            +---¦
              +- nservers -+
```

Note: This command does not have MBCS support.

Description

The **biod** command starts **nservers** asynchronous block I/O daemons. This command is used on a NFS client to buffer cache handle read-ahead and write-behind.

When a file that is opened by a client is unlinked (by the server), a file with a name of the form **.nfsXXX** (where **XXX** is a number) is created by the client. When the open file is closed, the **.nfsXXX** file is removed. If the client crashed before the file can be closed, the **.nfsXXX** file is not removed.

Flag

nservers	The number of file system request daemons to start. This number should be based on the load expected on this server. Four is a typical number.

Files

.nfsXXX	Client machine pointer to an open-but-unlinked file.

1.1.43 bj

Purpose

Plays blackjack.

Syntax

/usr/games/bj ---¦

Note: This command does not have MBCS support.

Description

The **bj** game plays the role of the dealer in blackjack. The following rules apply.

The bet is $2 every hand. If you draw a **natural** (blackjack), you win $3. If the dealer draws a natural, you lose $2. If you and the dealer both have naturals, you exchange no money (a **push**). If the dealer has an ace showing , you can make an **insurance** bet on the chance that the dealer has a natural, winning $2 if the dealer has a natural and lose $1 if not. If you are dealt two cards of the same value, you can **double**, that is, play two hands, each of which begins with one of these cards, betting $2 on each hand. If the value of your original hand is 10 or 11, you can **double down**, that is, double the bet to $4 and receive exactly one more card in that hand.

Under normal play, you can draw a card (**hit**) as long as your cards total 21 or less. If the cards total more than 21, you **bust** and the dealer wins the bet. When you **stand** (decide not to hit), the dealer hits until he has a total of 17 or more. If the dealer busts, you win. If both you and the dealer stand, the one with the highest total wins. A tie is a push.

The **bj** command deals, keeps score, and asks the following questions at appropriate times: "?" (Do you want a hit?) "Insurance?" "Double?" "Double down?". To answer "yes," press "yes"; to answer "no" press the **Enter** key.

The dealer tells you whenever the deck is being shuffled and displays the **action** (total bet) and **standing** (total won or lost). To quit the game, press INTERRUPT (**refer to keyboard definition**). The **bj** command displays the final action and standing and exits.

1.1.44 bs

Purpose
Compiles and interprets modest-size programs.

Syntax

```
      +-----------------------+
bs ---¦        +------------+ +---¦
      +- file -¦            +-+
               +--- flag ---+
                          ¦
                 +--------+
```

Note: This command does not have MBCS support.

Description
This compiler/interpreter provides interactive program development and debugging. To simplify program testing, it minimizes formal data declaration and file manipulation, allows line-at-a-time debugging, and provides trace and dump facilities and run-time error messages.

The optional command line parameter **file** specifies a file of program statements that the compiler reads before it reads from the standard input. By default, statements read from this file are compiled for later execution. Likewise, statements entered from the standard input are normally executed immediately (see the **compile** keyword on page 1.1.44.1 and the **execute** keyword on page 1.1.44.1). Unless the final operation is assignment, the result of an immediate expression statement is displayed.

Additional command line **flags** can be passed to the program using the built-in functions **arg** and **narg** (explained in more detail on page 1.1.44.5).

Program lines must conform to one of the following formats:

  statement
  label statement

The **bs** interpreter accepts labeled statements only when it is compiling statements. A **label** is a name immediately followed by a colon. A label and a variable can have the same name. If the last character of a line is a \ (backslash), the statement continues on the following physical line.

A statement consists of either an expression or a keyword followed by zero or more expressions.

Subtopics

1.1.44.1 Statement Syntax

break	Exits the innermost **for** or **while** loop.
clear	Clears the symbol table and removes compiled statements from memory. A **clear** statement is always executed immediately.
compile [expr]	Causes succeeding statements to be compiled (overrides the immediate execution default). The optional expression is evaluated and used as a file name for further input. In this latter case, the symbol table and memory are cleared first. A **compile** statement is always executed immediately.
continue	Transfers control to the loop-continuation test of the current **for** or **while** loop.
dump [name]	Displays the name and current value of every global variable or, optionally, of the named variable. After an error or interrupt, a **dump** statement displays the number of the last statement and (possibly) the user-function trace.
exit [expr]	Returns to the system level. The expression is returned as process status.
execute	Changes to immediate execution mode (pressing the INTERRUPT [**Ctrl-C**] key has the same effect). This statement does not cause stored statements to execute (see **run** on page 1.1.44.1).

for name=expr expr statement

for name=expr expr

statement...

next

for expr, expr, expr statement

for expr, expr, expr

statement...

next	Repetitively performs, under the control of a named variable, a statement (first format) or a group of statements (second format). The variable takes on the value of the first expression, and is then increased by one on each loop until its value exceeds the value of the second expression. The third and fourth formats require three expressions separated by commas. The first of these is the initialization, the second is the test (true to continue), and the third is the loop-continuation action.

fun f ([a,...]) [v,...]

statement...

nuf	Defines the function name (**f**), parameters (**a**), and local variables (**v**) for a user-written function. Up to 10

parameters and local variables are allowed. Such names cannot be arrays, nor can they be I/O associated. Function definitions can not be nested.

freturn Signals the failure of a user-written function. Without interrogation, the **freturn** statement returns zero. (See the unary interrogation operator **?** discussed on page 1.1.44.3.) With interrogation, the **freturn** statement transfers to the interrogated expression, possibly bypassing intermediate function returns.

goto name Passes control to the compiled statement with the matching label.

ibase n Sets the input base to **n**. The only supported values for **n** are 8, 10 (the default), and 16. Hexadecimal values 10-15 are printed as alphabetic characters **a-f**. An **ibase** statement is always executed immediately.

if expr statement

if expr

   statement...

[else

   statement...**]**

fi Performs a statement (first format) or group of statements (second format) if the expression evaluates to nonzero. The strings 0 and "" (null) evaluate as zero. In the second format, an optional **else** statement allows a group of statements to be performed when the first group is not. The only statement permitted on the same line with an **else** statement is an **if** statement; only other **fi** statements can be on the same line with an **fi** statement. You can combine an **else** and an **if** statement into an **elif** statement. Only a single **fi** statement is required to close an **if**...**elif**...[**else**...] sequence.

include expr The expression must evaluate to the name of a file containing program statements. Such statements become part of the program being compiled. An **include** statement may not be nested, and is always executed immediately.

obase n Sets the output base to **n**. The only supported values for **n** are 8, 10 (the default), and 16. Hexadecimal values 10-15 are entered as alphabetic characters a-f. A leading digit is required when a hexadecimal number begins with an alphabetic character (that is, **f0a** must be entered as **0f0a**). Like an **ibase** statement, an **obase** statement is always executed immediately.

onintr label

onintr Provides program control of interrupts. In the first format, control passes to the label given, just as if a **goto** had been performed when the **onintr** statement was executed. The effect of the **onintr** statement is cleared

	after each interrupt. In the second format, pressing the INTERRUPT (**Ctrl-C**) key ends the **bs** command.
return [expr]	Evaluates the expression and passes the result back as the value of a function call. If you do not provide an expression, the function returns zero.
run	Passes control to the first compiled statement. The random number generator is reset. If a file contains a **run** statement, it should be the last statement; a **run** statement is always executed immediately.
stop	Stops execution of compiled statements and returns to immediate mode.
trace [expr]	Controls function tracing. If you do not provide an expression or if it evaluates to zero, tracing is turned off. Otherwise, a record of user-function calls/returns is written. Each return decreases by one the **trace** statement expression value.

while expr statement

while expr

   statement...

next	A **while** statement is similar to a **for** statement except that only the conditional expression for loop continuation is given.
! AIXcmd	Runs an AIX command, then returns control to the **bs** command.
#comment	Inserts a comment line.

1.1.44.2 Expression Syntax

name

Specifies a variable or, when followed immediately by a colon, a label. Names are composed of a letter (uppercase or lowercase) optionally followed by letters and digits. Only the first six characters of a name are significant. Except for names declared locally in **fun** statements, all names are global. Names can take on numeric (double float) values or string values or be associated with input/output (see the built-in function **open** on page 1.1.44.8).

name(**[expr[, expr]...)**

Calls function **name** and passes to it the parameters in parentheses. Except for built-in functions (listed in the following text), **name** must be defined in a **fun** statement. Function parameters are passed by value.

name[**expr[, expr]...]**

References either arrays or tables (see built-in function **table** on page 1.1.44.9). For arrays, each expression is truncated to an integer and used as a specifier for the name. The resulting array reference is syntactically identical to a name; **a[1,2]** is the same as **a[1][2]**. The truncated expressions must be values between 0 and 32767.

number

Represents a constant numerical value. This number can be expressed in integer, decimal, or scientific notation (it can contain digits, an optional decimal point, and an optional **e** followed by a possibly signed exponent).

string

Character string delimited by " " (double quotation marks). The \ (backslash) is an escape character that allows the double quotation mark (\"), new-line character (\n), carriage return (\r), backspace (\b), and tab (\t) characters to appear in a string. When not immediately followed by these special characters, \ stands for itself.

(*expr***)**

Parentheses alter the normal order of evaluation.

(*expr*, *expr[*, *expr]*...**) [***expr***]**

The bracketed expression outside the parentheses functions as a subscript to the list of expressions within the parentheses. List elements are numbered from the left, starting at zero. The expression:

```
  (False, True) [ a == b ]
```

has the value **True** if the comparison is true.

expr op expr

Except for the assignment, concatenation, and relational operators, both operands are converted to numeric form before the operator is applied.

1.1.44.3 Unary Operators

?expr	The interrogation operator (**?**) tests for the success of the expression rather than its value. It is useful for testing end of file, for testing the result of the **eval** built-in function, and for checking the return from user-written functions (see **freturn** on page 1.1.44.1). An interrogation **trap** (end of file, for example), causes an immediate transfer to the most recent interrogation, possibly skipping assignment statements or intervening function levels.
-expr	Negates the expression.
++name	Increases by one the value of the variable (or array reference).
--name	Decreases by one the value of the variable.
!expr	The logical negation of the expression.

1.1.44.4 Binary Operators (in increasing precedence)

= The assignment operator. The left operand must be a name or an array element. It acquires the value of the right operand. The assignment operator binds right to left; all other operators bind left to right.

_ The concatenation operator (the underline character).

& | Logical AND operator and logical OR operator. The result of the following:

    expr & **expr**

is 1 (true) only if both of its parameters are nonzero (true); the result is 0 (false) if one or both of its parameters are 0 (false).

The result of this:

    expr | **expr**

is 1 (true) if one or both of its expressions are nonzero (true); the result is 0 (false) only if both of its expressions are 0 (false). Both operators treat a null string as a zero.

< <= > >= == != The relational operators (< less than; <= less than or equal to; > greater than; >= greater than or equal to; == equal to, != not equal to) return 1 if the specified relation is true. They return 0 (false) otherwise. Relational operators at the same level extend as follows: **a>b>c** is the same as **a>b & b>c**. A string comparison is made if both operands are strings. The comparison is based on the collating sequence specified in the environment variables **LANG** and **LC_COLLATE**.

+ - Addition and subtraction operators.

*** / %** Multiplication, division, and remainder operators.

^ Exponentiation operator.

1.1.44.5 Functions Dealing With Arguments

arg(i)	Returns the value of the **i**-th actual argument at the current function call level. At level zero, **arg** returns the **i**-th command-line argument. For example, **arg(0)** returns **bs**.
narg( )	Returns the number of arguments passed. At level zero, it returns the command line argument count.

1.1.44.6 Mathematical Functions

abs(x)	Returns the absolute value of **x**.
atan(x)	Returns the arctangent of **x**.
ceil(x)	Returns the smallest integer not less than **x**.
cos(x)	Returns the cosine of **x**.
exp(x)	Returns e raised to the power **x**.
floor(x)	Returns the largest integer not greater than **x**.
log(x)	Returns the natural logarithm of **x**.
rand( )	Returns a uniformly distributed random number between zero and one.
sin(x)	Returns the sine of **x**.
sqrt(x)	Returns the square root of **x**.

1.1.44.7 String Functions

size(s) Returns the size (length in bytes) of **s**.

format(f, a) Returns the formatted value of **a**, **f** being a format specification string in the style of the **printf** subroutine. Use only the **%...f**, **%...e**, and **%...s** formats.

index(x, y) Returns a number that is the first position in **x** containing a character that any of the characters in **y** matches. If there is no match, the **index** function yields zero. For two-byte extended characters, the **index** function returns the location of the first byte.

trans(s, f, t) Translates characters in the source string **s** which match characters in **f** with characters having the same position in **t**. Source characters that do not appear in **f** are copied unchanged into the translated string. If string **f** is longer than **t**, source characters that match characters found in the excess portion of **f** do not appear in the translated string.

subst(s, start, length)
Returns the substring of **s** defined by **start**ing position and **length**.

match(string, pattern)

mstring(n) This function returns the number of characters in **string** that match **pattern**. The characters ., *****, **?** [,], ^ (when inside square brackets), \(and \) have the following special meanings (see "ed, red" in topic 1.1.147 for a more detailed discussion of this special notation):

. Matches any character except the new-line character.

* Matches zero or more occurrences of the pattern element that it follows (for example, **.*** matches zero or more occurrences of any character except the new-line character).

$ Specifies the end of the line.

[.-.]

[...] Matches any one character in the specified range ([.-.]) or list ([...]), including the first and last characters.

[^.-.]

[^...] Matches any character except the new-line character and the remaining characters in the range or list. A circumflex (^) has this special meaning only when it immediately follows the left bracket.

```
[].-.]

[]...] Matches ] or any character in the list.  The
       right square bracket does not terminate such
       a list when it is the first character within
       the list (after an initial ^, if any).

\(...\)
       Marks a substring and matches it exactly.
```

To succeed, a pattern must match from the beginning of the string. It also matches the longest possible string. Consider, for example:

```
  match('a123ab123',".*\([a-z]\)") == 6
```

In this instance, **.*** matches **a123a** (the longest string that precedes a character in the range a-z); **\([a-z]\)** matches **b**, giving a total of six characters matched in the string. In an expression such as **[a-z]**, the minus means "through" according to the current collating sequence. A collating sequence may define equivalence classes for use in character ranges. See "Introduction to International Character Support" in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

Note: If you have selected a specific locale (using either the **LANG** or **LC_COLLATE** environment variable), it may not be possible to specify an equivalence class of characters using a range expression. In such cases, use a character class expression rather than a standard range expression. For information about character class expressions, see the note on page 1.1.147.1.1 under the **ed, red** command.

The **mstring** function returns the **n**th substring in the last call to **match** (**n** must be between 1 and 10 inclusive).

1.1.44.8 File-Handling Functions

open(name, file, mode)

close(name) — The **name** parameter must be a legal variable name (passed as a string). For the **open** function, the **file** parameter may be:

- A **0**, **1**, or **2** for standard input, output, or error output, respectively
- A string representing a file name
- A string beginning with an !, representing a command to be run (via the **sh -c**) command.

The **mode** flag must be either **r** (read), **w** (write), **W** (write without new-line character), or **a** (append). After a **close function**, the **name** becomes an ordinary variable. The initial associations are:

```
open("get", 0, "r")
open("put", 1, "w")
open("puterr", 2, "w")
```

access(p, m) — Performs the **access** system call. Parameter **p** is the path name of a file; **m** is a bit pattern representing the requested mode of access. This function returns a 0 if the request is permitted and -1 if it is denied. (See *AIX Operating System Technical Reference* for an extensive discussion of this system call.)

ftype(s) — Returns a single character indicating file type: **f** for regular file, **p** for FIFO (named pipe), **d** for directory, **b** for block special, or **c** for character special.

1.1.44.9 Table Functions

table(name, size)

A table in the **bs** compiler is an associatively accessed, one-dimensional array. Subscripts (called keys) are strings (numbers are converted). The **name** parameter must be a **bs** variable name (passed as a string). The **size** parameter sets the minimum number of elements to be allocated. On table overflow, the **bs** compiler writes an error message.

item(name, i)

key( )

The **item** function accesses table elements sequentially (in normal use, there is an orderly progression of key values). Where the **item** function accesses values, the **key** function accesses the subscript of the previous **item** call. The **name** parameter should not be quoted. Since exact table sizes are not defined, the interrogation operator should be used to detect end-of-table; for example:

```
table("t",100)
.
.
.
#If word contains "party", the following expression
#adds one to the count of that word:
++t[word]
.
.
.
# To display the key/value pairs:
for i=0, ?(s=item(t, i)), ++i if key() put=key()_":"_s
```

iskey(name, word)

Tests whether the key **word** exists in the table **name** and returns 1 for true or 0 for false.

1.1.44.10 Miscellaneous Functions

eval(string)

The string parameter is evaluated as an expression. The function is handy for converting numeric strings to numbers. The **eval** function can also be used as a crude form of indirection, as in:

```
name = "xyz"
eval("++"_name)
```

which increments the variable **xyz**.

In addition, **eval** preceded by the interrogation operator permits you to control **bs** compiler error conditions. For example:

```
?eval("open(\"X\",\"XXX\", \"r\")")
```

returns the value 0 if there is no file named **"XXX"** (instead of halting your program). The following performs a **goto** to the label **L:** (if it exists):

```
label="L:"
if!(?eval("goto"_label))puterr="no label"
```

plot(request, args)

The **plot** function produces output on devices recognized by the **tplot** command. Some requests do not apply to all plotters. All requests except 0 and 12 are implemented by piping characters to the **tplot** command. The requests are as follows:

Requests	***Function***
plot(0, term)	Causes further **plot** output to be piped into the **tplot** command with a flag of **-Tterm**.
plot(1)	"Erases" the plotter.
plot (2, string)	Labels the current point with **string**.
plot(3, x1, y1, x2, y2)	Draws a line between (**x1, y1**) and (**x2, y2**).
plot(4, x, y, r)	Draws a circle with center (**x, y**) and radius **r**.
plot(5, x1, y1, x2, y2, x3, y3)	Draws an arc (counterclockwise) with center (**x1, y1**) and endpoints (**x2, y2**) and (**x3, y3**).
plot(6)	Not implemented.
plot(7, x, y)	Makes the current point at (**x, y**).
plot(8, x, y)	Draws a line from the current point to (**x, y**).

plot(9, x, y)	Draws a point at (**x, y**).
plot(10, string)	Sets the line mode to **string**.
plot(11, x1, y1, x2, y2)	Makes (**x1, y1**) the lower left corner of the plotting area and (**x2, y2**) the upper right corner of the plotting area.

Requests	***Function***
plot(12, x1, y1, x2, y2)	Causes subsequent **x** (**y**) coordinates to be multiplied by **x1** (**y1**) and then added to **x2** (**y2**) before they are plotted. The initial scaling is **plot(12, 1.0, 1.0, 0.0, 0.0)**.

last()	In immediate mode, the **last** function returns the most recently computed value.

Related Information

See the following commands: "ed, red" in topic 1.1.147 and "sh, Rsh" in topic 1.1.420.

See the **access** system call, the **printf** subroutine, and the **plot** file in *AIX Operating System Technical Reference.*

1.1.45 bugfiler

Purpose
Intercepts, summarizes and stores bug reports.

Syntax

```
/usr/lib/bugfiler --- maildirectory ---¦
```

Note: This command does not have MBCS support.

Description

The **bugfiler** command is a program to automatically intercept bug reports, summarize them and store them in the appropriate sub directories of the mail directory specified on the command line or the (system dependent) default. It is designed to be compatible with the Rand MH mail system. **bugfiler** is normally invoked by the mail delivery program through **aliases** with a line such as the following in **/usr/lib/aliases**:

```
    bugs:"|/usr/lib/bugfiler /usr/bugs/mail"
```

It reads the message from the standard input or the named file, checks the format and returns mail acknowledging receipt or a message indicating the proper format. Valid reports are then summarized and filed in the appropriate folder; improperly formatted messages are filed in a folder named "errors." Program maintainers can then log onto the system and check the summary file for bugs that pertain to them. Bug reports should be submitted in RFC822 format and must contain the following header lines to be properly indexed:

```
    Date: <date the report is received>
    From: <valid return address>
    Subject: <short summary of the problem>
    Index: <source directory>/<source file> <version> [Fix]
```

In addition, the body of the message must contain a line which begins with "Description:" followed by zero or more lines describing the problem in detail and a line beginning with "Repeat-By:" followed by zero or more lines describing how to repeat the problem. If the keyword "Fix" is specified in the "Index" line, then there must also be a line beginning with "Fix:" followed by a **diff** of the old and new source files or a description of what was done to fix the problem.

The "Index" line is the key to the filing mechanism. The source directory name must match one of the folder names in the mail directory. The message is then filed in this folder and a line appended to the summary file in the following format:

```
<folder name>/<message number>    <Index info>
                                  <Subject info>
```

The bug report may also be redistributed according to the index. If the file **maildir**/.redist exists, it is examined for a line beginning with the index name followed with a tab. The remainder of this line contains a comma-separated list of mail addresses which should receive copies of bugs with this index. The list may be continued onto multiple lines by ending each but the last with a backslash (\).

Files

/usr/lib/sendmail	Mail delivery program.
/usr/lib/unixtomh	Converts unix mail format to mh format.
maildir/.ack	The message sent in acknowledgement.
maildir/.format	The message sent when format errors are detected.
maildir/.redist	The redistribution list.
maildir/summary	The summary file.
maildir/Bf??????	Temporary copy of the input message.
maildir/Rp??????	Temporary file for the reply message.

1.1.46 burst

Purpose
Explodes digests into messages.

Syntax

```
         +-----------+   +------- cur --------------------------------+
burst ---¦           +---¦ +---------- all --------------------------+ +---
         +- +folder -+   +-¦   +---------- sequence -----------+   +-+
                           +---¦  one of    +------------------+ +---+
                               ¦ +-------+  ¦      one of      ¦ ¦ ¦
                             ¦ +-¦ num   +--¦¦+--------------+ +-+ ¦
                             ¦   ¦ first ¦  +-¦ :num   -prev +-+   ¦
                             ¦   ¦ prev  ¦    ¦ :+num  -cur  ¦     ¦
                             ¦   ¦ cur   ¦    ¦ :-num  -.    ¦     ¦
                             ¦   ¦ .     ¦    ¦ -num   -next ¦     ¦
                             ¦   ¦ next  ¦    ¦ -first -last ¦     ¦
                             ¦   ¦ last  ¦    +--------------+     ¦
                             ¦   +-------+                         ¦
                             +-------------------------------------+

    +--- -noinplace -+   +-- -noquiet --+   +-- -noverbose --+
  ---¦     one of     +---¦    one of    +---¦     one of     +---¦
     ¦ +------------+ ¦   ¦ +----------+ ¦   ¦ +------------+ ¦
     +-¦ -inplace   +-+   +-¦ -quiet   +-+   +-¦ -verbose   +-+
       ¦ -noinplace ¦       ¦ -noquiet ¦       ¦ -noverbose ¦
       +------------+       +----------+       +------------+

burst --- -help ---¦
```

¦ Do not put a blank between these items.

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **burst** command is used to explode digests, messages forwarded by the **forw** command, and blind carbon copies sent by the **forw** and **send** commands. The **burst** command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **burst** command cannot explode more than about 1,000 messages from a single message. The command, however, generally does not place a specific limit on the number of messages in a folder after bursting is complete.

The **burst** command uses encapsulation boundaries to determine where to separate the encapsulated messages. If an encapsulation boundary is located within a message, the **burst** command may split that message into two or more messages.

Flags

+*folder msgs*   Specifies the messages that you want to burst. You can use *msgs* to specify several messages, a range of messages, or a single message. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default message is the current message in the current folder. If the **-inplace** flag is also specified, the first message burst becomes the current message. Otherwise, the first message extracted from the first digest becomes the current message.

-help	Displays help information for the command.
-inplace	Replaces each digest with a table of contents for the digest, places the messages contained in each digest directly after the digest's table of contents, and renumbers all subsequent messages in the folder to make room for the messages in the exploded digest. Warning: The **burst** command does not place text that appears after the last encapsulated message in a separate message. When you specify the **-inplace** flag, **burst** loses this trailing text. In digests, this text is usually an End-of-Digest string. However, if the sender appended remarks after the last encapsulated message, **burst** loses those remarks.
-noinplace	Preserves each digest, does not produce a table of contents for each digest, and places the messages contained in each digest at the end of the folder. The **burst** command does not affect messages that are not part of digests. This flag is the default.
-noquiet	Reports information about messages that are not in digest format. This flag is the default.
-noverbose	Does not report the general actions that the **burst** command performs while exploding the digests. This flag is the default.
-quiet	Does not report information about messages that are not in digest format.
-verbose	Reports the general actions that the **burst** command performs while exploding the digests.

Profile Entries

Current-Folder:	Sets your default current folder.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your **user_mh_directory**.

Files

$HOME/.mh_profile The MH user profile.

Related Information
See other MH commands: "forw" in topic 1.1.174, "inc" in topic 1.1.206, "msh" in topic 1.1.281, "packf" in topic 1.1.310, "send" in topic 1.1.416

and "show" in topic 1.1.423.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.47 cal

Purpose
Displays a calendar.

Syntax

```
       +---------+
cal ---¦         +-- year --¦
       +- month -+
```

Note: This command does not have MBCS support.

Description
The **cal** command writes to standard output a Gregorian calendar for the specified year or month.

The **month** parameter names the month for which you want the calendar. It can be a number between 1 and 12 (for January through December, respectively). If you are using a system that supports a multibyte character set, the name of the month and the abbreviation of the day of the week are locale-specific.

The **year** parameter names the year for which you want the calendar. Since the **cal** command can display a calendar for any year from 1 to 9999, enter the full year rather than just the last two digits.

Examples

1. To display a calendar for February 1984 at your work station:

   ```
   cal 2 1984
   ```

2. To print a calendar for 1984:

   ```
   cal 1984 | print
   ```

3. To display a calendar for the year 84 A.D.:

   ```
   cal 84
   ```

Related Information

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.48 calendar

Purpose
Writes reminder messages to standard output.

Syntax

```
              +-----+
calendar ---¦       +---¦
              +- - -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **calendar** command reads a file named **calendar**, which you create in your current (usually home) directory. It writes to standard output any line in the file that contains today's or tomorrow's date.

The **calendar** command recognizes date formats such as **Dec. 7** or **12/7**. It also recognizes the special character * (asterisk). For example, ***/7** uses the asterisk to replace the month; the asterisk is used as a wildcard for month. The **calendar** command does not recognize formats such as **7/***, **7 December**, or **DEC. 7**.

For you to get reminder service, your **calendar** file should have read permission for others (see "chmod" in topic 1.1.67).

Note: The calendar file is read by the C pre-processor. Any C pre-processor commands can be used in the calendar file. Also, **/*** is the beginning of a comment for the pre-processor and must end with the ***/** characters.

All error messages are generated by the C pre-processor.

Flag

- Calls **calendar** for everyone having a file **calendar** in his home directory and sends reminders by **mail**

Example

To display information in the **calendar** file that pertains to the next two business days:

```
  calendar
```

A typical **calendar** file might look like this:

```
  Aug. 12 - Fly to Denver
  aug 23 - board meeting
  Martha out of town - 8/23, 8/24, 8/25
  8/24 - Mail car payment
  sat aug/25 - beach trip
  August 27 - Meet with Simmons
  August 28 - Meet with Wilson
```

If today is Friday, August 24, the **calendar** command displays:

```
  */25 - Prepare monthly report
  Martha out of town - 8/23, 8/24, 8/25
  8/24 - Mail car payment
  sat aug/25 - beach trip
  August 27 - Meet with Simmons
```

Files

$HOME/calendar	
/usr/lib/calprog	The program that determines dates.
/etc/passwd	Used to identify users.
/tmp/cal*	Temporary files.

Related Information

See the following commands: "chmod" in topic 1.1.67, "mail, Mail" in topic 1.1.253, and "cpp" in topic 1.1.94.

1.1.49 cat

Purpose
Concatenates or displays files.

Syntax

```
      +------------+   +------------+
cat ---¦   +----+   +---¦            +---¦
      +---¦ -b +---+   +--- file ---+
          ¦ -e ¦                ¦
        ¦ ¦ -n ¦ ¦        +--------+
        ¦ ¦ -q ¦ ¦
        ¦ ¦ -s ¦ ¦
        ¦ ¦ -t ¦ ¦
        ¦ ¦ -u ¦ ¦
        ¦ ¦ -v ¦ ¦
        ¦ +----+ ¦
        +--------+
```

Description
The **cat** command reads each **file** in sequence and writes it to standard output. If you do not specify **file** or specify **-** (minus) instead of **file**, the **cat** command reads from standard input.

Warning: Do not redirect output to one of the input files using the **>** redirection symbol. If you do this, you will lose the original data in the input file because the shell truncates the file before the **cat** command can read it (see "sh, Rsh" in topic 1.1.420).

Flags

-b	Displays output lines preceded by line numbers omitting blank lines.
-e	A "$" character is displayed at the end of each line.
-n	Displays output lines preceded by line numbers, numbered sequentially from 1.
-q	Quiet option. Does not display a message if **cat** cannot find an input file.
-s	Squeeze option. Removes multiple adjacent empty lines so that output is displayed single-spaced.
-t	Tab characters print as ^ I.
-u	Does not buffer output.
-v	Displays non-printing characters so that they are visible.

Examples

1. To display a file at the work station:

```
   cat  notes
```

   This displays the file **notes**. If the file is more than about 23 lines long, some of it will scroll off the screen. To display a file one page at a time, use the **pg** command. (See "pg" in topic 1.1.315 for details.)

2. To concatenate several files:

   ```
   cat  section1.1  section1.2  section1.3  >section1
   ```

   This command creates a file named **section1** that contains the file **section1.1** followed by files **section1.2** and **section1.3**.

3. To suppress error messages about files that do not exist:

   ```
   cat  -q  section2.1  section2.2  section2.3  >section2
   ```

   If **section2.1** does not exist, this command concatenates **section2.2** and **section2.3** but does not issue an error message about the non-existence of the file. If you do not use the **-q** flag, the command concatenates the files, but displays this error message:

   ```
   cat: cannot open section2.1
   ```

   You may want to use the **-q** flag to suppress this message when you use the **cat** command in shell procedures.

4. To append one file to the end of another:

   ```
   cat  section1.4  >>section1
   ```

   This command uses the redirection symbol >> to append a copy of **section1.4** to the end of **section1**. For more on redirections, see "Redirection of Input and Output" in topic 1.1.420.16.

5. To add text to the end of a file:

   ```
   cat  >>notes
   Get milk on the way home
   Ctrl-D
   ```

   Get milk on the way home is added to the end of the file **notes**. The **cat** command does not prompt; it waits for you to enter text. Press **Ctrl-D** to indicate you are finished.

6. To concatenate several files with text entered from the keyboard:

   ```
   cat  section3.1  -  section3.3  >section3
   ```

   This command concatenates **section3.1**, text from the keyboard, and **section3.3**.

7. To concatenate several files with output from another command:

   ```
   li  |  cat section4.1 -  >section4
   ```

   This command copies **section4.1** and then the output of the **li** command to the file **section4**.

Related Information

See the following commands: "cp, copy" in topic 1.1.91, "pr" in topic 1.1.322, and "sh, Rsh" in topic 1.1.420.

1.1.50 catman

Purpose
Creates the cat files for the manual.

Syntax

```
                  +--------+   +-------------+
/etc/catman ---¦ one of +---¦ +---------+ +--- sections ---¦
                  ¦ +----+ ¦   +-¦ -p      +-+
                  +-¦ -n +-+     ¦ -M path ¦
                    ¦ -w ¦      ¦+---------+¦
                    +----+      +-----------+
```

Note: This command does not have MBCS support.

Description
The **catman** command creates the preformatted versions of the on-line manual from the **nroff** input files. Each manual page is examined and those whose preformatted versions are missing or out of date are recreated. If any changes are made, **catman** recreates the **whatis** data base.

Note: AIX online manuals are delivered preformatted. The **catman** command is used only for manuals delivered in **nroff** format.

If there is one parameter not starting with a "-", it is taken to be a list of manual sections to look in. For example:

```
catman 123
```

will cause updating only to manual sections 1, 2, and 3.

Flags

-n Prevents creations of the **whatis** data base.

-p Prints what would be done instead of doing it.

-w Causes only the **whatis** data base to be created. No manual reformatting is done.

-M Updates manual pages located in the set of directories specified by **path** (/usr/man by default). **path** has the form of a colon-separated list of directory names, for example:

```
/usr/local/man:/usr/man
```

If the environment variable **MANPATH** is set, its value is used for the default path.

If the **nroff** source file contains only a line of the form:

```
.so manx/yyy.x
```

a symbolic link is made in the catx directory to the appropriate preformatted manual page. This feature allows easy distribution of the preformatted manual pages among a group of associated machines with **rdist**. The **nroff** sources need not be distributed to all machines, thus saving the associated disk space. As an example, consider a local network with 5 machines, called mach1 through mach5. Suppose mach3 has the manual page **nroff** sources. Every night, mach3 runs **catman** via **cron** and later runs **rdist** with a distfile that looks like:

```
  MANSLAVES = ( mach1 mach2 mach4 mach5 )

  MANUALS = (/usr/man/cat[1-8no] /usr/man/whatis)

  ${MANUALS} -> ${MANSLAVES}
      install -R;
      notify root;
```

Files

/usr/man	Default manual directory location.
/usr/man/man?/*.*	Raw (**nroff** input) manual sections.
/usr/man/cat?/*.*	Preformatted manual pages.
/usr/man/whatis	**whatis** data base.
/usr/lib/makewhatis	Command script to make **whatis** data base.

Related Information
The following commands: "man" in topic 1.1.258 and "cron" in topic 1.1.97.

1.1.51 cb

Purpose
Puts C language source code into a form that is easily read.

Syntax

```
      +------+   +-------------+   +------+   +------------+
cb ---¦      +---¦             +---¦      +---¦            +---¦
      +- -s -+   +- -l length -+   +- -j -+   +--- file ---+
                                                     ¦
                                                +--------+
```

Note: This command does not have MBCS support.

Description
The **cb** command reads C language programs from standard input or from specified **file**s and writes them to standard output in a form that shows, through indentations and spacing, the structure of the code. When called without flags, the **cb** command does not split or join lines. Punctuation in preprocessor statements can cause indentation errors.

Flags

-j Joins lines that are split.

-l length
Splits lines that are longer than **length** characters.

-s Formats the source code according to the style of Kernighan and Ritchie in *The C Programming Language.* (Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1978.).

Example

To create a version of **pgm.c** called **pgm.pretty.c** that is easy to read:

```
  cb  pgm.c  > pgm.pretty.c
```

Related Information

See "cc" in topic 1.1.52.

See the discussion of **cb** command in *AIX Operating System Programming Tools and Interfaces*.

1.1.52 cc

Purpose
Compiles C programs.

Syntax

```
     +-------------+ +----------------+ +--------------+
     ¦    one of   ¦ ¦  +----------+  ¦ ¦              ¦
cc --¦ +---------+ +-¦  ¦ -g   -pg ¦  +-¦              +--
     +-¦ -c   -S +-+ +--¦ -G   -w  +--+ +-- -Hoption --+
       ¦ -E   -X ¦      ¦ -O   -z  ¦ ¦                ¦
       ¦ -P      ¦    ¦ ¦ -p       ¦ ¦   +------------+
       +---------+    ¦ +----------+ ¦
                      +--------------+
   +-------------+ +----------+ +-----------------------+
 --¦  +--------+  +-¦          +-¦²                      +--
   +--¦ -oname +--+ +- -Dname -+ ¦ d+   o1+   o4+   ddir ¦
      ¦ flag ¦ ¦ ¦               ¦ g+   o2+   v+    ename ¦
    ¦ +--------+ ¦               ¦ 1+   o3+   w-    lname ¦
    +------------+               +-----------------------+
   +-------------------------+          +---------+
 --¦           ¦+-----------+ +---- file ---¦      4 +-¦
   +- -F file --¦          +-+          ¦ +- -lkey -+
               +- :stanza -+    +--------+
```

¦ Use any flag belonging to **as**, **cpp**, or **ld** (except **-l key**).
² These options are available on PS/2 only.
¦ No space between these items.
4 Put this flag last if used (see the **ld command**).

Description
The **cc** command runs the C compiler. It accepts files containing C source code, assembler source code, or object code and changes them into a form that the operating system can run. **cc** compiles and assembles source files and then links them with any specified object files, in the order listed on the command line. It puts the resulting executable program in a file named **a.out**.

Running **cc** on an AIX PS/2 machine will generate an executable program that will run on any AIX PS/2 machine. Similarly, running **cc** on an AIX/370 machine will generate an executable program that will run on any AIX/370 machine. By default, an AIX/370 program so compiled cannot make use of the extended (31-bit) addressing mode of XA370 machines. To build an XA370 executable program, use the **-Hxa** flag.

If both the C Language and the Extended C Language compiler are installed on a PS/2, the Extended C compiler is invoked by default. The C compiler is invoked only if the **COMPILER** environment variable is set to **VSC**. If only one compiler is installed on a PS/2, **/bin/cc** invokes that C compiler.

The **cc** command uses the **cc.cfg** configuration file, which specifies the standard runtime options, the link options, and the libraries to be used with each version of the compiler.

For Berkeley Compability: When porting an application program from a 4.3 BSD UNIX system to AIX, it can be useful to link with the Berkeley Compatibility library. This library provides routines which more closely resemble the behavior of a 4.3 BSD system. To provide this link, define the **BSD** environment variable in your shell before running **cc**. Defining this variable instructs **cc** to place the command line flags **-D_BSD** and **-lbsd** on your command line when using **cc**. For more information on BSD compatibility, see BSD4.3 Library in the *AIX Technical Reference*.

Subtopics

1.1.52.1 Input File Types

The **cc** command recognizes and accepts as input the following file types:

file.c	The name of a C-language source file should end with **.c**. After **cc** compiles this source file, it gives the resulting object file the same name, except that it ends in **.o** rather than **.c**. If you use one command both to compile and to load a single C program, the compiler normally deletes the **.o** file when it loads the program. If you use the **-c** flag or compile multiple files, the compiler does not delete the **.o** files.
file.i	The name of a file that contains preprocessed C source code ends in **.i**.
file.o	The name of an object file should end in **.o**. The **cc** command sends these files to the **ld** command.
file.s	The name of an assembly language source program should end with **.s**. After **cc** assembles this source file, it gives the resulting object file the same name, except that it ends in **.o** rather than **.s**.

Flags

The **cc** command recognizes several flags. In addition, flags intended to modify the action of the linkage editor (**ld**), the assembler (**as**), or the preprocessor (**cpp**) may also appear on the **cc** command line. **cc** sends any flags it does not recognize to these commands for processing. The following list includes the most commonly used **cpp** flags (**-D**, **-I**), and **ld** flags (**-l**, **-L**, **-o**). See "as" in topic 1.1.25, "cpp" in topic 1.1.94, and "ld" in topic 1.1.226 for a complete list of additional flags.

Notes:

1. If you use the **-l** flag, it must be the last entry on the command line, following any file parameters.

2. Do not use the **-H** options on the C Language compiler. All **-H** options are Extended C Language-specific.

1.1.52.2 Ordinary Operation

-c Does not send the completed object file to the **ld** command. With this flag, the output of **cc** is a **.o** file for each **.c** or **.s** file.

-Dname[=def]
Defines **name** as in a **#define** directive. The default **def** is 1.

-F file [:stanza]
Uses an alternative file and/or stanza for **cc** configuration. See the *AIX Operating System Technical Reference* for a discussion of the configuration file, **cc.cfg**.

-Hansi Causes the compiler to accept only programs conforming to the proposed ANSI Standard. When you use the **-Hansi** option, you must also use the **-Hnoccp** option.

-Hanno Used in conjunction with the **-S** option. Specifies that the generated **.s** file is to be annotated with lines from the source file. In some cases, the annotated lines from the source file are not located immediately adjacent to the corresponding assembly code generated in the **.s** file.

-Hasm Directs the compiler to produce a (pseudo-) assembly listing of the generated code on standard output, by initializing the **Asm** toggle to **On**. The assembly listing is annotated with lines from the main source file, but not with lines from any included files. These lines appear as comments immediately preceding the corresponding assembly instructions. If the **-S** option (described below) is also specified, the generated **.s** file is annotated with lines from the source file, and no listing is written on standard output; for example, it has the same effect as **-Hanno**.

-Hcpp Specifies that the outboard C macro preprocessor (**/lib/cpp**) is to be used. **-Hcpp** is the default **-H** option. The preprocessor symbol **__STDC__** is defined only if **-Hpcc** is not specified and **-Hnocpp** is specified. When the **-P** or **-E** flag is used, the **-Hcpp** option is assumed.

-Hnocpp Specifies the use of the inboard C macro preprocessor. The preprocessor symbol **__STDC__** is defined only if **-Hnocpp** is specified and **-Hpcc** is not specified. When the **-P** or **-E** flag is used, the **-Hcpp** option is assumed.

-Hfsingle Specifies that single-precision arithmetic is to be used in computations involving only float expressions. That is, floating-point operations are not to be performed in double precision, which is the default. Non-prototyped functions declared to return **float** may actually return **double**, depending on the setting of the toggle **Double_return**. Some programs run much faster with this option, but beware of loss of significance due to lower-precision intermediate computations.

-Hlines=n Causes a page eject to occur after every **n** lines written to standard output. The default of **60** is appropriate for most 6-lines-per-inch printers, which allow a maximum of 66 lines per page for 11-inch paper. The setting of **-Hlines** is intended to allow some blank space at page boundaries. For

8-lines-per-inch (88 lines per page) printers, **-Hlines** should be set to **80 or 82**. This option is used in conjunction with the **-Hlist** and **-Hasm** options. If **n** is **0**, no page ejects are emitted.

-Hlist Causes the compiler to generate a source listing on standard output. It works by initializing the **List** toggle to **On**.

-Hxa Generates load modules to run on XA-machines link-edit (ed) flag.

Note: When you are running on an XA machine, you should specify the **-Hxa** option. If this option is not used, the space available for allocation by the **sbrk** system call and **malloc** subroutine is limited to 5 Mb (which would be 500 Mb when **-Hxa** is specified) and 32-bit addresses are truncated to 24 bits.

-Hon=toggle

-Hoff=toggle
Turns a toggle **On or Off**. See the *AIX Operating System C Using Guide*.

-Hpcc Specifies that the compiler is to run in "PCC" mode. In this mode, the compiler relaxes enough of the ANSI extensions to more or less emulate the Portable C Compiler. This permits old C programs that would not ordinarily compile to compile with little (if any) modification.

-H+w Issues all warnings by turning off the **PCC_msgs** toggle.

-E Runs the named C source file through only the preprocessor and writes the result to standard output. The option works the same as **-P** except **-E** writes to standard output. Specifying **-P** means that **-Hcpp** is assumed and the outboard preprocessor is used even if **-Hnocpp** is specified.

-g Produces additional information for use with the **dbx** command (the symbolic debugger).

-G Indicates that global variables are volatile. The optimizer makes fewer transformations when you specify this flag. To make a particular variable volatile, add the "volatile" specification to its declaration. The **-G** option is supported by the C compiler only.

-Idir Looks first in **dir**, then looks in the directories on the standard list for **#include** files with names that do not begin with / (slash).

-lkey Searches the specified library file, where **key** selects the file **libkey.a.ld** searches for this file in the directory specified by an **-L** flag, then in **/lib** and **/usr/lib**. The **ld** command searches library files in the order in which you list them on the command line.

-Ldir Looks in **dir** for files specified by **-l** keys. If it does not find the file in **dir**, **ld** searches the standard directories.

-oname	Assigns **name** rather than **a.out** to the output file.
-O	Turns on the code optimizers.
-p	Prepares the program so that the **prof** command can generate an execution profile. The compiler produces code that counts the number of times each routine is called. If programs are sent to **ld**, the compiler replaces the startup routine with one that calls the **monitor** subroutine at the start (see *AIX Operating System Technical Reference* for a discussion of this subroutine), and writes a **mon.out** file when the program ends normally.
-P	Sends the specified C source file to the macro preprocessor and stores the output in a **.i** file. Specifying **-P** means that **-Hcpp** is assumed and the outboard preprocessor is used even if **-Hnocpp** is specified.
-pg	Causes the compiler to produce counting code in the manner of **-p**, but invokes a run-time recording mechanism that keeps more extensive statistics and produces a **gmon.out** file at normal termination. Also, a profiling library is searched, in lieu of the standard C library. An execution profile can be generated by use of **gprof** (1, C) C Language compiler only.
-S	Compiles the specified C programs, storing assembly language output in a **.s** file.
-w	Prevents printing of warning messages.
-X	Generates an assembly listing to standard output. (This option is available with the C Language compiler only.)
-z	This option is available only with the C Language compiler. Uses the **libm.a** version, or a version specified by the user, of the following transcendental functions:

acos	asin	atan	atan2	cos	exp
log	log10	sin	sqrt	tan	

If this flag is not used, the compiler generates inline instructions for the 80387 math co-processor. For more information on **libm.a**, see **math.h** in *AIX Operating System Technical Reference*.

1.1.52.3 PS/2 Specific Options

The AIX PS/2 C Language Compiler-specific options are described below. They indicate which features are enabled or disabled when the compiler is invoked. The options are described in alphabetical order.

b+ **Floating Point Computation**

Instructs the compiler to promote all floating-point values to double precision before all floating-point computations.

d+ **Disassembler Information**

Produces additional information for use with the **dis** command (the disassembler).

Note: With this option, you can also use the **dbx** command (symbolic debugger). (See the **g+** option below.) However, the **.d** file does not contain symbolic information; therefore, you can only do machine level debugging.

efilename Error File

Instructs the compiler to place its error output in the file specified by **filename**. If the **filename** option is not specified, error messages are written to the standard error device.

g+ **Debugger Information**

Produces additional information for use with the **dbx** command (the symbolic debugger).

Notes:

1. With this option, you can also use the **dis** command disassembler), see the **d+** option above. However, allocation of variables into registers is turned off.

2. If both the **d+** and the **g+** command-line options are set, regardless of their order on the command line, the **g+** option has the higher priority.

3. The optimization process is disabled whenever the **g+** option is specified on the command line.

lfilename Listing File

Instructs the compiler to place its listing output in the file specified by **filename**. If the **lfilename** option is not specified, a listing file is not generated.

l+ **List to Standard Output Device**

Instructs the compiler to generate a listing to the standard output device.

o1+ **Optimization Level 1**

Instructs the compiler to use optimization level 1.

o2+ **Optimization Level 2**

Instructs the compiler to use optimization level 2.

o3+ **Optimization Level 3**

Instructs the compiler to use optimization level 3.

o4+ **Optimization Level 4**

Instructs the compiler to use optimization level 4.

v+ **Compiler Progress Information**

Instructs the compiler to generate information on the progress of the compile.

w- **No Warning Messages**

Instructs the compiler not to generate warning messages.

1.1.52.4 Debugging

-v	Displays the trace as with **-#** and invokes the programs.
-#	Displays a trace of the actions to be taken (for example, invoking the preprocessor), without actually invoking any programs.

1.1.52.5 Extended Functions

C Language compiler only

-Bprefix Constructs path names for substitute preprocessor, compiler, optimizer, assembler, or linkage editor programs. **prefix** defines part of a path name to the new programs. To form the complete path name for each new program, **cc** adds **prefix** to the standard program names. For example, if you enter the command:

```
  cc  testfile.c  -B/usr/bob/new
```

cc calls the following compiler programs:

1. /usr/bob/newcpp
2. /usr/bob/newvsc
3. /usr/bob/newvspass2
4. /usr/bob/newvspass3
5. /usr/bob/newas
6. /usr/bob/newld

Similarly, if you enter the command:

```
  cc  testfile.c  -B/usr/bob/new/
```

cc calls the following compiler programs:

1. /usr/bob/new/cpp
2. /usr/bob/new/vsc
3. /usr/bob/new/vspass2
4. /usr/bob/new/vspass3
5. /usr/bob/new/as
6. /usr/bob/new/ld

The default **prefix** is **/lib/o**.

-t[pcgfal]
Applies the **-B** flag instructions for constructing file names to only the designated preprocessor (**p**), compiler first (**c**), compiler second (**g**), compiler third (**f**), assembler (**a**), or linkage editor (**l**) passes. You can select any combination of **pcgfal**.

The **-t** flag with no additional **p**, **c**, **g**, **f**, **a**, or **l** designates by default the preprocessor and compiler programs. If you do not specify the **-B** flag when you specify the **-t** flag, the default file name **prefix** is **/lib/n**.

Note: You can specify this **prefix** with the **-B** flag. However, depending on what combination of the **-B** and the **-t** flags you specify, **prefix** can have two possible default values. If you specify **-B** but no accompanying **prefix**, the default **prefix** is **/lib/o**. If you specify the **-t** flag without also specifying the **-B** flag, the default **prefix** is **/lib/n**.

-Wc,flag1[,flag2...]
Gives the listed flags to the compiler program **c**; **c** can be any one of the values [**pcgfal**] discussed with the **-t** flag. For example, since both **ld** and **as** recognize a **-o** flag, use **-W** to

specify the program to which the flag is to be sent. That is, **-Wl,-o** sends it to **ld**. **-Wa,-o** sends it to **as**.

Examples

1. To compile and link a C program, creating an executable **a.out** file:

   ```
   cc pgm.c
   ```

2. To compile a program, producing an object file to be linked later:

   ```
   cc -c pgm.c
   ```

   This compiles **pgm.c** and produces an object file named **pgm.o**.

3. To view the output of the macro preprocessor:

   ```
   cc -P -c pgm.c
   ```

   This creates a file named **pgm.i** that contains the preprocessed program text including comments. To view this file, use an editor or see "pg" in topic 1.1.315. **cc** passes the **-P** and **-c** flags to the preprocessor. See "cpp" in topic 1.1.94 for more details about them.

4. To predefine macro identifiers:

   ```
   cc -DBUFFERSIZE=512 -DDEBUG pgm.c
   ```

   This assigns **BUFFERSIZE** the value **512** and **DEBUG** the value **1** before preprocessing. **cc** passes the **-D** flag to the preprocessor.

5. To use **#include** files located in nonstandard directories:

   ```
   cc -I/u/bob/include pgm.c
   ```

   This looks in the directory that contains **pgm.c** for the **#include** files with names enclosed in double quotes (**" "**), then in **/u/bob/include**, and then in the standard directories. It looks in **/u/bob/include** for **#include** file names enclosed in angle brackets (**< >**), then in the standard directories. **cc** passes the **-I** flag to the preprocessor.

6. To optimize the object code and produce assembler source code:

   ```
   cc -S -O pgm.c
   ```

   This uses the optimizing compiler (**-O** is minus, capital oh), and produces assembler source code in a file named **pgm.s** (**-S**).

Files

file.c	C source file.
file.i	Preprocessed C source file.
file.o	Object file.
file.s	Assembler file.
a.out	Linked output.
/etc/cc.cfg	**cc** configuration file.
/tmp/ctm*	Temporary.
/lib/cpp	C preprocessor.
/bin/as	Assembler.
/lib/ld	Linkage editor.

/lib/crt0.o	Runtime startoff.
/lib/mcrt0.o	Runtime startoff for profiling.
/lib/libc.a	Standard library.
/lib/libm.a	Standard math library.
/lib/librts.a	Runtime services.
/usr/include	Standard directory for #include files.

Related Information

See the following commands: "as" in topic 1.1.25, "ld" in topic 1.1.226, "vs" in topic 1.1.526, "cpp" in topic 1.1.94 and "dbx, mdbx" in topic 1.1.111.

See the C compiler in *AIX C Language Reference* and *AIX C Language User's Guide.*

See the **monitor** subroutine and the **a.out** and **cc.cfg** files in *AIX Operating System Technical Reference*.

1.1.53 cd

Purpose
Changes the currrent directory.

Syntax

```
      +- $HOME -----+
cd ---¦             +---¦
      +- directory -+
```

Description
The **cd** command moves you from your present directory to another. You must have execute (search) permission in the specified **directory**.

The **cd** command is actually a built-in subcommand of the **sh** and **csh** commands. This man page only describes the common behavior of these two subcommands. Both **sh** and **csh** provide shortcut ways to specify **directory**. The **sh** command uses the shell variable **CDPATH**, while **csh** uses the shell variable **cdpath** and other shell variables. See the **sh** and **csh** commands for more details.

Warning: When changing to a directory using a pathname which includes one or more symbolic links, the real name of the target directory may be different than the name issued in the **cd** command. Use the **pwd** command after the **cd** command to determine the actual name of the directory reached. Thereafter pathnames beginning with ".." will refer to the parent of the actual directory.

Examples

1. To change to your home directory:

   ```
   cd
   ```

2. To change to an arbitrary directory:

   ```
   cd  /usr/include
   ```

   This changes the current directory to **/usr/include**. Now file path names that do not begin with **/** or **../** specify files located in **/usr/include**.

3. To go down one level of the directory tree:

   ```
   cd  sys
   ```

   If the current directory is **/usr/include** and if it contains a subdirectory named **sys**, **/usr/include/sys** becomes the current directory.

4. To go up one level of the directory tree:

   ```
   cd  ..
   ```

   The special file name **..** (dot-dot) always refers to the directory immediately above the current directory. Note that **..** (dot-dot) is a hard link to the parent directory. If the current directory was entered via a symbolic link, the command **cd ..** may put you in a directory you do not expect. In the following command sequence, **pwd** will show **/aixps** (aixps is the default name of the local filesystem) since **/tmp** is a symbolic link to **<LOCAL>/tmp**.

```
cd /tmp
cd ..
pwd
```

Related Information

The following commands: "csh" in topic 1.1.100, "pwd" in topic 1.1.344, and "sh, Rsh" in topic 1.1.420.

The **chdir** system call in *AIX Operating System Technical Reference*.

1.1.54 cdc

Purpose
Changes the comments in a Source Code Control System (SCCS) delta.

Syntax

```
                 +------------+   +-------------+
cdc --- -rSID ---¦            +---¦             +--- file ---¦
                 +- -mmrlist -+   +- -ycomment -+          ¦
                                                  +--------+

                 +---- -m ----+   +---- -y -----+
cdc --- -rSID ---¦            +---¦             +--- - ---¦
                 +- -mmrlist -+   +- -ycomment -+
```

Description
The **cdc** command changes the Modification Requests (MRs) and comments for the **SID** specified by the **-r** flag for each named Source Code Control System (SCCS) **file**. If you specify a directory name, the **cdc** command performs the requested actions on all SCCS files in that directory (that is, all files with names that have the **s.** prefix). If you specify a **-** (minus) in place of **file**, the **cdc** command reads standard input and interprets each line as the name of an SCCS file. For more information on SCCS comments and Modification Requests, see *AIX Operating System Programming Tools and Interfaces*.

You can change the comments and MRs for an SID only if you made the SID or if you own the file and the directory. For more information on the permissions needed to change SCCS files, see "SCCS Files" in topic 1.1.186.1.

Flags

-m[mrlist] Supplies a list of MR numbers for the **cdc** command to add or delete in the SID specified by the **-r** flag. You can only use this flag if the **file** has the **v** header flag set (see Table 1-2 in topic 1.1.16.3). A null MR list has no effect.

In the **mrlist**, MRs are separated by blanks, tab characters, or both. To delete an MR, precede the MR number with an ! (exclamation point). If the MR you want to delete is currently in the list of MRs, it is changed into a comment line. The **cdc** command places a list of all deleted MRs in the comment section of the delta and precedes each MR with a comment line indicating which MRs was deleted.

If you do not specify the **-m** flag and the **v** header flag is set, MRs are read from standard input. If standard input is a terminal, the **cdc** command prompts you for the MRs. The first new-line character not preceded by a backslash ends the list on the command line. The **cdc** command continues to take input until it reads an end-of-file character (**Ctrl-D**) or a blank line. MRs are always read before comments (see the **-y** flag).

If the **v** flag has a value, the **cdc** command interprets the value as the name of a program which validates the MR numbers. If the MR number validation program returns a nonzero exit value, the command stops and does not change the MRs.

-rSID Specifies the SCCS identification number of the delta for which the **cdc** command can change the comments or MRs.

-y[comment] Specifies text to replace any **comment** already existing for the delta specified by the **-r** flag. The **cdc** command keeps the existing comments and precedes them with a comment line stating that they were changed. A null **comment** has no effect.

If you do not specify the **-y** flag, the **cdc** command reads comments from standard input until it reads an end-of-file character. If the standard input is a work station, the **cdc** command prompts for the comments and also allows a blank line to end input. If the last character of a line is a \ (backslash), the **cdc** command ignores it and continues to read standard input.

Note: If the **cdc** command reads standard input for file names (that is, when you specify a file name of **-**), you must use the **-y** and **-m** flags.

Related Information

See the following commands: "admin" in topic 1.1.16, "delta" in topic 1.1.117, "get" in topic 1.1.186, "sccshelp" in topic 1.1.411, and "prs" in topic 1.1.336.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.55 cdcat

Purpose
Concatenates CD-ROM files.

Syntax

```
cdcat --- cdfile ---¦
                 ¦
       +----------+
```

Note: This command is for the PS/2 only.

Description

The **cdcat** command is a CD-ROM command that reads each CD-ROM file in sequence and writes it to standard output. This command is most useful as the first element in a pipeline, so that data from a CD-ROM can be accessed without first using the **cdcp** command to copy the file(s) to disk.

Letters in file names on a CD-ROM are restricted to uppercase. However, **cdcat** performs an internal case translation so that mixed-case arguments match CD-ROM file names.

CD-ROM file names are written as an absolute path beginning with an alias for a device name, normally **/cd0**. If a file name is written in relative form (that is, without a leading slash), the **/cd0** prefix is assumed.

Files
/dev/rcd0
/dev/rcdrom0

Related Information
See the following commands: "cdcp" in topic 1.1.56 and "cdls" in topic 1.1.59.

1.1.56 cdcp

Purpose
Copies files from CD-ROM.

Syntax

```
cdcp --- cdfile --- diskfile ---¦

cdcp --- cdfile --- diskdir ---¦
                 ¦
      +----------+
```

Note: This command is for the PS/2 only.

Description

The **cdcp** command is a CD-ROM command. The **cdcp** command copies one or more files from a CD-ROM to a standard file system. In the first form, the single file **cdfile** is copied to **diskfile**. If **diskfile** already exists, it must be writable (and will be overwritten). In the second form, one or more CD-ROM files are copied to the directory **diskdir**. In this case, the names of the created files are the base names of the corresponding CD-ROM files.

Letters in file names on a CD-ROM are restricted to uppercase. However, **cdcp** performs an internal case translation so that mixed-case arguments match CD-ROM file names. This translation applies only to files on the CD-ROM; resulting file names are as typed. For example:

```
  cdcp /cd00/NextOne.txt homedir
```

copies the CD-ROM file **NEXTONE.TXT** into the file system as **homedir/NextOne.txt**.

CD-ROM file names are written as an absolute path beginning with an alias for a device name, normally **/cd0**. If a file name is written in relative form (that is, without a leading slash), the **/cd0** prefix is assumed.

Note: Once the alias name is defined by **cdmount**, **cdcp** should be able to use relative path name and have the system assume the prefix (that is, local alias). However, if the local alias is something other than **/cd0**, then the full path name must be used.

Files
/dev/rcd0
/dev/rcdrom0

Related Information
See the following commands: "cdls" in topic 1.1.59 and "cdcat" in topic 1.1.55.

1.1.57 cddaemon

Purpose
Supports the CD-ROM device facility of AIX PS/2.

Syntax

/etc/cddaemon --- **&** ---¦

Note: This command is for the PS/2 only.

Description
The **cddaemon** command enables the CD-ROM facility of AIX PS/2. If you have one or more CD-ROM devices on your system, **cddaemon** must be running in order for users on your system, or for users on other hosts connected to your system via TCP/IP to access the files on your CD-ROM devices.

A recommended way to start **cddaemon**, so that it is running whenever your system is in multi-user mode, is to put the following command line:

```
  /etc/cddaemon &
```

into the file **/local/local.init.dir/Singl2multi**.

Once **cddaemon** is running, you can access the files on one of your CD-ROM devices by using the commands **cdls**, **cdcat**, or **cdcp**:

cdls	lists file names of files on CD-ROM devices.
cdcat	copies one or more files from CD-Rom devices to the standard out file stream.
cdcp	copies one or more files from CD-ROM devices to an AIX file system.

Warning: No access protection is provided for CD-ROM devices. All files are publicly readable and all directories are publicly searchable. If you have TCP/IP enabled on your system, then as soon as you provide CD-ROM access by running **cddaemon**, you are providing this access to any remote host who can connect to your system via TCP/IP.

Files on a CD-ROM device can be named using a CD-ROM path name. The general form of a CD-ROM path name is:

```
  [hostd:]path[;version]
```

or

```
  /localname/path[;version]
```

where

host	is the optional TCP/IP hostname of the system where the CD-ROM device is attached. If this is omitted, then the local system is assumed.
localname	is a name you established for a local or remote CD-ROM device by using the **cdmount** command.

path	is a relative pathname to a file on either the default CD-ROM device (drive 0) of the specified host (first case) or to the named device on the implied host (second case). **Path** is made up of one or more filenames, separated by **/** characters. The filenames are not case sensitive: while all files on the CD-ROM are named with upper case letters, you may specify the file using upper case, lower case or mixed case letters.
version	is the optional version number of the file. If this is omitted, the highest version of the named file is selected.

The **cdmount** command allows you to establish a localname for a specific CD-ROM device on a specific host. It does so by maintaining the mount table in the file **/local/cdmtab**, which remains intact even after a system reboot of either your system or a remote host which is providing your CD-ROM service.

The **cdmount** command can be used even for local CD-ROM devices, especially if you have more than one CD-ROM device on your system.

If you have the Transparent Computing facility on your system, you should note that the local names established with **cdmount** may only be used by users and programs running on your local cluster site. This is because each site in the cluster maintains its own **/local/cdmtab** file. To avoid confusion, it is recommended that you establish the same local names on each site in the cluster by issuing the same cdmount commands on all sites.

The following commands are provided with the cdrom facility:

cdlook	reports the volume label for each CD-ROM device on a selected host.
cdmount	removes a localname established with cdmount.
cdshutdown	signals the local cddaemon process and removes **/local/cdmtab**.

Related Information
See the following commands: "cdcat" in topic 1.1.55, "cdcp" in topic 1.1.56, "cdlook" in topic 1.1.58, "cdls" in topic 1.1.59, "cdmount, cdumount" in topic 1.1.60, and "cdshutdown" in topic 1.1.61.

1.1.58 cdlook

Purpose
Reports which CD-ROM drives exist and what they contain.

Syntax

```
               +--------+
/etc/cdlook --¦        +--¦
               +- host -+
```

Note: This command is for the PS/2 only.

Description

The **cdlook** CD-ROM command reports the physical drive numbers present on the **host** machine (the default is the local machine). If any drives have CD-ROMs present, the command reports their volume ID strings. The output of **cdlook** is useful in a subsequent **cdmount** request.

Related Information
See the following command: "cdmount, cdumount" in topic 1.1.60.

1.1.59 cdls

Purpose
Lists contents of CD-ROM directories.

Syntax

```
        +--------+
cdls ---¦ +----+ +--- cddirectory ---¦
        +-¦ -a +-+                   ¦
          ¦ -l ¦¦  +---------------+
         ¦+----+¦
         +------+
```

Note: This command is for the PS/2 only.

Description

The **cdls** command is a CD-ROM command that is a restricted variant of the **ls** command. It can be used to display the contents of one or more CD-ROM directories. In the simplest form of the command, the contents of a directory is shown, one entry per line. If more than one directory is given on the command line, the contents of each directory is shown preceded by a blank line and a line containing the name of the directory followed by a colon.

Letters in file names on a CD-ROM are restricted to uppercase. Since this restriction conflicts with common AIX practice, file names displayed by **cdls** are shown in lowercase.

CD-ROM file names are written as an absolute path beginning with an alias for a device name, normally **/cd0**. If a file name is written in relative form (that is, no leading slash), the **/cd0** prefix is assumed.

Flags

-a	Lists all entries in the directory, including entries that begin a dot.
-l	Similar to the -l option of the **ls** command, but there are some differences in the directory structure and content of a CD-ROM. File permissions are always shown as **r--r--r--** and directories as **r-xr-xr-x**. The link count for a file is shown as 1; directories show an unknown count. User and group IDs are printed as *******. The time shown for a file is the field described as "recording time" on the CD-ROM itself.

Files
/dev/cd0
/dev/rcdrom0

Related Information
See the following commands: "cdcp" in topic 1.1.56 and "cdcat" in topic 1.1.55.

1.1.60 cdmount, cdumount

Purpose
Mounts and unmounts CD-ROM devices.

Syntax

```
                         +--------+          +---------+
cdmount --- localalias --¦        +-- drive --¦         +--¦
                         +- host -+          +- volid -+

cdmount --¦

cdumount --- localalias --¦
```

Note: This command is for the PS/2 only.

Description

cdmount and **cdumount** are CD-ROM commands. In the first form, the **cdmount** command informs a CD-ROM server that a particular physical drive will be used, verifies that a particular CD-ROM volume is present in the drive, and establishes a local alias for use in naming files on the drive. The **cdmount** command tells the server on the machine specified by the **host** argument that the physical device numbered **drive** should be considered in use. If no **host** is given, the request applies to the local machine. If the **volid** argument is present, the server is expected to verify that the indicated drive contains a volume (a CD-ROM) with that identification. Since the volume ID on a CD-ROM is restricted to uppercase, **cdmount** translates the supplied name into uppercase font. For example:

```
cdmount myrom 2 USA_map_data
```

establishes **myrom** as an alias for the CD-ROM in physical drive 2 of the local machine, and verifies it contains a volume labeled **USA_MAP_DATA**. A subsequent request to access a file might use, for example, **/myrom/co/mtns.dat** to access the file **CO/MTNS.DAT** on that CD-ROM.

The **cdmount** command with no arguments lists all aliases known on the local machine.

The **cdumount** command removes the mapping established by a previous **cdmount** and informs the server that the volume is no longer needed.

Note: **cddaemon** must be running before **cdmount** can be executed.

Files

/local/cdmtab	Local alias/mount table.

Related Information
See the following command: "cdlook" in topic 1.1.58.

1.1.61 cdshutdown

Purpose
Terminates CD-ROM server.

Syntax

```
/etc/shutdown --¦
```

Note: This command is for the PS/2 only.

Description

The **cdshutdown** command causes the local CD-ROM server to terminate, release all shared memory, and delete files. It allows a clean restart of the server daemon if necessary. The **cdshutdown** command should be run as part of system shutdown.

Files

/local/cdmtab	Local alias/mount table.
/local/cdserver	Server socket psuedo-file.

Related Information
See the following command: "cdmount, cdumount" in topic 1.1.60.

1.1.62 cflow

Purpose
Generates a C flow graph of external references.

Syntax

```
         +------------+
cflow ---¦ +--------+ +--- file ---¦
         +-¦ -d num +-+          ¦
           ¦ -r     ¦¦  +--------+
          ¦¦ -i_    ¦¦
          ¦¦ -ix    ¦¦
          ¦+--------+¦
          +----------+
```

Note: This command is for the PS/2 only.

Note: This command does not have MBCS support.

Description
The **cflow** command analyzes C, **yacc**, **lex**, assembler, and object **file**s and writes a chart of their external references to standard output.

The **cflow** command sends files with **.y**, **.l**, and **.c** suffixes to the **yacc**, **lex**, and **cpp** commands for the appropriate processing. This step is bypassed for **.i** files. The command then runs the output of this processing through the first pass of the **lint** command. It assembles files which end in **.s**, extracting information from the symbol table (as it does with **.o** files). From this output, the **cflow** command produces a graph of external references, which it writes to standard output.

Each line of output begins with a line number followed by sufficient tabs to indicate the level of nesting. Then comes the name of the global, a colon, and its definition. This name is normally a function not defined as external and not beginning with an underline character (see the **-i_** inclusion flag on p. 1.1.62). For information extracted from C source files, the definition consists of an abstract type declaration (for example, **char***), the name of the source file (surrounded by angle brackets), and the line number on which the definition was found. Definitions extracted from object files contain the file name and location counter under which the symbol appeared. The **cflow** command deletes leading underline characters in C-style external names.

Once the **cflow** command displays the definition of a name, later references to it contain only the **cflow** line number where the definition may be found. For undefined references, the **cflow** command displays only <>.

If the nesting level becomes too deep to display in available space, pipe the output from the **cflow** command to the **pr** command, using the **-e** flag to compress the tab expansion to something less than every eight spaces.

Note: Files produced by the **lex** and **yacc** command cause the reordering of line number declarations which can confuse the **cflow** command. To get proper results, provide the **cflow** command with **yacc** or **lex** command input.

Flags
The **cflow** command recognizes the following flags:

-dnum Sets to decimal integer **num** the depth at which the flow graph is cut off. By default, **num** is a very large number. Do not set the cutoff depth to a negative integer.

-ix Includes external and static data symbols. The default includes only functions.

-i_ Includes names that begin with an underline character. The default excludes these functions (and corresponding data if the **-ix** flag is used).

-r Produces an inverted listing, which shows the callers of each function, sorted by called function.

In addition to these flags, the **cflow** command recognizes the **-I**, **-D**, and **-U** flags of the **cpp** command.

Related Information

See the following commands: "as" in topic 1.1.25, "cc" in topic 1.1.52, "lex" in topic 1.1.229, "lint" in topic 1.1.233, "nm" in topic 1.1.298, "pr" in topic 1.1.322, and "yacc" in topic 1.1.547.

See the discussion of the **cflow** programming tool in *AIX Operating System Programming Tools and Interfaces*.

1.1.63 checknr

Purpose
Checks **nroff/troff** files for errors.

Syntax

```
           +-----------------------------+
checknr ---¦ +-------------------------+ +---¦
           +-¦ -s                      +-+
             ¦ -                       ¦¦
            ¦¦ -a.x1.y1.x2.y2 ... xn.yn ¦¦
            ¦¦ -c.x1.x2.x3 ... xn       ¦¦
            ¦+-------------------------+¦
            +---------------------------+
```

Note: This command does not have MBCS support.

Description
The **checknr command** checks a list of **nroff** or **troff** input files for certain kinds of errors involving mismatched opening and closing delimiters and unknown commands. If no files are specified, **checknr** checks the standard input. Delimiters checked are:

(1) Font changes using \fx ... \fP.

(2) Size changes using \sx ... \s0.

(3) Macros that come in open/close forms (for example, the .TS and .TE macros) which must always come in pairs.

The **checknr command** can handle both the **ms** and **me** macro packages.

Flags

-a	Adds pairs of macros to the list. This option must be followed by groups of six characters, each group defining a pair of macros. The six characters are a period, the first macro name, another period, and the second macro name. For example, to define a pair .BS and .ES, use -**a**.BS.ES.
-c	Defines otherwise undefined commands.
-f	Causes **checknr** to ignore \f font changes.
-s	Causes **checknr** to ignore \s size changes.

The **checknr** command is intended to be used on documents that are prepared with **checknr** in mind, much the same as the **lint** command. It expects a certain document writing style for \f and \s commands, in that each \fx must be terminated with \fP and each \sx must be terminated with \s0. While it will work to go directly into the next font or explicitly specify the original font or point size, and many existing documents actually do this, such a practice will produce warning messages from **checknr**.

Related Information
See the following command: "nroff, troff" in topic 1.1.301.

1.1.64 chfstore

Purpose
Changes file storage attribute for files in system replicated filesystems.

Syntax

```
            +------+   +- class --+
chfstore ---¦      +---¦          +--- file ---¦
            +- -o -+   +- number -+          ¦
                                    +--------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **chfstore** command is used to change the file storage (fstore) attribute of files in system replicated file systems. The most common example of a system replicated file system is the replicated root file system. The **fstore** attribute controls which cluster sites will store a copy of the file.

The **fstore** attribute can be specified as as an octal **number** or as a class of machines. The choices for **class** are limited to those classes defined in **/etc/fstore**. This file defines the following classes.

i386 - Causes the file to be stored on all AIX PS/2 cluster sites.

i370 - Causes the file to be stored on all AIX 370 cluster sites.

none - Causes the file to be stored on only the primary and backbone cluster sites.

all - Causes the file to be stored on all cluster sites.

Flags

-o	Use this flag to maintain the original **fstore** value. The storage file attribute is then both the old and new values. For example, if the old value had been **i386**, and you used the **-o** flag to add **i370**, that file is stored in both machine types.

Diagnostics
Errors occur if a file does not exist, a file is not in a system replicated filesystem, or the caller is not the owner or superuser.

Files

/etc/fstore	Defines the octal fstore values associated with various classes of machines.

Related Information
See the following commands: "store" in topic 1.1.443 and "where" in topic 1.1.534.

See the **fs** and **fstore** topics in *AIX Operating System Technical Reference.*

1.1.65 chgrp

Purpose
Changes the group ownership of a file or directory.

Syntax

```
          +--------+   +- groupname -+   +-----------+
chgrp ---¦ +----+ +---¦             +---¦ file      +---¦
          +-¦ -f +-+   +-- groupid --+   ¦ directory ¦ ¦
            ¦ -r ¦¦                     ¦ +-----------+ ¦
           ¦¦ -R ¦¦                     +---------------+
           ¦+----+¦
           +------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **chgrp** command changes the group associated with the specified **file** or **directory** to **groupname** or **groupid**. Unless you have superuser authority, you can change the group ownership only on files that you own and only to groups of which you are a member.

Flags

-f	Suppresses all error reporting.
-r	Causes the untranslated group ID to be used.
-R	Causes **chgrp** to descend recursively through its directory arguments, setting the specified group ID. When symbolic links are encountered, their group is changed, but they are not traversed.

Examples

To change the group ownership of the file or directory named **proposals** to **staff**:

```
  chgrp staff proposals
```

The group access permissions for **proposals** now apply to the **staff** group.

Files

/etc/group	File that identifies all known groups.

Related Information

See the following command: "groups" in topic 1.1.194.

See the **chown** and **fchown** system calls and the **group** file in *AIX Operating System Technical Reference*.

1.1.66 chkfstore

Purpose
Checks fstore values.

Syntax

```
             +------------+
chkfstore ---¦   +----+   +--- fstore_value --- device ---¦
             +---¦ -i +---+
                 ¦ -  ¦ ¦
               ¦ +----+ ¦
               +--------+
```

Description
The **chkfstore** command scans the file system on the specified device and displays all inode numbers of those inodes that match the specified octal fstore value, **fstore_value**. The device name should identify a device containing a primary or backbone copy of a replicated filesystem.

A common use of **chkfstore** is to check for files with an fstore value of 0 in the replicated root filesystem. 0 is the fstore value assigned to files when they are created and is often an indication of a file incorrectly installed. Files in the root file system where a 0 fstore value would be correct (that is, those files which should be stored only on the primary and backbone sites) should be installed with an fstore value of none (020000000000).

Run **chkfstore** on the same type of machine as the the machine whose file system is being checked.

This program is intended for the administrator.

Flags
Normally, **chkfstore** will require the **fstore_value** equal to an inode's fstore value for the two to match. If the **-i** option is set **chkfstore** considers the fstore values to match if all the one bits in **fstore_value** are also ones in the inode's fstore.

-l results in the inode fstore value to be displayed as well as the inode number.

Related Information
See the following commands: "chfstore" in topic 1.1.64, "store" in topic 1.1.443 and "where" in topic 1.1.534.

See **fs** and **fstore** in the *AIX Operating System Technical Reference.*

1.1.67 chmod

Purpose
Changes permission codes.

Syntax

```
SYMBOLIC                                              +-------+
                                              one of  ¦ r t u ¦
                                              +---+  ¦¦ w s g ¦
         +--------+   +------- a -------+   +-¦ + +---¦ x X o +-----+   +-----
chmod ---¦ +----+ +---¦ +----- a -----+ +---¦ ¦ - ¦   +-------+ ¦   +---¦ file
         +-¦ -f +-+   +-¦ +---------+¦+-+   ¦ +---+ +-----------+   ¦¦¦ ¦ dire
           ¦ -R ¦¦      +-¦ u  g  o +-+   ¦ ¦         +-----------+ ¦¦¦¦+-----
          ¦+----+¦     ¦  ¦ ug  uo  ¦     ¦ +--- = ---¦ +-------+ +-+¦¦+-----
          +------+     ¦  ¦ og  ugo ¦     ¦           +-¦ r t u +-+  ¦¦
                       ¦  +---------+     ¦             ¦ w s g ¦¦   ¦¦
                       ¦                  ¦            ¦¦ x X o ¦¦   ¦¦
                       ¦                  ¦            ¦+-------+¦   ¦¦
                       ¦                  ¦            +---------+   ¦¦
                       ¦                  +--------------------------+¦
                       ¦                        2                     ¦
                       +-------------------- , -----------------------+
```

```
ABSOLUTE
         +--------+                +-----------+
chmod ---¦ +----+ +--- permcode ---¦ file      +---¦
         +-¦ -f +-+                ¦ directory ¦ ¦
           ¦ -R ¦¦               ¦ +-----------+ ¦
          ¦+----+¦               +---------------+
          +------+
```

¦ Do not put a blank between these items.
[2] Do not put a blank on either side of the comma.

Description
The **chmod** command modifies the read, write, execute (**file**), or search (**directory**) permission codes of specified files or directories. You can use either symbolic or absolute mode to specify the desired permission settings.

You can change the permission code of a file or directory only if you own it or if you are operating with superuser authority.

Flags

-f	If the **-f** option is given, no errors are reported if **chmod** fails to change the mode on a file.
-R	When the **-R** option is given, **chmod** recursively descends its directory arguments, setting the mode for each file as described above. When symbolic links are encountered, their mode is not changed and they are not transversed.

Subtopics

1.1.67.1 Symbolic Mode

When you use the symbolic mode to specify permission codes, the first set of flags selects the permission field, as follows:

u	User (owner)
g	Group
o	All others
a	User, group, and all others.

If the permission field is omitted, the default is **a**, but the file creation mask (umask) is applied.

The second set of flags selects whether permissions are to be taken away, added, or set exactly as specified:

-	Removes specified permissions.
+	Adds specified permissions.
=	Clears the selected permission field and sets it to the code specified. If you do not specify a permission code following the **=**, the **chmod** command removes all permissions from the selected field.

The third set of flags of the **chmod** command selects the permissions as follows:

r Read permission.

w Write permission.

x Execute permission for files; search permission for directories.

X Set execute permission for files; search permission for directories. Set execute permission only if the file is a directory or at least one execute bit is set.

s Set user-ID or set group-ID permission. This permission bit sets the effective user-ID or group-ID to that of the **file** whenever the **file** is run. Use this permission setting in combination with the **u** or **g** field to allow temporary or restricted access to files not normally accessible to other users. An **s** appears in the user or group execute position of a long listing (see "ls, lf, lr" in topic 1.1.252 or "li, di" in topic 1.1.230) to show that the file runs set-user-ID mode or set-group-ID mode.

t Save text permission. Setting this permission bit causes the text segment of a program to remain in virtual memory after its first use. The system thus avoids having to transfer the program code of frequently accessed programs into the paging area. You can specify this permission only with the **u** field and only if you have superuser authority. (Except for directories. See below.) A **t** appears in the execute position of the "all others" field to indicate that the file has this bit (the "sticky" bit) set.

u The file permissions for user taken from the current mode.

g The file permissions for group taken from the current mode.

o The file permissions for other taken from the current mode.

You can specify multiple symbolic modes, separated with commas. Do not separate items in this list with spaces. Operations are performed in the order they appear from left to right.

AIX gives additional interpretation of the set group-ID and save text permission bits for certain file types or when used with certain other permission bits. The set group-ID permission bit on a regular file, if accompanied by no execute permission bits indicates that file locks should be treated as enforced locks. An S appears in the group execute permission of a long listing to indicate this situation.

The set group-ID permission bit on a directory causes subsequently created files to be assigned the group ID of the directory rather than the effective group IDs of the processes which created the files.

The saved text permission bit on a character special file is used to identify a multiplexed file.

The saved text permission bit on a directory makes it so that only the owner of the directory of the owner of a file within the directory can remove that file from the directory.

1.1.67.2 Absolute Mode

The **chmod** command also permits you to use octal notation to set each bit in the permission code. The **chmod** command sets the permissions to the **permcode** you provide. This **permcode** is constructed by combining (the logical OR of) the following values:

4000	Sets user-ID on execution (when set along with any of the execute (search) bits).
2000	Sets group-ID on execution (when set along with any of the execute (search) bits).
2000	Sets enforcement mode locking (when set without the 0010 bit).
2000	For directories, files take group ID from the directory, not creator.
1000	Retains memory image after execution (executable file)
1000	Marks a directory so that only owners may remove files.
1000	Indicates multiplexed character special file
0400	Permits read by owner
0200	Permits write by owner
0100	Permits execute or search by owner
0040	Permits read by group
0020	Permits write by group
0010	Permits execute or search by group
0004	Permits read by others
0002	Permits write by others
0001	Permits execute or search by others

All permission bits not explicitly specified are cleared.

Examples

1. To add a type of permission to several files:

   ```
   chmod  g+w  chap1 chap2
   ```

   This command adds write permission for group members to the files **chap1** and **chap2**.

2. To change several permissions at once:

   ```
   chmod  go-w+x  mydir
   ```

   This command denies group members and others the permission to create or delete files (**go-w**) in **mydir**. It allows them to search **mydir** or use it in a path name (**go+x**). This one-step change is equivalent to the following sequence of commands:

```
chmod  g-w  mydir
chmod  o-w  mydir
chmod  g+x  mydir
chmod  o+x  mydir
```

3. To permit only the owner to use a shell procedure as a command:

   ```
   chmod  u=rwx,go=  cmd
   ```

   This command gives read, write, and execute permission to the user who owns the file (**u=rwx**). It also denies the group and others the permission to access **cmd** in any way (**go=**).

   If you have permission to execute the shell command file **cmd**, you can run it by entering:

   ```
   cmd
   ```

   Entering this command may not work in some cases, depending on the value of the shell variable **PATH**. See page 1.1.420.10 for more information about **PATH**.

4. To use set-ID modes:

   ```
   chmod  ug+s  cmd
   ```

   When **cmd** is executed, the effective user and group IDs are set to those IDs that own the file **cmd**. Only the effective IDs associated with the subprocess that runs **cmd** are changed. The effective IDs of the shell session remain unchanged.

   This feature allows you to permit restricted access to important files. Suppose that the file **cmd** has the set-user-ID mode enabled and is owned user ID **dbms**. The user **betty** does not have permission to access any of the data files owed by **dbms** She does, however, have permission to execute **cmd**. When she does execute **cmd** her effective user ID is temporarily changed to **dbms** so that the **cmd** program can access the data files owned by **dbms**. Thus, **betty** can use **cmd** to access the data files, but she cannot accidentally damage them with the standard shell commands.

5. To use the absolute mode form of the **chmod** command:

   ```
   chmod  644  text
   ```

   This command sets read and write permission for the owner, and it sets read-only mode for the group and others.

6. To change the permissions of a file from the permissions of one of the file's current permission fields:

   ```
   chmod g=u text
   ```

   The owner permissions will become the permissions of the group, too. For example, if the permissions of the user field are set to **rwx**, the permissions of the group field will be set to **rwx**. The owner permissions will remain **rwx**.

Related Information

See the following commands: "ls, lf, lr" in topic 1.1.252, "li, di" in topic 1.1.230, and "umask" in topic 1.1.490.

1.1.68 chown

Purpose

Changes ownership of files or directories.

Syntax

```
          +--------+                +-----------+   +-----------+
chown ---¦ +----+ +--- owner ---¦¦          +---¦ file      +---¦
          +-¦ -f +-+             +- .group -+   ¦ directory ¦ ¦
            ¦ -R ¦                             ¦ +-----------+ ¦
           ¦+----+¦                            +---------------+
           +------+
```

```
----------------
¦ No space before this option.
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **chown** command changes the **owner** of the specified **files** and **directories** to **owner**. The **owner** may be either a decimal **user ID** or a **user name**. An optional **group** may also be specified. The **group** may be either a decimal **group ID** or a **group name** Only the superuser can specify a change in ownership.

Flags

-f	No errors are reported.
-R	Recursively descends through its directory arguments, setting the specified owner. When symbolic links are encountered, their ownership is changed, but they are not traversed.

Example

```
  chown  tom  program.c
```

The user access permissions for **program.c** now apply to **tom**. As the owner, **tom** can use the **chmod** command to permit or deny other users access to **program.c**. See "chmod" in topic 1.1.67 for details.

Files

/etc/passwd	File that contains user IDs.

Related Information

See the following commands: "passwd, chfn, chsh" in topic 1.1.312.

See the **chown** and **chownx** system calls and the **passwd** file in *AIX Operating System Technical Reference*.

1.1.69 chparm

Purpose

Changes or examines system parameters.

Syntax

```
          +----- nodename ------+   +-- /unix.std ---+
chparm ---¦                     +---¦                +---¦
          +- nodename=newvalue -+   +- kernel-image -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **chparm** command lets you change a system parameter or look at its current setting. Currently, only the **nodename** parameter may be examined or changed. The name assigned cannot be longer than eight characters. If you do not assign a **newvalue**, **chparm** writes the current value of **nodename** to standard output. The default **kernel-image** is **/unix.std**.

Changes affect the running system. Changing **nodename** causes the system to be brought down and rebooted. When changing **nodename**, be sure all users on the network are aware of this change and are logged off from that node. Inform the users they should not change any of the system files mentioned below while this operation is in progress.

Changing the **nodename** also changes the appropriate entries in the following system files:

/etc/filesystems

/etc/fsmap

/etc/hosts

/etc/hosts.equiv

/etc/init.state

/etc/site

/etc/sitegroup

/etc/timesync/sitelist

/generic/dev/nodename

/generic/devs.linst

/local/ident

/local/ports

/local/rc.tcpip.local

/local/system

/local/unix.std

/usr/adm/uucp/Devices

/usr/adm/uucp/Spools

/usr/lib/INnet/connect.con

Note: Some of these files may not exist on your system configuration.

The following files, if they exist on your system configuration, should be edited manually:

/etc/sites

/etc/resolv.conf

/local/named.local and other name daemon files that are created by the system administrator.

Examples

1. To display the **nodename** of your system:

   ```
   chparm nodename
   ```

   This displays the **nodename** of **/unix.std**, which is a file containing the kernel of the AIX operating system. This file is loaded and run when you startup the computer.

2. To change the **nodename** of a system:

   ```
   chparm nodename=COMP-CTR /unix.compctr
   ```

   This changes the **nodename** of **/unix.compctr** to COMP-CTR. **/unix.compctr** is a file that contains an alternate version of the operating system kernel. This change requires that the system be restarted with the kernel image specified on the command line.

1.1.70 chroot

Purpose

Changes the root directory of a command.

Syntax

chroot -- **directory** -- **command** --¦

Description

Warning: If special files in the new root directory have different major and minor device numbers than they have in the real root directory, it is possible to overwrite the file system.

The **chroot** command can be used only by a user operating with superuser authority (see "su" in topic 1.1.449). If you have superuser authority, the **chroot** command changes the root directory to the specified **directory** when executing **command**. The first / (slash) in any path name changes to **directory** for the specified **command** and any of its children. The **chroot** command simulates a call to the **setlocal** with the process's current <LOCAL> alias. As a result, the alias is re-evaluated within the new root directory.

Notice that:

chroot directory command > file

creates the **file** relative to the original root directory, not the new one.

The **directory** path name is always relative to the current root directory. Even if a **chroot** command is in effect, **directory** is relative to the current root directory of the running process.

Several programs may not operate properly after the **chroot** command runs. For example, the command **ls -l** fails to give user and group names if the location of the current root directory makes the **/etc/passwd** beyond reach. In addition, utilities that depend on description files produced by the **ctab** command (see page 1.1.103) may fail altogether if these files are also not in the new root file system. It is your responsibility to ensure that all vital data files are present in the new root file system and that the path names accessing such files are changed as necessary.

Examples

1. To run a subshell with another file system as the root:

```
chroot  /diskette0  /bin/sh
```

   This makes the directory name / refer to **/diskette0** for the duration of the command **/bin/sh** It also makes the original root file system inaccessible. The file system on **/diskette0** must contain the standard directories of a root file system. In particular, the shell looks for commands in **/bin** and **/usr/bin** on the **/diskette0** file system.

   Running the command **/bin/sh** creates a subshell, which runs as a separate process from your original shell. Press the END OF FILE (**Ctrl-D**) key to end the subshell and go back to where you were in the original shell. This restores the environment of the original shell, including the meanings of the current directory (.) and the root directory (/).

2. To run a command in another root file system and save the output:

   ```
   chroot  /diskette0  /bin/cc  -E  /u/bob/prog.c  >prep.out
   ```

   This command runs the **/bin/cc** command with / referring to **/diskette0**. It saves the output in the file **prep.out**, which is in the original root file system.

   This command runs also the C language preprocessor (**/bin/cc -E**) on the file **/diskette0/u/bob/prog.c**, reading **#include** files from **/diskette0/usr/include**, and putting the preprocessed text in **prep.out** on the primary root file system.

Related Information

See the following commands: "cc" in topic 1.1.52, "cpp" in topic 1.1.94, and "sh, Rsh" in topic 1.1.420.

See **chdir** and **chroot** system calls in *AIX Operating System Technical Reference*.

1.1.71 cleanloc

Purpose

Moves commit cluster information from local filesystem.

Syntax

```
                          +--------+
/etc/lpp/cleanloc ---¦ one of +--- dev --- lpp_name ---¦
                          ¦ +----+ ¦
                          +-¦ -c +-+
                            ¦ -r ¦
                            +----+
```

Description
The **cleanloc** command is used on an individual site to move local commit cluster information from the local file system to external storage media (for example tape). The main purpose of such action is to open room in the local file system so that it does not get clogged up with backout information.

When the **-c** flag is used, the commit information from a particular LPP **lpp_name,** is moved from the local file system to the media referred to by **dev**.

When the **-r** flag is used, all commit information on the media, **dev,** is restored to the local filesystem.

Note: When the LPP information is cleaned from a local filesystem, uncommitting (backing out) updates to an LPP is no longer possible. Therefore, you must return the commit information for that LPP to the local filesystem before attempting to uncommit (back out) that LPP. Should you try to uncommit (back out) an LPP whose local commit information is missing, a queue processing error occurs. **lconfig** analysis reveals which local commit information is missing and needs to be restored using **cleanloc -r**.

1.1.72 cleanup

Purpose

Moves backout cluster information from file system.

Syntax

```
                     +-- -c --+
/etc/lpp/cleanup ---¦         +--- dev --- frame_number ---¦
                     +-- -r --+
```

Description

This command is a tool used to move backout stack frames from the file system to storage media (for example, tape). The main purpose of such action is to open up room in the file system so it does not get clogged up with backout information.

Note: When the LPP's information is cleaned from the backup stack, uncommitting (backing out) updates to an LPP is no longer possible. Therefore, you must return the commit information for that LPP to the backout stack before attempting to uncommit (back out) that LPP. When you attempt an uncommit (back out), the stack is automatically checked for cleaned up information. If any information exists, you are notified and asked to restore them using **cleanup -r**.

If you need to free the space used by the last update, you should use the **cleanup** program.

For each LPP installed and for each invocation of **updatep -c** to commit an LPP (or group of LPPs applied together), a **backout stack frame** is created as a subdirectory under **/usr/lpp.save**. A stack frame contains information which permits an update to be uncommitted. Stack frames are sequentially numbered and allocated in a LIFO (Last In First Out) stacked manner.

The **/etc/lpp/cleanup** utility enables stack frames to be archived to tape or diskette so that valuable root file space can be released for subsequent updates. Prior to applying a PTF update, the following commands will archive all stack frames to blank tape mounted on **/dev/rmt0**. Be sure the tape in the drive is write-enabled in ready status, and is attached to the AIX/370 guest at address 380.

```
/etc/lpp/cleanup -c /dev/rmt0 [0-9]
```

Note: The above **/dev/rmt0 [0-9]** is correct shell syntax.

After **cleanup** is run, further updates may be applied and committed. Note that new stack frame numbers will be allocated to continue the previously established sequence; numbering does not revert to 1.

If updates committed before running **cleanup** subsequently need to be uncommitted or rejected, the archived stack frames must first be restored from the tape.

Flags

-c Backout information from one or more frames is moved from the backout stack to the media referred to by a device, specified by **dev**.

-r	Backout information on the media referred to by the **dev** parameter is restored to the backout stack.

1.1.73 clear

Purpose
Clears screen.

Syntax

```
clear ---¦
```

Description
The **clear** command clears your screen if this is possible. It looks in the environment for the terminal type and then in **/usr/lib/terminfo** to figure out how to clear the screen.

Files

/usr/lib/terminfo Terminal information data base.

Related Information
See the following command: "tput" in topic 1.1.478.

1.1.74 clri

Purpose
Clears a specified inode.

Syntax

```
        +--------+
clri ---¦ one of +-- filesystem -- inumber --¦
        ¦ +----+ ¦                           ¦
        +-¦ -f +-+                 +---------+
          ¦ -q ¦
          +----+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

Warning: Use this command only in emergencies and with extreme care.

The **clri** command is used to clear inode entries for files that do not appear in a directory. In general, you do not need to use this program because, the **fsck** command can deal with most file system inconsistencies.

Running the **clri** command can create dangling references or missing blocks. Therefore, after you run this command, you should run the **fsck** command on the file system. These can be fixed if they are attended to promptly. Do not run the system when the file system has dangling directory references or a bad free list.

The **clri** command zeroes over the flags word of the inode, thus freeing the inode for reallocation. The **inumber** parameter specifies the inode and **filesystem** specifies the file system the inode is on. The **inumber** parameter should be a decimal number, while **filesystem** can be either the name of the device on which the file system resides or the name by which it is normally mounted. In the latter form, the file system must be one normally mounted on the local cluster site.

If you use the **clri** command to remove an inode that does appear in a directory, you should track down and remove all inode entries. Otherwise, when the inode is reallocated to a new file, the old entry still points to that file. At that point, removing the old entry destroys the new file and the new entry again points to an unallocated inode.

By default, the **clri** command displays information about the file and asks for confirmation before it destroys the file. If you enter a **y** or **yes**, the file is destroyed.

Note: If the file is open, the **clri** command is likely to be ineffective. For this reason, you should run this command only on an unmounted file system.

Flags

-f Destroys the file without confirmation, but writes a description of the file.

-q Destroys the file without confirmation, but does not write a description of the file.

Example

To clear inodes **170** and **368** of the file system **/diskette0** and then clean up the file system:

```
  clri  /diskette0  170  368
  fsck  /diskette0
```

Related Information

See the following commands: "fsck, dfsck" in topic 1.1.177 and "fsdb" in topic 1.1.178.

See the **fs** file in *AIX Operating System Technical Reference*.

1.1.75 clusterstart, clusterstop

Purpose

clusterstart enables cluster communication.
clusterstop disables cluster communication.

Syntax

```
clusterstart ----¦

clusterstop  ----¦
```

Description

The **clusterstart** command enables TCF cluster communication and telecommunications adapter (TCA) activity. If cluster communication was previously enabled, this program has no effect. Conversely, **clusterstop** disables TCF cluster communication. If cluster communication was previously enabled, the site on which **clusterstop** was run attempts to notify the other sites that it is dropping off of the network. If cluster communication was not previously enabled, this program has no effect.

A system will not be able to communicate to any other machines until a **clusterstart** is done, hence it is generally done before the site enters multi-user state. For a diskless system (one which does not have its own copies of the root file systems), the **clusterstart** operation is done automatically in the kernel.

Under AIX/370, the **clusterstart** command also enables TCA traffic but **clusterstop** does not disable it. This enables traffic so that AIX Access for DOS (file service and terminal emulation) can communicate with AIX/370 over aTCA line. Therefore, AIX/370 systems that are not to participate in cluster communication should do a **clusterstart** followed by an immediate **clusterstop** before entering multi-user mode.

Diagnostics

clusterstart and **clusterstop** will both issue a message indicating the previous status of traffic.

Related Information

See the following command: "probe" in topic 1.1.330.

1.1.76 cmp

Purpose
Compares two files.

Syntax

```
       +--------+
cmp ---¦ one of +-- file1 -- file2 --¦
       ¦ +----+ ¦
       +-¦ -l +-+
         ¦ -s ¦
         +----+
```

Description
The **cmp** command compares **file1** and **file2** and writes the results to standard output. If you specify a - (minus) for **file1**, the **cmp** command reads standard input. Under default conditions, the **cmp** command displays nothing if the files are the same. If they differ, the **cmp** command displays the byte and line number where the first difference occurs. If one file is an initial subsequence of the other (that is, if the **cmp** command reads an end-of-file character in one file before finding any differences), the command notes this. Normally, you use the **cmp** command to compare non-text files and the **diff** command to compare text files.

Flags

-l Displays, for each difference, the byte number in decimal and the differing bytes in octal.

-s Returns only an exit value. (0 indicates identical files; 1 indicates different files; 2 indicates inaccessible file or a missing argument).

Examples

1. To determine whether two files are identical:

   ```
   cmp  prog.o.bak  prog.o
   ```

   This command compares **prog.o.bak** and **prog.o**. If the files are identical, a message is not displayed. If the files differ, the location of the first difference is displayed. For instance:

   ```
   prog.o.bak  prog.o  differ:  char  5,  line  1
   ```

   If the message **cmp: EOF on prog.o.bak** is displayed, the first part of the file **prog.o** is identical to the file **prog.o.bak**, but there is additional data in the **prog.o** file to compare.

2. To display each pair of bytes that differ:

   ```
   cmp  -l  prog.o.bak  prog.o
   ```

   This command compares the files, and then displays the byte number (in decimal) and the differing bytes (in octal) for each difference. For example, if the fifth byte is octal **101** in the file **prog.o.bak** and **141** in the file **prog.o**, **cmp** displays:

```
      5  101  141
```

3. To compare two files without writing any messages:

   ```
   cmp  -s  prog.c.bak  prog.c
   ```

   This command issues an exit value of **0** if the files are identical, a **1** if different, or a **2** if an error occurs. This form of the command is normally used in shell procedures. For example:

   ```
   if  cmp  -s  prog.c.bak  prog.c
   then
         echo  No  change
   fi
   ```

   This partial shell procedure displays **No change** if the two files are identical. See page 1.1.420.22 for details about the **if** command.

Related Information

See the following commands: "comm" in topic 1.1.83, "diff" in topic 1.1.124, and "sh, Rsh" in topic 1.1.420.

1.1.77 col

Purpose
Processes text having reverse line feeds and forward and reverse half-line feeds for output to standard output.

Syntax

```
       +-----------+
col ---¦ +-------+ +---¦
       +-¦ -b -p +-+
         ¦ -     ¦¦
        ¦+-------+¦
        +---------+
```

Note: This command does not have MBCS support.

Description
The **col** command reads from standard input and writes to standard output. It performs the line overlays implied by reverse line feeds (ASCII ESC-7), and by forward and reverse half-line feeds (ASCII ESC-9 and ASCII ESC-8). The **col** command is particularly useful for filtering multi-column output made by the **nroff.rt** command and output from the **tbl** command. The input format accepted by the **col** command matches the output format produced by the **-T37** or by the **-Tlp** flag in the **nroff** command. Use the **nroff-T37** and the **col -f** commands if the output is being sent to a device that can interpret half-line motions; use the **nroff-Tlp** command.

The **col** command assumes that the ASCII control characters SO (\017) and SI (\016) begin and end text in an alternate character set. The **col** command keeps track of the character set each input character belongs to, and, on output, generates SI and SO characters as appropriate to ensure that each character is printed in the correct character set.

On input, the **col** command accepts only the control characters for space, backspace, tab, return, the new-line character, SI, SO, VT, and ESC-7, 8, or 9. VT (\013) is an alternate form of full reverse line feed included for compatibility with some earlier programs of this type. The **col** command ignores all other non-printing characters.

Notes:

1. The maximum number of lines that can be backed up is 128.

2. Up to 800 characters, including backspaces, are allowed on a line. If you are using a system that supports a multibyte character set, the 800-character limit can be reduced by as much as 50%, depending on the character code set being used.

3. Local vertical motions that would result in backing up over the first line are ignored. As a result, the first line must not contain any superscripts.

Flags

-b Assumes that the output device in use is not capable of backspacing. In this case, if two or more characters are to appear in the same position, only the last one read appears in the output.

-f Suppresses the default treatment of half-line motions in the input. Normally, the **col** command does not emit half-line motions on output, although it does accept them in its input. With this flag, output may contain forward half-line feeds (ESC-9) but not reverse line feeds (ESC-7 or ESC-8).

-p Displays unknown escape sequences as characters, subject to overprinting from reverse line motions. Normally, the **col** command ignores them. You should be fully aware of the textual position of escape sequences before you use this flag.

Related Information

See the following commands: "nroff, troff" in topic 1.1.301 and "tbl" in topic 1.1.463.

See the discussion of **col** in the *Text Formatting Guide*.

1.1.78 colcrt

Purpose
Provides virtual line feeds.

Syntax

```
          +--------+
colcrt ---¦ +----+ +--- file ---¦
          +-¦ -  +-+
            ¦ -  ¦¦
           ¦+----+¦
           +------+
```

Note: This command does not have MBCS support.

Description
The **colcrt** command provides virtual half-line and reverse line feed sequences for terminals without such capability, and on which overstriking is destructive. Half-line characters and underlining (changed to dashing '-') are placed on new lines in between the normal output lines.

Flags

-	Suppresses underlining; useful for previewing boxed tables from **tbl**.
-2	Causes all half-lines to be printed, effectively double-spacing the document. Useful when printing output with subscripts and superscripts on a line printer, where half lines normally do not appear.

Examples

A typical use of **colcrt** would be

```
      tbl exum2.n | nroff -ms | colcrt - | more
```

Related Information
The following commands: "nroff, troff" in topic 1.1.301, "col" in topic 1.1.77, "more, page" in topic 1.1.277 and "ul" in topic 1.1.489.

1.1.79 colpro

Purpose
Column filter for IBM 4201 Proprinter.

Syntax

```
                          +--------+
/usr/lib/ibmlp/colpro ---¦ +----+ +---¦
                          +-¦ -b +-+
                            ¦ -f ¦¦
                           ¦¦ -h ¦¦
                           ¦+----+¦
                           +------+
```

Note: This command does not have MBCS support.

Description
The **colpro** command reads the standard input and writes the standard output. It performs the line overlays implied by reverse line feeds (**Esc-7** in ASCII) and by forward and reverse half line feeds (**Esc-9** and **Esc-8**). **colpro** is particularly useful for filtering multiple-column output made with the **nroff.rt** command of **nroff** and output resulting from use of the **tbl** preprocessor. **colpro** is intended for use with **prfl**.

Although **colpro** accepts half line motions in its input, it normally does not emit them on output. Instead, text appearing between lines is moved to the next lower full line boundary. This treatment can be suppressed by the **-f** (fine) option. In this case the output from **colpro** may contain forward half line feeds **Esc-9**, but will still never contain either kind of reverse line motion.

If the **-b** option is given, **colpro** assumes the output device in use is not capable of backspacing. In this case, if several characters are to appear in the same place, only the last one read will be taken.

The **colpro** command may also be used with the IBM 5152 Graphics Printer. Even though the graphics printer cannot back up, the **-b** option should not be used since backspaces tell the post filter to use underlining.

Esc sequences are treated according to one of three rules:

1. **Esc-7**, **Esc-8**, and **Esc-9** are line positioning commands and are treated as above.

2. **Esc-Ax** sequences are assumed to be printer control sequences (initialize printer, start bold, stop bold, etc.). These are echoed to standard output and are treated as if they have "0" width.

3. All other **Esc** sequences are assumed to be 3 characters long (for example **Esc c1 c2**), and are echoed to standard output as is (even if one of the characters is a control character). The filter assumes the three-character escape sequence specifies a single printable character.

If the **-h** option is given, **colpro** converts white space to tabs to shorten printing time.

No control characters are passed to the output except space, backspace, tab, return, newline, **Esc** (033) followed by any 2 characters (except 7, 8, or 9) and VT (013). This last character is an alternate form of full reverse line feed, for compatibility with some other hardware conventions. All other non-printing characters are ignored.

The **colpro** command is provided primarily for use by **proff**.

Related Information

See the following commands: "col" in topic 1.1.77, "nroff, troff" in topic 1.1.301, "proff" in topic 1.1.333 and "tbl" in topic 1.1.463.

1.1.80 colrm

Purpose
Extracts columns from a file.

Syntax

```
                  +--------+
colrm --- first ---¦        +---¦
                  +- last -+
```

Note: This command does not have MBCS support.

Description

The **colrm** command removes selected columns from a file. Input is taken from standard input. Output is sent to standard output.

If called with one parameter (**first**) the columns of each line will be removed starting with the specified column. If called with two parameters the columns from the **first** column to the **last** column will be removed.

Column numbering starts with column 1.

Related Information
See the following command: "cut" in topic 1.1.107.

1.1.81 comb

Purpose
Combines Source Code Control System (SCCS) deltas.

Syntax

```
        +-------------+
comb ---¦ +---------+ +--- file ---¦
        +-¦ -o      +-+
          ¦ -s      ¦¦
         ¦¦ -p SID  ¦¦
         ¦¦ -c list ¦¦
         ¦+---------+¦
         +-----------+
```

Note: This command does not have MBCS support.

Description
The **comb** command writes to standard output a shell procedure that can combine the specified deltas (**SID**s) or all deltas into one delta. You may reduce the size of your SCCS file by running the resulting procedure on the file. You can see how much the file will actually be reduced by running the command with the **-s** flag. If you specify a directory in place of **file**, the **comb** command performs the requested actions on all SCCS files (that is, those with file names with the **s.** prefix). If you specify a **-** (minus) in place of **file**, the **comb** command reads standard input and interprets each line as the name of an SCCS file. The **comb** command continues to take input until it reads END OF FILE (**Ctrl-D**).

If you do not specify any flags, the **comb** command preserves only leaf deltas and the minimal number of ancestors needed to preserve the tree (see "delta" in topic 1.1.117).

Note: The **comb** command may rearrange the shape of the tree deltas. It may not save any space; in fact, it is possible for the reconstructed file to actually be larger than the original.

Flags
Each flag or group of flags applies independently to each named file.

-c list Specifies a list of deltas (**SID**s) that the shell procedure preserves (see the **-ilist** flag the SID list format in topic 1.1.186.2). The procedure combines all other deltas.

-o Accesses the reconstructed file at the release of the delta to be created for each **get -e** command generated; otherwise, accesses the reconstructed file at the most recent ancestor. Using the **-o** flag may decrease the size of the reconstructed SCCS file. It may also alter the shape of the delta tree of the original file.

-p SID Specifies the **SID** of the oldest delta for the resulting procedure to preserve. All older deltas are combined in the reconstructed file.

-s Causes the **comb** command to generate a shell procedure that produces a report that for each file gives the file name, size (in blocks) after combining, original size (in blocks), and percentage of change computed by the formula:

```
100 * (original - combined) / original
```

You should run the **comb** command using this flag and run its procedure before combining SCCS files in order to judge how much space will actually be saved by the combining process.

Files

s.COMB	The name of the reconstructed SCCS file.
comb*	Temporary files.

Related Information

See the following commands: "admin" in topic 1.1.16, "delta" in topic 1.1.117, "get" in topic 1.1.186, "sccshelp" in topic 1.1.411, and "prs" in topic 1.1.336.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.82 comlist

Purpose
Finds recently committed files.

Syntax

comlist --- **gfs_number --- site_number --- low_water_mark** ---¦

Description
The **comlist** command scans a device to determine what files have had recent commits. It assumes that it is running on the site storing the device to be scanned. The device is assumed to be mounted and is identified via the specified global file system number, **gfs_number**. The **site_number** argument is largely ignored except for diagnostic messages. The **comlist** command displays inode number and low-water-mark pairs for every file on the device with a low water mark greater than **low_water_mark**.

This command is not intended for use by anyone other than the system administrator.

Related Information
See the following commands: "primrec" in topic 1.1.325 and "recmstr" in topic 1.1.364.

1.1.83 comm

Purpose
Selects or rejects lines common to two sorted files.

Syntax

```
       +----------------+
       ¦     one of     ¦
comm --¦+--------------++--- file1 --- file2 ---¦
       ¦¦ -1   -2  -3  ¦¦
       +¦ -12  -13 -23 ++
        ¦ -123         ¦
        +--------------+
```

Description
The **comm** command reads **file1** and **file2** and writes, by default, a three-column output to standard output. The columns consist of:

1. Lines that are only in **file1**
2. Lines that are only in **file2**
3. Lines that are in both **file1** and **file2**.

If you specify - (minus) for one of the file names, the **comm** command reads standard input. Both **file1** and **file2** should be sorted according to the collating sequence specified by the environment variables **LANG** and **LC_COLLATE** (see "sort" in topic 1.1.432).

Flags

-1	Suppresses the display of the first column (lines in **file1**).
-2	Suppresses the display of the second column (lines in **file2**).
-3	Suppresses the display of the third column (lines common to **file1** and **file2**).

Note: Specifying **-123** does nothing (a no-op).

Examples

1. To display the lines unique to each file and common to both:

   ```
   comm  things.to.do  things.done
   ```

   If the files **things.to.do** and **things.done** contain:

things.to.do	**things.done**
buy soap	2nd revision
groceries	interview
luncheon	luncheon
meeting at 3	system updat
system updat	tech. review
tech. review	weekly repor

   then **comm** displays:

```
                2nd revision
         buy soap
         groceries
                interview
                        luncheon
         meeting at 3
                        system update
                        tech. review
                weekly report
```

The first column contains the lines found only in the file **things.to.do**. The second column, indented with a tab character, lists the lines found only in the file **things.done**. The third column, indented with two tabs, lists the lines common to both files.

2. To display the lines that appear in only one file:

```
comm  -23  things.to.do things.done
```

This command suppresses the display of second and third columns of the **comm** command listing. If the files are the same as in Example 1, the following lines are displayed:

```
buy soap
groceries
meeting at 3
```

Related Information

See the following commands: "cmp" in topic 1.1.76, "diff" in topic 1.1.124, "sdiff" in topic 1.1.413, "sort" in topic 1.1.432, and "uniq" in topic 1.1.495.

See the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See "Overview of International Character Support" in *Managing the AIX Operating System*.

1.1.84 commit

Purpose
Commits a file within a shell procedure.

Syntax

```
          +------------+
commit ---¦            +---¦
          +- filedesc -+
```

Description

The **commit** command invokes the commit operation on the file descriptors specified. This is useful when a shell file calls another shell file, redirecting output to another file. If the **commit** command is not performed in the shell file being called, some of the output may be lost if the system goes down.

The files are specified as file descriptors. **STDIN** or **stdin** may be used for standard input, **STDOUT** or **stdout** may be used for standard output, and **STDERR** or **stderr** may be used for standard error. If no files are specified, commits are performed on **stdout** and **stderr**.

Related Information
See **fsync, fcommit** in *AIX Operating System Technical Reference.*

1.1.85 comp

Purpose
Composes a message.

Syntax

```
        +------------- -form file --------------+
comp ---¦ +-----------+   +------- cur -------+ +---
        +-¦           +---¦      one of       +-+
          +- +folder -+   ¦ +---------------+ ¦
                          +-¦ num      cur  +-+
                            ¦ sequence .    ¦
                            ¦ first    next ¦
                            ¦ prev     last ¦
                            +---------------+

    +---------------------------------------------------------------------+
    ¦         one of                                                      ¦
 ---¦   +----------------+                                                +---
    ¦ +-¦ -file file     +----------------------------------------------+ ¦
    +-¦ ¦ -nodraftfolder ¦                                              +-+
      ¦ +----------------+                                              ¦
      ¦                 one of                                          ¦
      ¦ +------------------------------------+   +------- new --------+ ¦
      +-¦ -draftfolder +folder -draftmessage +---¦      one of        +-+
        ¦ -draftfolder +folder               ¦   ¦ +----------------+ ¦
        ¦ -draftmessage                      ¦   +-¦ num      .     +-+
        +------------------------------------+     ¦ sequence next  ¦
                                                   ¦ first    last  ¦
                                                   ¦ prev     new   ¦
                                                   ¦ cur            ¦
                                                   +----------------+

    +------------+   +-----------------+   +----------------------------+
 ---¦   one of   +---¦     one of      +---¦           one of           +---¦
    ¦ +--------+ ¦   ¦ +-------------+ ¦   ¦ +------------------------+ ¦
    +-¦ -use   +-+   +-¦ -editor cmd +-+   +-¦ -whatnowproc cmdstring +-+
      ¦ -nouse ¦       ¦ -noedit     ¦       ¦ -nowhatnowproc         ¦
      +--------+       +-------------+       +------------------------+

comp --- -help ---¦
```

Note: This command does not have MBCS support.

Description

The **comp** command is used to create and modify messages. This command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

By default, the **comp** command copies a message form to a new draft message and invokes an editor. You can then fill in the message header fields **To:** and **Subject:**, fill in or delete the other header fields (such as **cc:** and **Bcc:**), and add the body of the message. When you exit the editor, the **comp** command invokes the MH command **whatnow**. You can specify any of the **whatnow** subcommands, or you can press **Enter** to see a list of the subcommands. These subcommands enable you to continue composing the message, direct the disposition of the message, or end the processing of the **comp** command. See "whatnow" in topic 1.1.533 for a description of the subcommands.

You can specify the **form**, or format, of the message by using the **-form** flag or the **+*folder*** flag. If you do not specify one of these flags, the **comp** command uses your default message format located in the file **user_mh_directory/components**. If this file does not exist, the **comp** command uses the system default message format located in the file **/usr/lib/mh/components**.

You can compose a new message, or you can specify the **-use** flag and continue composing an existing message. The **-file**, **-draftfolder**, and **-draftmessage** flags enable you to specify the new or existing message that you want to compose.

Note: The line of dashes or a blank line must be left between the header and the body of the message for the message to be identified when it is sent.

Flags

-draftfolder +*folder*
Places the draft message in the specified folder. If you do not specify this flag, the **comp** command selects a default draft folder according to the information supplied in the MH profiles. You can define a default draft folder in the **$HOME/.mh_profile** file. If **-draftfolder +*folder*** is followed by *msg*, *msg* represents the **-draftmessage** attribute.

-draftmessage *msg*
Specifies the draft message. You can specify one of the following message references for *msg*:

num	*sequence*	**fi**
prev	**cur**	**.**
next	**last**	**ne**

If the **-use** flag is specified, the default draft message is **cur**. Otherwise, the default draft message is **new**.

-editor *cmd*
Specifies that *cmd* is the initial editor for composing the message. If you do not specify this flag, the **comp** command selects a default editor or suppresses the initial edit according to the information supplied in the MH profiles. You can define a default initial editor in the **$HOME/.mh_profile** file.

-file *file*
Places the draft message in the specified file. If you do not specify the absolute path name for *file*, the **comp** command places *file* in the **user_mh_directory** file. If *file* exists, the **comp** command prompts you for the disposition of the draft.

+*folder msg*
Uses the form of the specified message in the specified folder. You can specify one of the following message references for *msg*:

num	*sequence*	**fi**
prev	**cur**	**.**
next	**last**	

The default message is the current message in the current folder.

-form *file* — Uses the form contained in the specified file. The **comp** command treats each line in *file* as a format string.

-help — Displays help information for the command.

-nodraftfolder — Places the draft in the file **user_mh_directory/draft**.

-noedit — Suppresses the initial edit.

-nouse — Creates a new message.

-nowhatnowproc — Does not invoke a program that guides you through the composing tasks. The **-nowhatnowproc** flag also prevents any edit from occurring.

-use — Continues composing an existing draft of a message.

-whatnowproc *cmdstring* — Invokes *cmdstring* as the program to guide you through the composing tasks. See "whatnow" in topic 1.1.533 for information about the default **whatnow** program and its subcommands.

Note: If you specify **whatnow** for *cmdstring*, the **comp** command invokes an internal **whatnow** procedure rather than a program with the file name **whatnow**.

Profile Entries

Draft-Folder:	Sets your default folder for drafts.
Editor:	Sets your default initial editor.
fileproc:	Specifies the program used to refile messages.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your **user_mh_directory** file.
whatnowproc:	Specifies the program used to prompt "What now?" questions.

Files

/usr/lib/mh/components	The system default message form.
user_mh_directory/components	The user's default message form. (If the file exists, it overrides the system default message form.)
$HOME/.mh_profile	The MH user profile.
user_mh_directory/draft	The draft file.

Related Information
See other MH commands: "ali" in topic 1.1.17, "dist" in topic 1.1.131, "forw" in topic 1.1.174, "prompter" in topic 1.1.334, "repl" in topic 1.1.369, "refile" in topic 1.1.366, "send" in topic 1.1.416,

"whatnow" in topic 1.1.533, and "whom" in topic 1.1.539.

See the **mh-alias**, **mh-format**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.86 compress, uncompress, zcat

Purpose
Reduces the size of named files.

Syntax

```
            +--------------+   +------------+
compress ---¦ +----------+ +---¦            +---¦
            +-¦ -f -v -c +-+   +--- name ---+
              ¦ -b bit   ¦¦             ¦
             ¦+----------+¦      +--------+
             +------------+

              +--------+   +------------+
uncompress ---¦ +----+ +---¦            +---¦
              +-¦ -v +-+   +--- name ---+
                ¦ -c ¦              ¦
               ¦+----+¦      +--------+
               +------+

        +------------+
zcat ---¦            +---¦
        +--- name ---+
                ¦
          +--------+
```

Description
The **compress** command reduces the size of the named files using adaptive Lempel-Ziv coding. Whenever possible, each file is replaced by one with the extension **.Z** while keeping the same ownership modes, access and modification times. If no files are specified, the standard input is compressed to the standard output. Compressed files can be restored to their original form using **uncompress** or **zcat**.

The **compress** command uses the modified Lempel-Ziv algorithm popularized in "A Technique for High Performance Data Compression", Terry A. Welch, **IEEE Computer**, vol. 17, no. 6 (June 1984), pp. 8-19.

The amount of compression obtained depends on the size of the input, the number of **bits** per code, and the distribution of common substrings. Typically, text such as source code or English is reduced by 50-60%. Compression is generally much better than that achieved by Huffman coding as used in **pack** or adaptive Huffman coding (**compact**), and takes less time to compute.

If the program finishes normally, the exit status is 0. The exit status is 1 if an error occurs. If the last file was not compressed because it became larger, the exit status is 2.

Flags

-f	Forces compression of **name** even if it does not actually shrink or the corresponding **name** file already exists. Except when run in the background under **/bin/sh** if **-f** is not given, the user is prompted as to whether an existing **name** file should be overwritten.
-c	Makes **compress/uncompress** write to the standard output; no

files are changed. The nondestructive behavior of **zcat** is identical to that of **uncompress -c**.

-b bits Specifies the maximum number of **bits** to use to replace common substrings in the file. The default for **bits** is 16, with values of 9 through 16 acceptable. First, the algorithm uses 9-bit codes 257 through 512. Then it uses 10-bit codes, continuing until **-b** is reached. (Not permitted with **uncompress**.)

After the **bits** limit is attained, **compress** periodically checks the compression ratio. If it is increasing, **compress** continues to use the existing code dictionary. However, if the compression ratio decreases, **compress** discards the table of substrings and rebuilds it from scratch. This allows the algorithm to adapt to the next "block" of the file.

-v Causes the percentage reduction of each file to print.

Diagnostics

Usage: compress [-fvc] [-b maxbits] [file . . .]
Invalid options were specified on the command line.

Missing maxbits
Maxbits must follow **-b**.

file not in compressed format
The file specified to **uncompress** has not been compressed.

file compressed with xx bits, can only handle yy bits
file was compressed by a program that could deal with more **bits** than the compress code on this machine. Recompress the file with smaller **bits**.

file already has .Z suffix -- no change
The file is assumed to be already compressed. Rename the file and try again.

file already exists; do you wish to overwrite (y or n)?
Respond "y" if you want the output file to be replaced; "n" if not.

uncompress: corrupt input
A SIGSEGV violation was detected which usually means that the input file is corrupted.

Compression: xx.xx%
Percentage of the input saved by compression (relevant only for **-v**).

-- not a regular file: unchanged
When the input file is not a regular file, (for example, a directory), it is left unaltered.

-- has xx other links: unchanged
The input file has links; it is left unchanged. See **ln** for more information.

-- file unchanged

No savings is achieved by compression. The input remains unchanged.

Note

Although compressed files are compatible between machines with large memory, **-b**12 should be used for file transfer to architectures with a small process data space (64KB or less).

1.1.87 comsat

Purpose
Receives reports of incoming mail.

Syntax

```
/etc/comsat ---¦
```

Description

The **comsat** command is the server process which receives reports of incoming mail and notifies users if they have requested this service. **comsat** receives messages on a datagram port associated with the **biff** service specification. The one line messages are of the form:

user@mailbox-offset

If the **user** specified is logged in to the system and the associated terminal has the owner execute bit turned on (by a **biff y**), the **offset** is used as a seek offset into the appropriate mailbox file and the first 7 lines or 560 characters of the message are printed on the user's terminal. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the 560-character limit can be reduced by as much as 50%, depending on the character code set being used. Lines which appear to be part of the message header other than the **From**, **To**, **Date**, or **Subject** lines are not included in the displayed message.

Note: The **comsat** command works only when TCP/IP is installed.

Files

/etc/utmp	To find out who's logged on and on what terminals.

Related Information
See the following command: "biff" in topic 1.1.41.

1.1.88 config

Purpose
Builds kernel configuration modules from system configuration files.

Syntax

```
          +- -m /etc/master -+   +- -c conf.c -+   +- -s summary  -+
config ---¦                  +---¦             +---¦               +---
          +---- -m mfile ----+   +- -c cfile --+   +---------------+

    +---------+
 ---¦ +-----+ +--- systemfile ------¦
    +-¦ -q  +-+
      +-----+
```

Description

The **config** program uses the configuration information in the system and master files to create modules that are used to build a new kernel. It can also produce a list of configured devices used to automate the creation of special files.

The **config** program reads the master file (the default is **/etc/master**) to determine the device drivers available for linking into the kernel and the system parameters that can be specified. It then reads the system file (the default is **/etc/system**) to determine the specific device drivers and system parameters to be used in building a new kernel. Any system parameters not specified in the system file assume default values specified in the master file. A C configuration module (the default is **conf.c**) is then produced. This module can be compiled and used to build a new kernel. A configuration summary is optionally produced in order to build a new kernel when a C compiler is not available.

In the AIX Operating System, the configuration summary is used to build several files:

A linkage editor script to select which device drivers should b included in the new kernel

A linkage editor script to select optional kernel support for license programs such as IBM AIX 370 DOS Merge and IBM AIX 370 TCP/IP

A script for updating configurable parameters in the new kernel

The **newkernel** command automatically runs the **config** command, processes the configuration summary to obtain the scripts described above, and then builds a new kernel.

Flags

-c cfile	Writes the output C configuration module into the named file instead of **conf.c**.
-m mfile	Reads the master configuration menu from the named file instead of **/etc/master**.
-s sfile	Writes a complete configuration summary into the file **sfile**. The configuration summary is not produced if

this flag is not specified.

-q	Suppresses information-only warning messages.

Note: Run print **-rr** on all the sites (nodes) which are listed in the 'node' field of the stanza in the **qconfig** file.

Files

/etc/master	Default master configuration menu.
/etc/system	Default system configuration file.
conf.c	Default C configuration file.

Related Information

See the following command: "newkernel" in topic 1.1.291.

See the **master** and **system** files in *AIX Operating System Technical Reference*.

See "Generating a New Kernel" in *Managing the AIX Operating System*.

1.1.89 conflict

Purpose
Searches for alias and password conflicts.

Syntax

```
                            +--------------+   +--- -search /usr/mail ---+   +- /u
/usr/lib/mh/conflict ---¦              +---¦                        +---¦
                            +- -mail user -+   +--- -search directory ---+   +---
                                                                     ¦
                                                 +---------------------+       +--

/usr/lib/mh/conflict --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **conflict** command is used to find conflicts in aliases and to find invalid mail drops. **conflict** is not designed to be run directly by the user; it is designed to be called by **cron** and other programs used for system accounting. The **conflict** command is a system administrator command that is usually invoked by its full path name. The **conflict** command is part of the Message Handling (MH) package.

The **conflict** command searches all specified alias files for duplicate alias names that do not resolve to the same address. By default, **conflict** searches **/usr/lib/mh/MailAliases**. **conflict** also searches all specified maildrop directories for mailbox files with names that do not correspond to valid users defined in **/etc/passwd**.

The **conflict** lists its output on the display, unless you specify the **-mail** flag. **-mail** causes **conflict** to mail its output to the specified user.

Flags

-help	Displays help information for the command.
-mail user	Sends the results of the **conflict** command to the specified user.
-search directory	Searches the indicated directories for invalid mailboxes. You can specify any number of **-search** flags. The default mailbox directory is **/usr/mail**.

Files

/usr/lib/mh/mtstailor	The MH tailor file.
/etc/passwd	List of users.
/etc/group	List of groups.
/usr/mail/$USER	The location of the mail drop.

Related Information
See MH commands: "ali" in topic 1.1.17 and "whom" in topic 1.1.539.

See **mh-alias, mh-mail,** and **mh-profile** files in *AIX Operating System Technical Reference.*

See *"Overview of the Message Handling Package"* in *Managing the AIX Operating System*.

1.1.90 connect

Purpose
Establishes a connection to a remote system.

Syntax

```
                +---------+
connect -- -b --¦         ¦+-- rmthost --¦
                +- file: -+

           +-------- -zO --------+   +--------+   +--------+
connect ---¦                     +---¦ one of +---¦ one of +---
           ¦ +-----------------+ ¦   ¦ +----+ ¦   ¦ +----+ ¦
           +-¦ -iname -mprompt +-+   +-¦ -f +-+   +-¦ -q +-+
             ¦ -wse   -parg    ¦¦      ¦ -h ¦       ¦ -d ¦
            ¦¦ -targ  -xarg    ¦¦      +----+       +----+
            ¦+-----------------+¦
            +-------------------+

    +----------------+   +----------------+   +--- rmthost ----+  +---------
 ---¦      +-------+ +---¦      +--------+ +---¦                +--+- !cmd  --
    +- -e -¦¦      +-+   +- -s -¦¦       +-+   +- file:rmthost -+  +- -rpgm --
           +- esc -+            +- rate -+
```

¦ Do not put a blank between these lines.

Description
The **connect** command lets you establish a connection to a remote host, one which is usually not a site in your cluster. The **connect** command runs in two parts. The first part makes the connection with the remote system specified by **rmthost**. The second part is a program called the **talker**, which runs automatically and exchanges data with the **rmthost**. For information about the **talker** program, see the **connect** command in *AIX Operating System Technical Reference*. Any flags that you specify are passed directly to the **talker** without interpretation. The default **talker** for asynchronous links is **atalk**.

The **connect** command uses a system-wide control file, **connect.con**, which is located in the **/usr/lib/INnet** file. You can specify an additional control file, **file:rmthost**. If you do not specify an additional file, the **connect** command searches the **$HOME/bin** file for a **connect.con** file. Information needed to complete the connection is found in one of these files.

Attributes needed to complete the connection are taken from the control file or from the command line assignment **var=val**. For a description of the parameters, see the **connect** command in *AIX Operating System Technical Reference*.

When the **atalk** program detects an escape sequence in the input, it places the work station in its former mode of operation and prompts you with the local prompt. You can then use the flags that follow. Once the flag has run, **atalk** returns to its former mode.

The **connect** command does not limit access to the phone system to control dialing based on the number to be called.

Warning: The **connect** command lets you set up and maintain connections through a wide variety of communications devices. It interacts with you through the file **connect.con** which is free-format. Problems with the format of this file may cause unpredictable results.

Flags

Note: There are no spaces between the flags and the associated parameters.

-b Sends a break to the port. This is done by lowering the transmission speed to 75 bps and transmitting an ASCII NULL on the port. If the speed is too low (that is, less than 100 bps), this may not work.

-d -q Closes, quits (**q**), or disconnects (**d**) the port. This does **not** end your job or session at the remote site. After closing the port, the **connect** command exits.

-e[esc] Sets the escape sequence to the character string **esc**. If you do not specify **esc**, the **connect** command displays the escape sequence. It either takes the default escape sequence from the environment variable **CONESC**, if defined, or sets the escape sequence to:

Ctrl-VuCtrl-M

-f Sets full duplex mode. Disables local echoing.

-h Sets half duplex mode. Enables local echoing.

-iname Writes file **name** to the port.

Warning: If you are connected to the remote host by RS-232 lines, data from the file may be lost if the remote host cannot keep up with the input.

Normally, this flag is used to transfer a small file from the local site to the remote site. File transmission must be ended manually by pressing **Ctrl-D**. For example:

```
cat > newfile
[escape sequence]
LOCAL: ifred
.
.
.
```

Ctrl-D

-mprompt Sets the local prompt to the **prompt** character string. The **connect** command displays this prompt when it recognizes the escape sequence. By default, it sets the prompt to the value of the environment variable **CONPMT**. If this variable is not set, it uses the **LOCAL:** string.

-parg Sets parity as specified by **arg**, where **arg** is one of the following characters: **o** (odd), **e** (even), **7** (both odd and even), or **8** (eight data bits).

-rpgm Runs the network program **pgm**. Anything following **pgm** on the command line is passed to **pgm** as an argument, along with the additional arguments **-i3 -o3**. The port is set up as file descriptor 3. The program is run as a child process.

-srate Sets the transmission speed to **rate**, which is one of the following: 0, 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, exta, or extb (0 effectively turns off the port). If you do not specify **rate**, the current transmission speed is displayed.

-targ Enables or disables transcripts. If **arg** is any character string other than a - (minus) or + (plus) sign, the transcript function is enabled with the specified file **arg** as a transcript. When you use an existing file as a transcript file, new data is added to its end. Use **t-** to disable the transcript function, and **t+** to enable the transcript to the previous transcript file (no default).

-wsec Sets the inter-line delay of the include function to cause a delay interval of the specified seconds between each line written to the port. The default value is 0.

-xarg Enables or disables input or output flow control. If the input flow control is enabled, **Ctrl-S** and **Ctrl-Q** are automatically sent to the remote host to control the rate at which it transmits data. If the output flow control is enabled, **Ctrl-S** and **Ctrl-Q** are automatically honored if received from the host. This is useful when using the **include** command. **xi+** enables input flow control. **xi-** disables input flow control. **xi** displays the current state. For control of output flow control, replace **xi** with **xo**. See the discussion of IXON and IXOFF in the **termio** file in *AIX Operating System Technical Reference*.

!cmd Runs the AIX command **cmd**. Anything that follows **!**, including arguments to **cmd**, is passed to the local shell to be run by the **system** system call. In particular, all I/O redirection and piping works.

-? Displays a summary of flags.

Files

/usr/lib/INnet/connect.con	System-wide connection control file.
$HOME/bin/connect.con	Private connection control file.
/usr/lib/INnet/dialers/*	System-wide dialer programs.
$HOME/bin/*	Private dialer programs.
/usr/lib/INnet/atalk	Default **talker** program, asynchronous lines.
/etc/sites	Network sites file.
/etc/locks	Directory for locks on ports (devices) used for logins and out-going connections.

Related Information

See the **system** and **exec** system calls, the **connect** subroutine, and the **termio** special facility in *AIX Operating System Technical Reference*.

1.1.91 cp, copy

Purpose
Copies files.

Syntax

```
 one of
+------+   +--------+   +- infile --- directory -+
¦ cp   +---¦ +----+ +---¦          ¦             +---¦
¦ copy ¦   +-¦ -f +-+   ¦+---------+             ¦
+------+     ¦ -i ¦     +-- infile --- outfile --+
            ¦¦ -p ¦¦
            ¦¦ -r ¦¦
            ¦¦ -- ¦¦
            ¦+----+¦
            +------+
```

Note: This command does not have MBCS support.

Description
The **cp** (copy) command copies a source file or the files in a source directory to a target file or directory. If your output is to a directory, the files are copied to that directory with the same file name.

If **infile** is a symbolic link, the link is followed when the copy is performed. If the link cannot be followed, an error is indicated. If **infile** is a hidden directory, a component in that directory is selected. If no selectable component is present, an error is indicated.

If **outfile** is a symbolic link, it is followed when the copy is performed. If **outfile** is a hidden directory, a component in that directory is selected. If the standard input is a work station, the user is asked if that component should be overwritten before the copy is performed. No questions are asked if the **-f** option is given. Components of hidden directories are not created implicitly by the **cp** command. If no selectable component is found in the hidden directory, an error is indicated. Hidden directory components may be created only by **cp** using the '@' character syntax.

You can also copy special device files. If the file is a named pipe, the data in the pipe is copied into a regular file. If the file is a device, the file is read until the end of file is reached and that data is copied into a regular file.

Notes:

1. Do not name **outfile** as one of the input files.

2. If you specify a directory for the **outfile**, the directory must already exist.

3. If the **infile** contains subdirectories and the subdirectories do not exist, the system creates them.

Flags

-f No questions asked if **outfile** is a hidden directory.

-i Prompts the user with the name of the file whenever the copy will cause an existing file to be overwritten. An answer of **y** causes the **cp** command to continue. Any other answer prevents it from overwriting the file.

-p Preserves the modification times and modes of the **infile** for the copy.

-r Copies each subtree rooted at the **infile** (recursive copy). If the **infile** is a directory, the **outfile** must be a directory.

-- Indicates that the arguments following this flag are to be interpreted as file names. This null flag allows the specification of file names that start with a (minus).

Examples

1. To make another copy of a file in the current directory:

   ```
   cp  prog.c  prog.bak
   ```

   This command copies the file **prog.c** to the file **prog.bak**. If the file **prog.bak** does not already exist, the **cp** command creates it. If the file **prog.bak** does exist, the **cp** command replaces it with a copy of **prog.c**.
2. To copy a file to another directory:

   ```
   cp  jones  /u/nick/clients
   ```

   This command copies the file **jones** to the file **/u/nick/clients/jones**.
3. To copy a file to a new file and preserve the modification date and time:

   ```
   cp -p smith smith.jr
   ```

   This command copies the file **smith** to the file **smith.jr**. Instead of creating the file with the current date and time stamp, the system gives the file **smith.jr** the same date and time as the file **smith**.
4. To copy the files and subdirectories in a directory to another new directory (target directory does not previously exist):

   ```
   cp -r /u/nick/clients /u/nick/customers
   ```

   This command creates the new directory **customers** containing copies of the files, subdirectories, and files rooted at the directory **clients**. If **/u/nick/customers** already exists, this command creates a directory **clients** within the directory **customers**.
5. To copy a specific set of files to another directory:

   ```
   cp  jones lewis smith  /u/nick/clients
   ```

   This command copies the files **jones**, **lewis**, and **smith** to the directory **/u/nick/clients**.
6. To use pattern-matching characters to copy files:

   ```
   cp  programs/*.c  .
   ```

   This command copies the files that end with **.c** that are located in the directory **programs** to the current directory (.). You must type a space between the **c** and the final period.

Related Information

See the following commands: "cpio" in topic 1.1.93, "link, unlink" in topic 1.1.232, "ln" in topic 1.1.234, and "mv, move" in topic 1.1.282.

1.1.92 cpcmd

Purpose
Issue a CP (Virtual Machine Control Program) command.

Syntax

```
         +-----------+
cpcmd ---¦           +---¦
         +- command -+
```

Note: This command is for the System/370 only.

Note: This command does not have MBCS support.

Description

AIX/370 runs in a virtual machine which is controlled by CP, the VM/SP or VM/XA Control Program. The **cpcmd** command sends a CP command to the control program, receives the response, and prints it on Unix standard output. This is an AIX/370 command only.

If no command is specified on the command line, **cpcmd** gives a prompt and waits for the user to enter a command at the terminal. It sends the command to CP, prints the response, and issues another prompt. To exit from the **cpcmd** interpreter, the user should enter an empty command line at the prompt.

The superuser is allowed to issue any CP command except **logout** and **disconnect**. Other users can issue only the CP **query** command by typing **q** or **query** followed by one or more spaces and the remainder of the command.

Avoid using **cpcmd** to issue CP commands that require input from the console terminal. For example, the CP command **link** requires users to input a password if it has not been specified on the CP command line. Therefore, users should avoid issuing **link** with the **cpcmd** command.

The **cpcmd** command cannot display CP command output that exceeds 4096 bytes. Any output above this limit is discarded.

When using an * (asterisk) with **cpcmd**, you should enter **cpcmd** and press return. Then, in the next line, enter the CP command:

```
  $cpcmd
  :spool cons *
```

This is necessary so that the AIX shell does not expand the *****.

Related Information

See the **cpcmd** special file in the *AIX Operating System Technical Reference*.

1.1.93 cpio

Purpose
Copies files into and out of archived storage and directories.

Syntax

```
               +-------+   +-------------+
cpio --- -o ---¦¦+---+ +---¦¦  one of    +---¦
               +-¦ a +-+   ¦ +---------+ ¦
                 ¦ c ¦     +-¦ B       +-+
                ¦¦ h ¦¦      ¦ Cvalue  ¦
                ¦¦ v ¦¦      +---------+
                ¦+---+¦
                +-----+

               +-------------+   +-------------+   +--- "*" ---+
cpio --- -i ---¦¦+---------+ +---¦¦  one of    +---¦           +---¦
               +-¦ b c d f +-+   ¦ +---------+ ¦   +- pattern -+
                 ¦ m r t u ¦¦    +-¦ B       +-+               ¦
                ¦¦ v s S 6 ¦¦      ¦ Cvalue  ¦      +---------+
                ¦+---------+¦      +---------+
                +-----------+


               +-------------+
cpio --- -p ---¦¦+---------+ +--- directory ---¦
               +-¦ a d h l +-+
                 ¦ m r u   ¦¦
                ¦+---------+¦
                +-----------+

-----------------
¦ Do not put a blank between these items.
```

Description

Warning: If you redirect the output from the **cpio** command to a special file (device), you should redirect it to the raw device and not the block device. Because writing to a block device is done asynchronously, there is no way to know if the end of the device has been reached.

Note: If you are using different machine types for the input and output of **cpio**, you must use the **-c** flag.

Subtopics

1.1.93.1 cpio -o

This command reads file path names from standard input and copies these files to standard output along with path names and status information. Path names cannot exceed 128 characters. Avoid giving the **cpio** command path names made up of many uniquely linked files because it may not have enough memory to keep track of them thereby losing linking information.

1.1.93.2 cpio -i

This command reads from standard input an archive file created by the **cpio -o** command and copies from the archive file all files with names that match **pattern**. These files are copied into the current directory tree. You may list more than one **pattern**, using the file name notation described under "sh, Rsh" in topic 1.1.420. However, in this application the special characters *****, **?**, and [...] match the / (slash) in path names, in addition to their use as described under "sh, Rsh" in topic 1.1.420. The default **pattern** is ***** (which specifies the selection of all files in the current directory).

In an expression such as **[a-z]**, the - minus means "through" according to the current collating sequence. A collating sequence may define **equivalence classes** for use in character ranges. See the "Introduction to International Character Support" in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, a collating sequence does not define equivalence classes for use in range expressions. Therefore, to avoid unpredictable results when using a range expression to match a class of characters, use a character class expression rather than a standard range expression. For more information about character class expressions, see the note on page 1.1.147.1.1 under the **ed, red** command.

Note: When the **cpio** files are written, the original values for the owner and group are set only when the superuser runs the **cpio** command. This restriction also applies for the **-p** option.

1.1.93.3 cpio -p

This command reads file path names from standard input and copies these files into the named *directory*. The specified directory must already exist. If these path names include directory names that do not already exist, you must use the **d** flag as well.

Note: When using the **cpio** command on internal tape backup units, use the **-B** option (5120-byte blocks) or **-C** option (specified with an even number of blocks). For example:

```
find . -print | cpio -ovB > /dev/rst0
```

The default 512-byte blocks are not a multiple of the PS/2 internal tape backup unit's block size and, therefore, cannot be used.

Flags

All flags must be listed together, without any blanks between them. Not all of the following flags can be used with each of the **-o**, **-i**, and **-p** flags.

a Resets access times of input files after they have been copied. This is only done if access permissions permit (see **utime** in *AIX Operating System Technical Reference*).

b Swaps both bytes and halfwords.

Note: If there are an odd number of bytes or halfwords in the file being processed, data can be lost.

B Performs block input/output, 5120 bytes to a record.

c Writes header information in ASCII character form. Use this flag for compatibility between machine types.

Cvalue Performs block input/output, **value** * 512 bytes to a record.

Note: The **C** flag and the **B** flag are mutually exclusive. If you list both, the **cpio** commands uses the last one it encounters in the flag list.

d Creates directories as needed.

f Copies all files except those matching **pattern**.

h Follows symbolic links. Normally, symbolic links are not followed. This option should be used only with **cpio -o** or **cpio -p** command.

l Links files rather than copies them, whenever possible. This flag is usable only with the **cpio -p** command.

m Retains previous file modification time. This flag does not work when copying directories.

r Renames files interactively. If you do not want to change the file name, enter the current file name or press the **Enter** key only. In this last case, the **cpio** command does not copy the file.

s Swaps bytes. This flag is usable only with the **cpio -i** command.

Note: If there are an odd number of bytes in the file being processed, data can be lost.

S Swaps halfwords. This flag is usable only with the **cpio -i** command.

Note: If there are an odd number of halfwords in the file being processed, data can be lost.

t Creates a table of contents. This command does not copy any files.

u Copies unconditionally. An older file now replaces a newer file with the same name.

v Lists file names. If you use this flag with the **t** flag, the output looks similar to that of the **ls -l** command.

6 Processes an old file (one written in UNIX Sixth Edition format). This flag is usable only with the **cpio -i** command.

Examples

1. To copy files onto diskette:

   ```
   cpio  -ov  <file names  >/dev/rfd0
   ```

   This command copies the files with path names that are listed in the file **file names** in a compact form onto the diskette (>**/dev/rfd0**). The **-v** flag causes the **cpio** command to display the name of each file as it is copied. This command is useful for making backup copies of files. The diskette must already be formatted, but it must not contain a file system or be mounted.

2. To copy files in the current directory onto diskette:

   ```
   ls  *.c  |  cpio  -ov  >/dev/rfd0
   ```

   This command copies all the files in the current directory whose names end with **.c** onto the diskette (>**/dev/rfd0**).

3. To copy the current directory and all subdirectories onto diskette:

   ```
   find  .  -print  |  cpio  -ov  >/dev/rfd0
   ```

   This command saves the directory tree that starts with the current directory (.) and includes all of its subdirectories and files. A faster way to do this is:

   ```
   find  .  -cpio  /dev/rfd0  -print
   ```

   The **-print** identify displays the name of each file as it is copied.

4. To list the files that have been saved onto a diskette with the **cpio** command:

```
cpio  -itv  </dev/rfd0
```

This command displays the table of contents of the data previously saved onto **/dev/rfd0** in **cpio** command format. The listing is similar to the long directory listing produced by the **li -l** command. To list only the file path names, use only the **-it** flags.

5. To copy the files previously saved with the **cpio** command from a diskette:

   ```
   cpio  -idmv  </dev/rfd0
   ```

   This command copies the files previously saved onto **/dev/rfd0** by the **cpio** command back into (**-i**) the file system. The **-d** flag allows the **cpio** command to create the appropriate directories if a directory tree was saved. The **-m** flag maintains the last modification time that was in effect when the files were saved. The **-v** flag causes the **cpio** command to display the name of each file as it is copied.

6. To copy selected files from diskette:

   ```
   cpio  -i  "*.c"  "*.o"  </dev/rfd0
   ```

   This command copies the files that end with **.c** or **.o** from diskette. The patterns **"*.c"** and **"*.o"** must be enclosed in quotation marks to prevent the shell from treating the ***** as a pattern-matching character. This is a special case in which the **cpio** command itself decodes the pattern-matching characters.

7. To rename files as they are copied from diskette:

   ```
   cpio  -ir  </dev/rfd0
   ```

   The **-r** flag causes the **cpio** command to ask you whether or not to rename each file before copying it from diskette. For example, the message:

   ```
   Rename  <prog.c>
   ```

   asks whether the file saved as **prog.c** should be given a new name as it is copied in. To rename the file, type the new name and press **Enter**. To keep the same name, you must enter the name again. To avoid copying the file at all, simply press the **Enter** key.

8. To copy a directory and all of its subdirectories:

   ```
   mkdir  /u/tom/newdir
   find  .  -print  |  cpio  -pdl  /u/tom/newdir
   ```

   This command duplicates the current directory tree, including the current directory and all of its subdirectories and files. The duplicate is placed in the new directory **/u/tom/newdir**. The **-l** flag causes the **cpio** command to link files instead of copying them, when possible.

Related Information

See the following commands: "ar" in topic 1.1.23, "find" in topic 1.1.165, and "ln" in topic 1.1.234.

See the **cpio** file format in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.94 cpp

Purpose
Performs file inclusion and macro substitution on a C language source file.

Syntax

```
            +-----------+   +---------------+   +--------------+   +----------
/lib/cpp ---¦ +-------+ +---¦  +--------+   +---+- -Dname   ¦ -+---¦
            +-¦ -C    +-+   +--¦ -Uname +---+   +- -Dname=def -+   +- infile ·
              ¦ -P    ¦¦       ¦ -Xdeps ¦ ¦
             ¦¦ -Idir ¦¦      ¦¦ -ttarg ¦ ¦
             ¦¦ -H    ¦¦      ¦+--------+ ¦
             ¦¦ -M    ¦¦      +-----------+
             ¦+-------+¦
             +---------+
```

¦ The default **def** is 1.

Description
The **cpp** program is the C-language preprocessor. It reads **infile** and writes to **outfile** (standard input and standard output, by default). Although you can use this preprocessor by itself, it is best to use it through the **cc** command, which by default sends a C-language source file to the **cpp** command as the first pass in compilation.

The **cpp** program recognizes two special names, **__LINE__** (the current line number) and **__FILE__** (the current file name). These names can be used anywhere just as any other defined name.

All **cpp** directive lines must begin with a # (number sign). These directives are:

#define name token-string
Replaces subsequent instances of **name** with **token-string**.

#define name(arg,...,arg) token-string
Replaces subsequent instances of the sequence **name(arg ,...,arg**) with **token-string**, where each occurrence of **arg** in **token-string** is replaced with the corresponding token in the comma-separated list. Note that there must not be any space between **name** and the left parenthesis.

#undef name
Ignores the definition of **name** from this point on.

#include "file"

#include <file>
Includes at this point the contents of **file**, which the **cpp** command then processes.

If you enclose **file** in double quotation marks (**" "**), the **cpp** command first searches first in

	the directory of **infile**, then in directories named with the **-I** flag, and finally in directories on a standard list. If you use the <**file**> notation, the **cpp** command searches for **file** only in the standard places and in any directory named with the **-I** flag. It does not search the directory in which **infile** resides.
#if expr	Places subsequent lines in the output only if **expr** evaluates to nonzero. All the binary non-assignment C operators, the **?:** operator, and the unary **-**, **!**, and ~ operators are legal in **expr**. The precedence of the operators is the same as that defined in the C Language. There is also a unary operator, **defined**, which can be used in **expr** in these two forms: **defined** (**name**) **defined name** This operator allows the **#ifdef** and **#ifndef** utilities in a **#if** directive. Only these operators, integer constants, and names which are known by the **cpp** command should be used in **expr**. See **cc.cfg** in the *AIX Technical Reference* for information on predefined C-processor symbols. The **sizeof** operator is not available.
#ifdef name	Places the subsequent lines in the output only if **name** has been defined by a previous **#define** and has not been undefined by an intervening **#undef**.
#ifndef name	Places the subsequent lines in the output only if **name** has not been defined by a previous **#define** directive or has been undefined by an intervening **#undef** directive.
#else	Places subsequent lines in the output only if the expression in the preceding **#if** directive evaluates to false (and hence the lines following the **#if** and preceding the **#else** directives have been ignored).
#elif expr	Places subsequent lines in the output only if the expression in the preceding **#if** directive evaluates to false, and the **expr** expression evaluates to nonzero.
#endif	Ends a section of lines begun by a test directive (**#if**, **#ifdef**, or **#ifndef**). Each test directive must have a matching **#endif** directive.
#line num ["file"]	Includes line control information for the next pass of the C compiler. The **num** is the line number of the next line and **file** is the file

from which it comes. If you omit "**file**", the current file name remains unchanged.

#	The null statement consists of a single # on a line of its own and performs no action.

You can nest the test directives and the possible **#else** directives.

Flags

-C Copies source file comments to the output file. If you omit this flag, the **cpp** command removes all comments (except those found on **cpp** directive lines).

-Dname[=def]
Defines **name** as in a **#define** directive. The default **def** is **1**.

-Idir Looks first in **dir**, then looks in the directories on the standard list for **#include** files with names that do not begin with a / (slash). See the previous discussion of the **#include** directive.

-M Generates Makefile dependencies and sends the result to standard error.

-P Preprocesses input without producing line control information for the next pass of the C compiler.

-Uname Removes any initial definition of **name**, where **name** is a reserved symbol predefined by the preprocessor.

-Xdeps Generates Makefile dependencies to file deps. **-X** does not suppress the generation of the expanded output for the next pass of the C compiler as does **-M**. Together **-X** and **-M** suppress expanded output and write the makefile dependencies to deps.

-ttarg Changes the default target name in the makefile dependency file to targ. See example 5.

Examples

1. To display the text that the preprocessor sends to the C compiler:

```
/lib/cpp  pgm.c
```

   This command preprocesses the file **pgm.c** and writes the resulting text to standard output. You may want to see the preprocessor output when looking for errors in your macro definitions.

2. To create a file containing more readable preprocessed text:

```
/lib/cpp  -P  -C  pgm.c  pgm.i
```

   This command preprocesses the file **pgm.c** and stores the result in the file **pgm.i**. It omits line numbering information intended for the C compiler (**-P**) and includes program comments (**-C**).

3. To predefine macro identifiers:

```
/lib/cpp  -DBUFFERSIZE=512  -DDEBUG  pgm.c  pgm.i
```

This command defines **BUFFERSIZE** with the value **512** and **DEBUG** with the value **1** before preprocessing.

4. To use **#include** files located in nonstandard directories:

   ```
   /lib/cpp  -I/u/tom/include  pgm.c
   ```

   This command looks in the current directory for quoted **#include** files, then in the directory **/u/tom/include**, and finally in the standard directories. It looks in the directory **/u/tom/include** for angle-bracketed **#include** files (< >) and then in the standard directories.

5. To generate makefile dependencies in file **depsfile**:

   ```
   /lib/cpp pgm.c -Xdepsfile
   ```

   file **depsfile** would contain dependencies in the form:

   ```
   pgm.o: /usr/include/stdio.h
   ```

   The **-t** option allows the user to change the target name (left side of the **:**), for example:

   ```
   /lbin/cpp pgm.c -Xdepsfile -tpgm
   ```

   would generate dependencies in the form:

   ```
   pgm: /usr/include/stdio.h
   ```

Files

/usr/include	Standard directory for **#include** files.

Related Information

See the following commands: "cc", "m4", and "ld".

1.1.95 craps

Purpose
Plays craps.

Syntax

/usr/games/craps ---¦

Description

The **craps** game plays a form of the game of craps that is played in Las Vegas. It simulates the ***roller*** while you place bets. Bet with the roller by making a positive bet or with the ***House*** by making a negative bet.

You start with a $2000 bankroll. When the program prompts with "bet?", you may bet all or part of your bankroll. If you bet more than your bankroll, the program repeats the prompt until you make a legal bet. Then the roller throws the dice. The payoff odds are one to one. The player wins depending on whether the bet is placed with the roller or with the House. The **first** roll is the roll immediately following a bet.

The following rules apply. On the first roll, 7 or 11 wins for the roller; 2, 3, or 12 wins for the House; and any other number becomes the ***point*** and you roll again (the next rule then applies). On subsequent rolls, the point wins for the roller; 7 wins for the House; and any other number rolls again.

If you lose your bankroll, the House prompts "marker?" offering to lend you an additional $2000. Accept the loan by responding "y" or "yes". Any other response ends the game. When you hold markers, the House reminds you before a bet how many markers are outstanding. When you have markers and your bankroll exceeds $2000, **craps** asks "Repay marker?" If you want to repay part or all of your loan, respond with "y" (or "yes"). If you have more than one marker, **craps** asks you "How many?" If you respond with a number greater than the number of markers you hold, it repeats the prompt until you enter a valid number. If you accumulate 10 markers (a total loan of $20,000), **craps** tells you so and exits. If you accumulate a bankroll of more than $50,000 while holding markers, the money owed is repaid automatically.

A bankroll of more than $100,000 breaks the bank, and **craps** will prompt "New game?" To quit the game, press INTERRUPT (**refer to keyboard definition**). The **craps** command displays whether you have won, lost, or broken even and exits.

1.1.96 crash

Purpose
Examines system images.

Syntax

```
        +- -d /dev/kem -+  +- -n /unix -----+  +------+
crash --¦               +--¦                +--¦      +--¦
        +- -d dump -----+  +- -n namelist --+  +- -a -+
```

Description
The **crash** command is an interactive utility for examining an operating system image (a core image or the running kernel). It has facilities for interpreting and formatting the various control structures in the system and certain miscellaneous functions useful for examining a dump.

When run without the **-d** flag, the **crash** command examines an active system. If you specify **dump**, the **crash** command assumes that it is a system dump file, and it sets the default process to the process running at the time of the crash. When specifying **namelist** file you must select the file from which the running system was booted if the **-d** flag is not specified. Otherwise, you must select the file from which the system was booted before the system dump was performed.

Notes:

1. While using the **crash** command, it may be helpful to have a source listing of the system header which identifies the flags the **crash** command uses.

2. Stack tracing of the current process on a running system does not work.

The **crash** command recognizes several aliases in the **format** specification accompanying the subcommands.

Format	**Aliases**	**Format**	**Aliases**
byte	b	inode	ino, i
character	char, c	longdec	ld, D
decimal	dec, e	longoct	lo, O
directory	direct, dir, d	octal	oct, o
hexadecimal	hexadec, hex, h, x	write	w

Subcommands

The **crash** command presents a prompt (**>**) when it is ready to interpret subcommands entered at the work station. The general subcommand format for **crash** is:

subcommand [**flags**] [**structures to be displayed**]

When allowed, **flags** modify the format of the data displayed. If you do not specify which structure elements you want to examine, all valid entries are displayed. In general, those subcommands that perform I/O with addresses assume hexadecimal notation.

Some of the subcommands recognized by the **crash** command have ***aliases*** (abbreviated forms that give the same result). If a subcommand has an alias, it is listed below. The **crash** command recognizes the following subcommands:

active [-]
Displays only active entries. When the - option is used, **active** displays all entries.

addr [table-index]
Alias: **a**. Displays the address of an item in an array (for example, the procedure table).

all
Displays all information available.

buf [list-of-buffer-headers]...
Alias: **b**. Displays the system buffer headers.

buffer [format][list-of-buffers]...
Aliases: **bufhdr**, **hdr**. Displays the data in a system buffer according to **format**. If you do not provide a **format** parameter, the previous **format** parameter you specified is used. Valid format parameters include **decimal, octal, hex, character, byte, directory, inode** and **write**. The **write** parameter creates a file in the current directory containing the buffer data.

bufhash [list-of-hash-chains]
Alias: **bufh**. Displays the buffer hash chains.

buflist [queue] Alias: **bufl**
Displays the buffer free list.

callout Aliases: **calls, call, c, timeout, time, tout**
Displays all entries in the callout table.

dcache Alias: **dca**
Displays information in the directory cache.

ds [list-of-data-addresses]...
Finds the data symbols closest to the given addresses.

dump [sym or addr] [cnt or -] [fmt or -] [proc-list]
Aliases: **od**, **hd**, **rd**. Dumps **cnt** data values starting at the symbol name **sym** or address **addr** given. Allowable format parameters for **fmt** are **octal**, **decimal**, **character**, **hex**, or **byte**. The default format is **hex**. The - flags for **cnt** and **fmt** are necessary only if you use **proc-list**.

file [-i islot][list-of-file-table-entries]... Aliases: **files, f**
Displays the file table. Unless specific file entries are requested, only those with a nonzero reference are displayed.

gensw
Displays information in the **gensw** structure.

header [-v]
Displays core header. When the **-v** option is used, **header** displays the contents of the dump directory.

help Alias: **?**
Displays a summary of **crash** commands.

inode [-l] [-f] [-n] [g gfs] [-i ino] [inode-slots]
Alias: **ino**, **i**. Displays the inode table. Unless specific **inode-slots** are requested, only those with a non-zero reference are displayed. The record lock structure is displayed when using the **-l** option. The **-n** option displays the NFS inodes only. The **-f** option displays the FIFO inodes only. When **g gfs** is specified, **inode** displays the inodes listed in **gfs**. When **-i ino** is specified, **inode** displays the all inode slots where the inode number equals **ino**.

malloc [-] [count[addr]] Alias: **mall**
Traces malloc chain starting at address **addr** showing **count** links in the chain. The - option shows the amount of free space available.

mount [-l] [-n] [list-of-gfs-numbers]
Alias: **mnt**, **m**. Displays the mount table. Unless the **list-of-gfs-numbers** option is used, mount displays only those mount tables that are in use. When the **-l** option is used, mount displays the local entries only. When the **-n** option is used, mount displays the NFS entries only.

netbuf [-a] [-l] [-v] [list-of-netmsg-slots]
Alias: **netb**, **netmsg**. Displays net message buffers. When the **-a** flag is used, **netbuf** displays the most common fields. When the **-l** flag is used, **netbuf** displays both the common and variant portions of **netbuf**. When the **-v** flag is used, **netbuf** displays only the variant fields.

netlist
Displays net message free list and nmheader.

netlog [-s siteno] [-{1,2,3} value] [n]
Displays the log of recent net messages. The **n** option indicates the number of entries to display (default is all). The **-s siteno** option restricts the net messages to the given site. The **-1**, **-2**, and **-3** flags are used to filter the net messages. **-1 value** restricts net messages to those whose long field matches **value**. **-2 value** restricts net messages to those whose short field matches **value**. **-3 value** restricts net messages to those whose long2 field matches **value**.

netswitch Alias: **netsw**
Displays information in the net switch structure.

nm [list-of-symbols]...
Displays symbol address as found in the **kernel-image** file.

osm [bytecnt] Aliases: **printf, printfs**
Displays the most recent messages printed on the console.

pcb
Displays information about the process control block.

plock [-tdu]
Lock crash into memory. With no options, text, data, and user structure are locked. The **-t** flag causes only text to be locked. The **-d** flag causes only data to be locked. The **-u** flag causes all locks to be released.

proc [-alp] [list-of-processes]
Alias: **p**. Displays the process table. (See the **/usr/include/sys/proc.h** file for this structure definition.) The **-l**

flag displays only runable processes. The **-p** flag displays **list-of-processes**, which is a list of PIDs. The **-a** flag displays all information to be displayed.

prop
Displays propagation list.

pvseg [-p] [-q] [list-of-processes]
Displays the vseg structure of processes. If no list is given, information about the current running process is displayed. If **-q** is specified, the vsegs for all processes are displayed. If **-p** is specified, the **list-of-processes** should be PIDs, not process slots.

quit Alias: **q**
Exits from the **crash** command.

rmsleep Alias: **rms**
Displays remote sleep table.

rmwakeup Alias: **rmw**
Displays remote wakeup table.

site [list-of-site-numbers]
Displays information in the site table and site data.

slot [list-of-addresses] Alias: **s**
Displays the name and slot for a given address.

sptab Alias: **sp**
Displays server process table information.

stack [process]... Alias **kernel, k**
Displays a dump of the kernel stack of a process. The addresses shown are virtual data addresses rather than true physical locations. If you do not specify an entry, information about the last running process is displayed. You can not trace the stack of the current process on a running system.

stat
Displays statistics found in the dump. These include the panic message (if a panic occurred), the time of the crash, and the system name.

tabgrow Alias: **systab**
Displays how many slots in all system tables are used and how many are available.

token [-] [list-of-token-slots] Alias: **tok**
Displays token control block table. The - flag provides additional information.

tokreq [list-of-file-numbers]
Displays token site request table information for the **list-of-site-numbers**. If no list is given, **tokreq** displays the table information for the site on which it was executed.

topology Alias: **top**
Displays the topology status.

trace [-p] [list-of-processes]

Alias: **t**. Displays a kernel stack trace of the current process. The trace starts at the bottom of the stack and attempts to find valid stack frames. The **-p** flag displays **list-of-processes**, which is a list of PIDs.

ts [list-of-text-addresses]...
Finds the text symbols closest to the given addresses.

tty [type] [-] [tty-entry]... Aliases: **term, dz, dh**
Displays the **tty** structures. The **type** parameter specifies which structure is used (such as **ksr**, or **rs**). The last **type** entered with the **tty** command becomes the default. The **-** (minus) flag displays the **stty** command parameters for the given line.

usage [command-list]
Displays the syntax for the given command.

user [-] [list-of-processes]... Aliases: **uarea, u_area, u**
Displays the user structure of the named process as determined by the information contained in the process table entry. (See the **/usr/include/sys/user.h** file for this structure definition.) If you do not specify the entry, the information about the last running process is displayed. Attempting to display a paged process produces an error message. If - is specified, **list-of-processes** should be a list of PIDs.

var Aliases: **tunables, tunable, tune, v**
Displays the tunable system parameters.

vseg
Displays kernel vseg structures.

!
Runs shell commands.

Flags

-d **dump**	Specifies the file containing the system image to be examined. The default file is **/dev/kmem**.
-n **namelist**	Specifies the file containing the kernel symbol definitions. The default file is **/unix**.
-a	Prints all useful information and then exits.

Files

/usr/include/sys/*.h	Header files for table and structure information.
/dev/kmem	Default system-image file.
/unix	Default name list file.
buf.#	Files containing buffer data.

Related Information
See the following commands: "mount" in topic 1.1.278, "nm" in topic 1.1.298, "ps" in topic 1.1.337, "sh, Rsh" in topic 1.1.420, and "stty, STTY" in topic 1.1.447.

1.1.97 cron

Purpose
Runs commands automatically.

Syntax

```
    ¦   +--------------+   +---------------+
cron ---¦              +---¦               +---¦
        +- -s logsize -+   +- -f openfreq -+
```

```
-----------------
¦ Not usually run from the command line, but included in /etc/rc.
```

Description
The **cron** command runs shell commands at specified dates and times. Regularly scheduled commands can be specified according to instructions contained in **crontab** files. You can submit your **crontab** file via the **crontab** command (see page 1.1.98). Use the **at** command (see page 1.1.26) to submit commands that are to be run only once. Because the **cron** command never exits, it should be run only once. This is best done by running the **cron** command from the initialization process through the **/etc/rc** command file (see page 1.1.354).

The **cron** command examines **crontab** files and **at** command files only during process initialization and when a file changes. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

The **cron** command also executes a **sync** system call approximately once a minute to ensure that all information in memory that should be on disk (buffered output) is written out. These periodic updates minimize the possibility of file system damage in the event of a crash. In addition, the **cron** command keeps a number of frequently used system directories open to keep their inodes in kernel memory for faster access.

The **cron** command creates a log of its activities in the file **/usr/adm/cron/log**.

For a discussion of how to schedule commands, see "crontab" in topic 1.1.98.

Note: To increase system performance, the **cron** command will open, and hold open, very frequently used system files and directories. This ensures that these frequently used files and directories are open when they are needed by other commands, and thus the other commands will often run faster. The system administrator can specify which files should be held open by putting a list of file names into the file **/etc/openfiles**. At periodic intervals (default is 20 minutes), **cron** will open the files named in this file. The default time interval may be changed with the **-f** option described below.

Flags

-s logsize Specifies the maximum size, in kilobytes, for the log file **/usr/adm/cron/log**. When **cron** is started, if **/usr/adm/cron/log** is more than **logsize** kilobytes in length, it is renamed to **/usr/adm/cron/log.old** and a new log file **/usr/adm/cron/log** is created.

-f openfreq	Specifies the number of minutes between readings of the file **/etc/openfiles**. Changes to this file will go undetected for up to **openfreq** minutes (default 20 minutes).

Files

/usr/adm/cron	Main cron directory.
/usr/adm/cron/log	Accounting information.
/usr/spool/cron	Spool area.
/etc/openfiles	List of files to be held open.
/usr/adm/cron/log.old	Saved accounting log.

Related Information

See the following commands: "at, batch" in topic 1.1.26, "crontab" in topic 1.1.98, and "rc" in topic 1.1.354.

See the **sync** system call file and the **openfiles** special file in *AIX Operating System Technical Reference*.

1.1.98 crontab

Purpose
Submits a schedule of commands to the **cron** command.

Syntax

```
           +------------+
crontab ---¦ +- file -+ +---¦
           +-¦ one of +-+
             ¦ +----+ ¦
             +-¦ -l +-+
               ¦ -r ¦
               +----+
```

Description
The **crontab** command copies the specified **file**, or standard input if you do not specify **file**, into a directory that holds all users' **crontab** files. The **cron** command runs commands according to the instructions in these **crontab** files. It then mails you the output from standard output and standard error for these commands, unless you redirect standard output or standard error. When **crontab** replaces a **crontab** file, the previous contents of the file are erased.

You may use the **crontab** command if your logname appears in the file **/usr/adm/cron/cron.allow**. If that file does not exist, the **crontab** command checks the file **/usr/adm/cron/cron.deny** to determine if you should be denied use of the command. If neither file exists, you can submit a job only if you are operating with superuser authority. The **allow/deny** files contain one user name per line.

Notes:

1. If your login ID is associated with more than one login name, the **crontab** command uses the first login name that appears in the **/etc/passwd** file, regardless of which login name you might actually be using.

2. If the **cron.allow** command exists, the superuser's logname must appear there for the superuser to be able to use the command.

3. Each user has a distinct **crontab** file on each site in a cluster. The **crontab** command only sets, lists, or removes the user's local **crontab** file. To access your **crontab** files on other cluster sites, use the **on** command to specify another cluster site.

4. The **crontab** file for the root user is used to run system management commands at regular intervals. This **crontab** files runs the three shell scripts **/usr/adm/daily**, **/usr/adm/weekly**, and **usr/adm/monthly**. These scripts can be customized for your system if you have superuser authority.

Each **crontab** file entry consists of a line with six fields, separated by spaces and tabs, that contain, respectively:

1. The minute (0-59)
2. The hour (0-23)
3. The day of the month (1-31)
4. The month of the year (1-12)

5. The day of the week (0-6 for Sunday-Saturday)
6. The shell command.

Each of these fields can contain:

A number in the specified rang
Two numbers separated by a minus to indicate an inclusive rang
A list of numbers separated by commas, which selects all numbers i the list
An asterisk, meaning all legal values

The specification of days may be made in two fields (day of the month and day of the week). If you specify both as a list of elements, both are adhered to. For example, the following entry:

```
0 0 1,15 * 1 command
```

runs **command** on the first and fifteenth days of each month, as well as every Monday. To specify days in only one field, the other field should contain an ***** (asterisk).

The **cron** command runs the command named in the sixth field at the selected date and time. If you include a **%** (percent sign) in the sixth field, the **cron** command treats everything that precedes it as the command invocation and makes all that follows it available to standard input, unless you escape or quote the percent sign (**\%** or **"%"**). Any additional % (percent sign) in the line are treated as newline in the standard input stream.

Note: The shell runs only the first line of the command field (up to a **%** or end of line). All other lines are made available to the command as standard input.

The **cron** command invokes a subshell from your **$HOME** directory. This means that it does not run your **.profile** file. If you schedule a command to run when you are not logged in and you want to have commands in your **.profile** file run, you must explicitly do so in the **crontab** file. (For a more detailed discussion of how **sh** can be invoked, see "sh, Rsh" in topic 1.1.420).

The **cron** command supplies a default environment for every shell, defining **HOME**, **LOGNAME**, **SHELL** (=**/bin/sh**), and **PATH** (=**:/bin:/usr/bin**).

Flags

-l	Lists your **crontab** file.
-r	Removes your **crontab** file from the **crontab** directory.

Examples

The following examples show valid **crontab** file entries.

1. To write the time to the console every hour on the hour:

   ```
   0 * * * * echo The hour is `date`. >/dev/console
   ```

   This example uses command substitution. For more information, see "Command Substitution" in topic 1.1.420.5.

2. To run the **calendar** command at 6:30 a.m. every day:

```
30 6 * * * /usr/bin/calendar -
```

3. To define text for the standard input to a command:

   ```
   0 16 10-31 12 5 /etc/wall%HAPPY HOLIDAYS!%Remember to turn in your time
   ```

   This command writes a message to all users logged in at 4:00 p.m. every day between December 10 and 31.

   The text following the **%** (percent sign) defines the standard input to the **wall** command as:

   ```
   HAPPY HOLIDAYS!
   Remember to turn in your time card.
   ```

Files

/usr/adm/cron	Main **cron** command directory.
/usr/spool/cron/crontabs	Spool area.
/usr/adm/cron/cron.allow	List of allowed users.
/usr/adm/cron/cron.deny	List of denied users.

Related Information

See the following commands: "cron" in topic 1.1.97 and "sh, Rsh" in topic 1.1.420.

1.1.99 crypt

Purpose

crypt - encodes/decodes data

Syntax

```
         +------------+
crypt ---¦            +---¦
         +- password -+
```

Description

The **crypt** command encodes and decodes data that you supply. You can use **crypt** to keep others from reading or using sensitive data. The encoded data appears to be gibberish. ASCII and binary files may be encoded and decoded with **crypt**.

The password "key" used when encoding the data must be supplied to decode the data. Be sure to remember the password, and keep it secret.

By default, **crypt** reads from the standard input and writes on the standard output. You must use redirection if you want to read and write files. For example, to create the encoded file **secrets** from the existing ASCII file **mydiary** with the password **rosebud**, type:

```
  crypt rosebud < mydiary > secrets
```

Note that the file **mydiary** will be unchanged, and thus potentially readable by others. So, usually you will want to delete the file **mydiary**. To view the contents of the encoded file **secrets**, type:

```
  crypt rosebud < secrets
```

The contents will print to standard output, usually the screen. To create a decoded version of **secrets**, type:

```
  crypt rosebud < secrets > journal
```

The decoded file **journal** will be written to the current directory. The file **secrets** will remain encoded.

If you don's supply **crypt** with a password, **crypt** will prompt you for one with: **Enter key:**. Here, **key** means **password** - you may enter more than one character. Press return to complete password entry. The password will not be echoed to the screen while you type.

Files encrypted by **crypt** are compatible with those created by the editor **ed** in encryption mode.

The security of encrypted files depends on three factors: the fundamental method must be hard to solve; direct search of the key space must be infeasible; 'sneak paths' by which keys or cleartext can become visible must be minimized.

The **crypt** command implements a one-rotor machine designed along the lines of the German Enigma, but with a 256-element rotor. Methods of attack on such machines are known, but not widely; moreover the amount of work required is likely to be large.

The transformation of a key into the internal settings of the machine is deliberately designed to be expensive, for example, to take a substantial fraction of a second to compute. However, if keys are restricted to (say) three lower-case letters, then encrypted files can be read by expending only a substantial fraction of five minutes of machine time.

Since the key is an argument to the **crypt** command, it is potentially visible to users executing **ps** or a derivative. To minimize this possibility, **crypt** takes care to destroy any record of the key immediately upon entry. No doubt the choice of keys and key security are the most vulnerable aspect of **crypt**.

Files

/dev/tty for typed key

Related Information

See the following commands: "ed, red" in topic 1.1.147 and "makekey" in topic 1.1.256.

1.1.100 csh

Purpose
Interprets commands read from a file or entered from the keyboard.

Syntax

```
                                                 +-------------------------+
       +--------+   +--------+   +-----------+   ¦         one of          ¦
csh ---¦ one of +---¦ one of +---¦           +---¦   +-----------------+   +--
       ¦ +----+ ¦   ¦ +----+ ¦   ¦ +-------+ ¦   ¦   ¦ -c"cmd string"  ¦   ¦
       +-¦ -v +-+   +-¦ -x +-+   +-¦ -b    +-+   ¦ +-¦ -s              +--+¦
         ¦ -V ¦       ¦ -X ¦       ¦ -e -f ¦     +-¦ ¦ -t              ¦  +-+
         +----+       +----+      ¦¦ -i -n ¦¦      ¦ +-----------------+  ¦
                                  ¦+-------+¦      ¦           +-------+  ¦
                                  +---------+      +--- file --¦       +-+
                                                               +- arg -+
                                                                       ¦
                                                                +------+
```

Description
The **csh** command is a command language interpreter incorporating a history mechanism (see "History Substitutions" in topic 1.1.100.1), job control facilities (see "Jobs"), interactive file name and user name completion (see "File Name Completion"), and a C-like syntax. To use the **csh** job control facilities, you must also use the tty driver described in the **termino** file (see the **termino** special file in *AIX Operating System Technical Reference*). The tty driver enables generation of interrupt characters from the keyboard to stop jobs. See "stty, STTY" in topic 1.1.447 for details on setting options in the tty driver.

The **csh** command begins by executing commands from the **.cshrc** file in the home directory of the invoker. If this is a login shell, **csh** also executes commands from the **.login** file in the invoker's **home** directory. Typically, users put the command **stty crt** in their **.login** file and also set their terminal type for full-screen commands such as **vi**.

The user may want certain commands to be run only from interactive shells. To accomplish this, add instructions to the **.cshrc** file, using the following model:

```
  if ($?prompt) then
     # Do interactive-only things
  endif
```

Normally, the shell then begins reading commands from the terminal, prompting with **%**. If the user redefines the prompt, it should be done after the test above.

The shell then repeatedly performs the following actions: a line of command input is read and broken into words. This sequence of words is placed on the command history list and then is parsed. Finally, each command in the current line is executed.

When a login shell terminates, it executes commands from the **.logout** file in the user's home directory.

Lexical Structure

The **csh** command interprets commands according to the following lexical structure:

The **csh** shell splits input lines into words at blanks and tabs. (Japanese characters must be separated by spaces if they are to be treated as separate words.) The characters **& | ; < > (**) form separate words. The **&&**, **||**, **<<**, or **>>** terms also form single words. You can make these parser metacharacters part of other words or prevent the metacharacters from having their special meaning by preceding them with ****. A new line preceded by a **** is equivalent to a blank.

Strings enclosed in matched pairs of quotations (' ` or ") form parts of a word. In these strings, metacharacters (including blanks and tabs) do not form separate words. (The semantics of quotations are described in "Quotations with Single and Double Quotes" in topic 1.1.100.2.) Within pairs of ' or " characters, a new line preceded by a \ gives a true new-line character.

When the shell's input is not a terminal, the character # introduces a comment, which continues until the end of the input line. The # character is prevented from having this special meaning when it is preceded by **** or is in quotations (`, ', or ").

Commands

In the **csh** interpreter, a simple command is a sequence of words, the first of which specifies the command to be executed. A simple command or a sequence of simple commands separated by **|** characters forms a pipeline. The output of each command in a pipeline is connected to the input of the next. Sequences of pipelines can be separated by semicolons (**;**) and then are executed sequentially. When a sequence of pipelines is followed by an **&**, it is executed without waiting for the preceding pipeline to terminate.

Commands and pipelines can be placed in parentheses **( )** to a form simple commands, which, in turn, can be used as components of a pipeline. You also can separate pipelines with **||** or **&&**, indicating, as in C language, that the second is to be executed only if the first fails or succeeds, respectively (see "Expressions" in topic 1.1.100.5).

Jobs

The **csh** shell associates a job with each pipeline. The shell keeps a table of current jobs, printed by the **jobs** command (see page 1.1.100.5 for information on the **jobs** built-in command), and assigns them small integer numbers. When a job is started asynchronously with **&**, the shell prints a line which looks like this:

```
  [1] 1234
```

This line indicates that the job which was started asynchronously was job number 1 and had one top-level process with an ID of 1234.

If you are running a job and wish to do something else, you can press **Ctrl-Z** to send a stop signal to the current job. The shell then normally indicates the job has been stopped and prints another prompt. You then can manipulate the state of this job, putting it in the background with the **bg** command (see page 1.1.100.5 for information on the **bg** built-in command), running some other commands, and eventually bringing the job back into the foreground with the **fg** command (see page 1.1.100.5 for information on the **fg** command). **Ctrl-Z** takes effect immediately. Like an interrupt, **Ctrl-Z** causes pending output and unread input to be

A job running in the background stops if it tries to read from the terminal. Background jobs are normally allowed to produce output, but this can be disabled by the command **stty tostop**. If you set this tty option, background jobs stop when they try to produce output as well as when they try to read input.

There are several ways to refer to jobs in the **csh** shell. The character **%** introduces a job name. For example, if you wish to refer to job number 1, you can name it **%1**. Naming a job brings it to the foreground; that is, **%1** is a synonym for **fg %1**. Similarly, **%1 &** puts job 1 in the background. Jobs also can be named by prefixes of the string that would start them, if these prefixes are unambiguous. For example, **%ex** restarts a suspended **ex** job if there is only one suspended job whose name begins with the string **ex**. You can also use **%?string** to specify a job whose text contains **string** if there is only one such job.

The shell maintains a record of the current and previous jobs. In output pertaining to jobs, the current job is marked with a **+** and the previous job with a **-**. The abbreviation **%+** refers to the current job and **%-** refers to the previous job. For close analogy with the syntax of the history mechanism (see "History Substitutions" in topic 1.1.100.1), **%%** is also a synonym for the current job.

Status Reporting

The **csh** shell learns immediately whenever a process changes state. If a job becomes blocked so that no further progress is possible, the shell informs you, but not until just before it prints a prompt. However, if you set the shell variable **notify** (see page 1.1.100.5 for information on the **notify** variable), the shell notifies you immediately of changes of status in background jobs. There is also a **notify** shell command (see page 1.1.100.5) that marks a single process so that its status changes are immediately reported. By default, **notify** marks the current process; simply enter **notify** after starting a background job to mark it.

When you try to leave the shell while jobs are stopped, you are warned that you have stopped jobs. You can use the **jobs** command to see what they are. If you use the **jobs** command or immediately try to exit again, the shell does not warn you a second time, and the suspended jobs are terminated.

File Name Completion

When the file name completion feature of the **csh** command is enabled by setting the shell variable **filec**, the **csh** command interactively completes file names and user names from unique prefixes input from the terminal and followed by the escape character (the **Esc** key or **Ctrl-[**). For example, if the current directory looks like this:

```
  DSC.OLD    bin         cmd       lib     xmpl.c
  DSC.NEW    chaosnet    cmtest    mail    xmpl.o
  bench      class       dev       mbox    xmpl.out
```

and the input is:

```
% vi ch<Esc>
```

the **csh** command completes the prefix **ch** to the only matching file name,

chaosnet, changing the input line to:

```
% vi chaosnet
```

However, given a prefix that matches more than one file name:

```
% vi D<Esc>
```

the **csh** expands the input to:

```
% vi DSC.
```

and sounds the terminal bell to indicate the expansion is incomplete since there are two file names beginning with **D**.

If, instead of completing a name, you enter a partial file name followed by the end-of-file character (usually **Ctrl-D**), the **csh** command lists all file names matching the prefix. For example, the input:

```
% vi D<Ctrl-D>
```

causes all files beginning with D to be listed:

```
DSC.NEW    DSC.OLD
```

while the input line remains unchanged.

The same sequence of escape and end-of-file characters can also be used to expand partial user names if the word to be completed begins with the character **~**. For example, entering:

```
cd ~ro<Ctrl-D>
```

produces the expansion:

```
cd ~root
```

The use of the terminal bell to signal errors or multiple matches can be inhibited by setting the variable **nobeep**.

Normally, all files in the particular directory are candidates for name completion. Files with certain suffixes can be excluded from consideration by setting the variable **fignore** to the list of suffixes to be ignored. Thus, if **fignore** is set by the command:

```
% set fignore = (.o .out)
```

then entering:

```
% vi x<Esc>
```

results in the completion to

```
% vi xmpl.c
```

ignoring the files **xmpl.o** and **xmpl.out**. However, if the only possible completions involve files named with one of these suffixes, **.o** and **.out** are not ignored. (Note: The **fignore** variable does not affect usage of **Ctrl-D**. If you begin a partial file name and then press **Ctrl-D**, all files, regardless of suffix, are listed.)

The use of file completion requires the terminal setting of **ctlecho** (see page 1.1.447.9 for information on **ctlecho**). Also, the use of **Esc** for file name completion conflicts with the use of **Esc** for other purposes. Therefore, these two functions cannot be used at the same time.

Substitutions

The following sections describe the various transformations the **csh** shell performs on the input in the order in which they occur: history, alias, variable, command, and file-name substitutions.

Subtopics

1.1.100.1 History Substitutions

History substitutions place words from previous command input as portions of new commands, making it easy to repeat commands, repeat arguments of a previous command in the current command, or fix spelling mistakes in the previous command. History substitutions begin with the character **!** and can begin anywhere in the input stream as long as they are not nested. The **!** can be preceded by a \ to negate its special meaning; for convenience, a **!** is passed unchanged when it is followed by a blank, tab, new-line character, **=**, or **(**. (History substitutions also occur when an input line begins with **^**.) Any input line which contains history substitution is echoed on the terminal before it is executed as it would have been typed without history substitution.

Commands input from the terminal are saved on the history list. The history substitutions reintroduce sequences of words from these saved commands into the input stream. The size of the history list is controlled by the **history** variable (see 1.1.100.5 for information on the history variable). The previous command is always retained, regardless of its value. Commands are numbered sequentially from 1.

For example, consider the following output from the **history** command:

```
 9  write michael
10  ex write.c
11  cat oldwrite.c
12  diff *write.c
```

The commands are shown with their event numbers. The current event number can be made part of the prompt by placing the **set prompt=\!** command in the **.schrc** file, or by entering the **set prompt** command at the command line.

If the current event is 13, we can refer to previous events in several ways: by event number (for example, **!11**); relatively (for example, **!-2**, referring to event 11); by a prefix of a command word (for example, **!d** for event 12 or **!wri** for event 9); or by a string contained in a word in the command (for example, **!?mic?**, referring to event 9). Without further modification, these forms simply reintroduce the words of the specified events, each separated by a single blank. As a special case, **!!** refers to the previous command.

To select words from an event, we can follow the event specification by a colon (**:**) and a designator for the desired words. The words of an input line are numbered from 0, the first (and usually command) word being 0, the second word (first argument) being 1, and so on. The basic word designators are:

0	First (command) word.
n	**n**th argument.
^	First argument (for example, **1**).
$	Last argument.
%	Word matched by (immediately preceding) ?**s**? search.
x-y	Range of words.
-y	Abbreviation for **0-y**.
*****	Abbreviation for **^ - $**, or nothing if only one word in event.
x*	Abbreviation for **x - $**.
x-	Like **x***, but omitting word **$**.

The **:** (colon) separating the event specification from the word designator can be omitted if the argument selector begins with a **^**, **$**, *****, **-**, or **%**.

After the optional word designator, a sequence of modifiers can be placed, each preceded by a colon. The following modifiers are defined:

h	Removes a trailing path name component, leaving the head.
r	Removes a trailing **.xxx** component, leaving the root name.
e	Removes all but the extension part. (**yyy.xxx** becomes **xxx**.)
s/l/r/	Substitutes **l** for **r**.
t	Removes all leading path name components, leaving the tail.
&	Repeats the previous substitution.
g	Applies the change globally, prefixing the above, for example, **g&**.
p	Prints the new command but does not execute it.
q	Quotes the substituted words, preventing further substitutions.
x	Like **q**, but breaks into words at blanks, tabs, and new-line characters.

Unless preceded by a **g**, the modification is applied only to the first modifiable word. Using a substitution that does not apply to any word is considered an error.

The left-hand side of substitutions are not regular expressions as editors would treat them, but instead are strings. Any character can be used as the delimiter in place of /; a **** quotes the delimiter into the **l** and **r** strings. The character **&** in the right-hand side is replaced by the text from the left. A **** quotes **&** also. A null **l** uses the previous string either from a **l** or from a contextual scan string **s** in **!?s?**. The trailing delimiter in the substitution can be omitted if a new line follows immediately, as can the trailing **?** in a contextual scan.

A history reference can be given without an event specification. For example, in the case of **!$**, the reference is to the previous command unless a previous history reference occurred on the same line, in which case this form repeats the previous reference. Thus, **!?foo?^ !$** gives the first and last arguments from the command matching **?foo?**.

A special abbreviation of a history reference occurs when the first non-blank character of an input line is a **^**. This is equivalent to **!:s^**, providing a convenient shorthand for substitutions on the text of the previous line. Thus **^lb^lib** fixes the spelling of **lib** in the previous command.

Finally, a history substitution can be surrounded with **{** and **}** if necessary to insulate it from the characters which follow. Thus, after **ls -ld ~paul** we could enter **!{l}a** to do **ls -ld ~paula**, while **!la** would look for a command starting **la**.

1.1.100.2 Quotations with Single and Double Quotes

The quotation of strings by ' and " can be used to prevent all or some of the remaining substitutions. Strings enclosed in ' are prevented from any further interpretation. Strings enclosed in " can be expanded as described below. In both cases, the resulting text becomes all or part of a single word. Only in one special case (see "Command Substitution" in topic 1.1.100.5) does a string quoted in " yield parts of more than one word; strings quoted in ' never do.

1.1.100.3 Alias Substitutions

The **csh** shell maintains a list of aliases which can be established, displayed, and modified by the **alias** and **unalias** commands (see pages 1.1.100.5 and 1.1.100.5 for information about these built-in commands). After a command line is scanned, it is parsed into distinct commands, and the first word of each command, left to right, is checked to see if it has an alias. If it does, the text which is the alias for that command is reread with the history mechanism as though that command were the previous input line. The resulting words replace the command and argument list. If no reference is made to the history list, the argument list is left unchanged.

For example, if the alias for **ls** is **ls -l**, the command **ls /usr** would map to **ls -l /usr**. Similarly, if the alias for **lookup** was **grep !^ /etc/passwd**, **lookup bill** would map to **grep bill /etc/passwd**.

If an alias is found, the word transformation of the input text is performed, and the aliasing process begins again on the reformed input line. Looping is prevented if the first word of the new text is the same as the old by flagging it to prevent further aliasing. Other loops are detected and cause an error.

The alias substitution mechanism allows aliases to introduce parser metasyntax. For example, the command **alias p 'pr \!* | print'** creates an alias which applies the **pr** command to its arguments.

1.1.100.4 Variable Substitution

The **csh** shell maintains a set of variables, each of which has as its value a list of zero or more words. Some of these variables are set by the shell or referred to by it. For instance, the **argv** variable is an image of the shell's argument list, and words of this variable's value are referred to in special ways. No multibyte characters are allowed in variable names. However, the contents of the variables can contain multibyte characters.

The values of variables can be displayed and changed by using the **set** and **unset** commands (see page 1.1.100.5 for more information on these built-in commands). Of the variables referred to by the shell, many are toggles. For instance, the verbose variable is a toggle which causes command input to be echoed. The setting of this variable results from the **-v** command line option.

Other operations treat variables numerically. The @ command (see page 1.1.100.5 for more information on the @ built-in command) permits numeric calculations to be performed and the result assigned to a variable. Variable values are, however, always represented as strings. For numeric operations, the null string is considered to be zero, and the second and subsequent words of multi-word values are ignored.

After the input line is aliased and parsed, and before each command is executed, variable substitution is performed with **$** characters. This expansion can be prevented by preceding the **$** with a backslash (\) except within double quotes ("), where it always occurs, and within single quotes ('), where it never occurs. Strings quoted by single back quotes (`) are interpreted later (see "Command Substitution" in topic 1.1.100.5), so **$** substitution does not occur there until later, if at all. A **$** is passed unchanged if followed by a blank, tab, or end-of-line character.

Input and output redirections are recognized before variable expansion and are variable-expanded separately. Otherwise, the command name and entire argument list are expanded together. Thus, it is possible for the first command word to generate more than one word, the first of which becomes the command name, and the rest of which become arguments.

Unless enclosed in **"** or given the **:q** modifier, the results of variable substitution can eventually be command and file-name substituted. Within **"**, a variable whose value consists of multiple words expands to a portion of a single word, with the words of the variables value separated by blanks. When the **:q** modifier is applied to a substitution, the variable expands to multiple words, with each word separated by a blank and quoted to prevent later command or file name substitution.

The following meta-sequences are provided for introducing variable values into the shell input. Unless otherwise noted, referencing an unset variable is an error.

$name
${name}
Replaced by the words of the value of the variable specified by **name**, each word separated by a blank. Braces separate **name** from characters which otherwise would be part of it. Shell variables can have names consisting of up to 20 letters and numbers, starting with a letter. The underscore character is considered a letter.

If **name** is not a shell variable but is set in the environment, that value is returned; however, the **:** modifiers (see page 1.1.100.1 for more information on the **:** modifiers) are not available in this case.

$name[selector]
${name[selector]}
Selects words from the value of **name**. The selector is subjected to **$** substitution and can consist of a single number or two numbers separated by a **-**. The first word of a variable's value is numbered 1. If the first number of a range is omitted, it defaults to 1. If the last member of a range is omitted, it defaults to **$#name**. The selector ***** selects all words. A range can be empty if the second argument is omitted or is in the range.

$#name
${#name}
Specifies the number of words in the variable; useful for later use in a **selector**.

$0
Substitutes the name of the file from which command input is being read.

$number
${number}
Equivalent to **$argv[number**] (see page 1.1.100.5 for information on the **argv** variable).

$*
Equivalent to **$argv[***] (see page 1.1.100.5).

The modifiers **:h**, **:t**, **:r**, **:q**, and **:x** can be applied to the substitutions above as can the combinations **:gh**, **:gt**, and **:gr**. If braces **{ }** appear in the command form, the modifiers must appear within the braces. Only one **:** modifier on each $ expansion is allowed. The following substitutions cannot be modified with **:** modifiers.

$?name
${?name}
Substitutes the string **1** if **name** is set and **0** if it is not.

$?0
Substitutes **1** if the current input file name is known and **0** if it is not known.

$$
Substitutes the process number (decimal) of the parent shell.

$<
Substitutes a line from the standard input with no further interpretation; used to read from the keyboard in a shell script.

1.1.100.5 Command and File-Name Substitution

Command and file-name substitution are applied selectively to the arguments of built-in commands. Portions of expressions which are not evaluated are not subjected to these expansions. For commands which are not internal to the shell, the command name is substituted separately from the argument list. This occurs in a child of the main shell after input and output redirection is performed.

Command Substitution

Command substitution is indicated by a command enclosed in '. The output from such a command is normally broken into separate words at blanks, tabs, and new lines, with null words being discarded. The resulting text then replaces the original string. Within ", only new lines force new words; blanks and tabs are preserved. In any case, the single final new line does not force a new word. Consequently, a command substitution can yield only part of a word even if the command outputs a complete line.

File-Name Substitution

If a word contains any of the characters *****, **?**, **[**, or **{**, or begins with the character **~**, that word is a candidate for file-name substitution. This word is then regarded as a pattern and replaced with an alphabetically sorted list of file names that match the pattern. The current collating sequence is used, which can be specified by the environment variables **LC_COLLATE** or **LANG**. In a list of words specifying file-name substitution, not every pattern has to match an existing file name; however, at least one pattern must match. Only the metacharacters *****, **?**, and **[** imply pattern matching.

In matching file names, the character **.** (period) at the beginning of a file name or immediately following a **/**, as well as the character **/**, must be matched explicitly. The character ***** matches any string of characters, including the null string. The character **?** matches any single character. The sequence **[...]** matches any one of the characters enclosed. Within **[...]**, a pair of characters separated by **-** matches any character lexically between the two. The characters that match this pattern are defined by the current collating sequence (see "ctab" in topic 1.1.103).

The character **~** at the beginning of a filename refers to home directories. Standing alone, the **~** character expands to the invoker's home directory as reflected in the value of the variable $HOME. When **~** is followed by a name consisting of letters, digits, and **-** characters, the shell searches for a user with that name and substitutes the user's home directory. Thus, **~ken** expands to **/usr/ken** and **~ken/chmach** to **/usr/ken/chmach**. If the character **~** is followed by a character other than a letter or **/**, or if it does not appear at the beginning of a word, it is left undisturbed.

For example, the meta-notation **a{b,c,d}e** is a shorthand for **abe ace ade**. Left-to-right order is preserved, with results of matches being sorted separately at a low level to preserve this order. File name substitution can be nested. For example, **~source/s1/{oldls,ls}.c** expands to **/usr/source/s1/oldls.c /usr/source/s1/ls.c** if the home directory for **source** is **/usr/source**. Similarly, **../{memo,*box}** expands to **../memo ../box ../mbox**. (Note that **memo** is not sorted with the results of matching ***box**.) As a special case, the characters **{**, **}**, and **{}** are passed unexpanded.

Input/Output

The standard input and standard output of a command can be redirected with the following syntax:

< name
Opens file **name** (which is first variable, command, and file-name expanded), as the standard input.

<< word
Reads the shell input up to a line which is identical to **word**. The **word** is not subjected to variable, file-name, or command substitution. Each input line is compared to **word** before any substitutions are done on this input line. Unless a quoting \, **"**, **'**, or **`** appears in **word**, variable and command substitutions are performed on the intervening lines, allowing **\\** to quote **$**, **\\** , and **`**. Commands which are substituted have all blanks, tabs, and new lines preserved, except for the final new line, which is dropped. The resultant text is placed in an anonymous temporary file, which is given to the command as standard input.

> name
>! name
>& name
>&! name
The file **name** is used as standard output. If the file does not exist, it is created; if the file exists, it is truncated and its previous contents are lost.

If the variable **noclobber** is set, the file must not exist or be a character special file (for example, a terminal or **/dev/null**), which helps to prevent accidental destruction of files. The **!** forms can be used to suppress this check.

The forms involving **&** route the diagnostic output into the specified file as well as the standard output. The **name** specifier is expanded in the same way as **<** input file names are.

>> name
>>& name
>>! name
>>&! name
Uses file **name** as standard output like **>** but places output at the end of the file. If the variable **noclobber** is set, it is an error for the file not to exist, unless one of the **!** forms is given. Otherwise similar to **>**.

A command executes in the environment in which the shell was invoked as modified by the input-output parameters and the presence of the command in a pipeline. Thus, unlike some previous shells, commands run from a file of shell commands have no access to the text of the commands by default; rather, they receive the original standard input of the shell. The **<<** mechanism should be used to present inline data. This permits shell command scripts to function as components of pipelines and allows the shell to block read its input. Note that the default standard input for a command run detached is **not** modified to be the empty file **/dev/null**; rather, the standard input remains as the original standard input of the shell. If this is a terminal and if the process attempts to read from the terminal, the process is blocked and the user is notified.

Diagnostic output can be directed through a pipe with the standard output.

Simply use the form **|&** rather than just **|**.

Expressions

A number of the built-in commands (to be described later) take expressions, in which the operators are similar to those of C, with the same precedence. These expressions appear in the @, **exit**, **if**, and **while** commands. The following operators are available:

```
       ||  &&  |  ^  &  ==  !=  =~  !~
       <=  >=  <  >  <<  >>  +  -  *  /  %  !  ~  (  )
```

Here the precedence increases to the right, **== != =~** and **!~**, **<= >= <** and **>**, **<<** and **>>**, **+** and **-**, *** /** and **%** being, in groups, at the same level. The **==**, **!=**, **=~**, and **!~** operators compare their arguments as strings; all others operate on numbers. The operators **=~** and **!~** are like **!=** and **==** except that the right-hand side is a pattern (containing, for example, *****, **?**, and instances of **[...]**) against which the left-hand operand is matched. This reduces the need for use of the **switch** statement in shell scripts when all that is really needed is pattern matching.

Strings that begin with **0** are considered octal numbers. Null or missing arguments are considered **0**. The result of all expressions are strings, which represent decimal numbers. No two components of an expression can appear in the same word; they should be surrounded by spaces except when adjacent to components of expressions which are syntactically significant to the parser (**& | < > ( )**).

Also available in expressions as primitive operands are command executions enclosed in **{** and **}** and file inquiries of the form **-l file** where **l** is one of:

r	The **file** exists and permits read access.
w	The **file** exists and permits write access.
x	The **file** exists and permits execute access.
e	The **file** exists.
o	The **file** exists and is owned by this user.
z	The **file** exists and has zero size.
f	The **file** exists and is a regular file.
d	The **file** exists and is a directory.

The specified **file** is command and file-name expanded and then tested to see if it has the specified relationship to the real user. If the file does not exist or is inaccessible, all inquiries return false (that is, 0). Command executions succeed, returning true (that is, 1) if the command exits with status 0; otherwise, they fail, returning false (0). If more detailed status information is required, the command should be executed outside of an expression and the variable status examined.

The following example checks for the existence of the file **/u/janie/core**. If the file is found, the **echo** command runs.

```
  if ( -e /u/janie/core ) then
         echo "core file found."
  fi
```

Control Flow

The **csh** shell contains a number of commands which can be used to regulate the flow of control in command files (shell scripts) and (in limited but useful ways) from terminal input. These commands all operate by forcing the shell to reread or skip in its input and, because of the implementation, restrict the placement of some of the commands.

The **foreach**, **switch**, and **while** statements, as well as the **if-then-else** form of the **if** statement require that the major keywords appear in a single simple command on an input line as shown below.

If the shell's input is not seekable, the shell buffers up input whenever a loop is being read and performs seeks in this internal buffer to accomplish the rereading implied by the loop. (To the extent that this allows, backward **gotos** will succeed on nonseekable inputs.)

Built-in Commands

Built-in commands are executed within the **csh** shell. If a built-in command occurs as any component of a pipeline except the last, it is executed in a subshell.

alloc
Shows the amount of dynamic memory acquired, broken down into used and free memory. With an argument, shows the number of free blocks in each size category. The categories start at size 8 and double at each step. For example:

```
alloc
alloc "64"
```

alias
alias name
alias name wordlist
The first form prints all aliases. The second form prints the alias for **name**. The final form assigns the specified **wordlist** as the alias of **name**; **wordlist** is command and file-name substituted. **name** is not allowed to be **alias** or **unalias**.

bg
bg %job...
Puts the current or specified jobs into the background, continuing them if they were stopped.

break
Causes execution to resume after the **end** of the nearest enclosing **foreach** or **while**. The remaining commands on the current line are executed. Multi-level breaks are thus possible by writing them all on one line.

breaksw
Causes a break from a **switch**, resuming after the **endsw**.

case label:
A label in a **switch** statement as discussed below.

cd
cd name
chdir
chdir name
Changes the shell's working directory to directory **name**. If no argument is given, changes to the home directory of the user.

If **name** is not found as a subdirectory of the current directory (and does not begin with **/**, **./**, or **../**), each component of the variable **cdpath** is checked to see if it has a subdirectory **name**. Finally, if all else fails but **name** is a shell variable whose value begins with **/**, this is tried to see if it is a directory.

continue
Continues execution of the nearest enclosing **while** or **foreach**. The rest of the commands on the current line are executed.

default:
Labels the default case in a **switch** statement. The default should come after all **case** labels.

dirs
Prints the directory stack; the top of the stack is at the left, the first directory in the stack being the current directory.

echo wordlist
echo -n wordlist
The specified words are written to the shell's standard output, separated by spaces, and terminated with a new-line character unless the **-n** option is specified.

else
end
endif
endsw
See the description of the **foreach**, **if**, **switch**, and **while** statements below.

eval arg...
The arguments are read as input to the shell and the resulting command(s) executed in the context of the current shell. This is usually used to execute commands generated as the result of command or variable substitution, since parsing occurs before these substitutions.

exec command
The specified **command** is executed in place of the current shell.

exit
exit (expr)
The shell exits either with the value of the **status** variable (first form) or with the value of the specified **expr** (second form).

fg
fg %job...
Brings the current or specified jobs into the foreground, continuing them if they were stopped.

foreach name (wordlist)
...
end
The variable **name** is successively set to each member of **wordlist**, and the sequence of commands between this command and the matching **end** are executed. (Both **foreach** and **end** must appear alone on separate lines.)

The built-in command **continue** may be used to continue the loop prematurely, and the built-in command **break** used to terminate it prematurely. When this command is read from the terminal, the loop is read up once prompting with **?** before any statements in the loop are executed. If you make a mistake typing in a loop at the terminal, you can backspace to retype it (depending on the user setup).

glob wordlist
Like **echo**, but no **** escapes are recognized and words are delimited by null characters in the output. Useful for programs which wish to use the shell to file-name expand a list of words.

goto word
The specified **word** is file-name and command expanded to yield a string of the form **label**. The shell rewinds its input as much as possible and searches for a line of the form **label:**, possibly preceded by blanks or tabs. Execution continues after the specified line.

history
history n
history -r n
history -h n
Displays the history event list; if **n** is given, only the **n** most recent events are printed. The **-r** option reverses the order of printout to be most recent first rather than oldest first. The **-h** option causes the history list to be printed without leading numbers. This is used to produce files suitable to **source** using the **-h** option.

if (expr) command
If the specified expression evaluates true, the single **command** with arguments is executed. Variable substitution on **command** happens early, at the same time it does for the rest of the **if** command. **command** must be a simple command, not a pipeline, a command list, or a parenthesized command list. Input/output redirection occurs even if **expr** is false, the second command is **not** executed.

if (expr) then
...
else if (expr2) then
...
else
...
endif
If the specified **expr** is true, the commands up to the first **else** are executed; otherwise, if **expr2** is true, the commands up to the second **else** are executed, etc. Any number of **else-if** pairs are possible; only one **endif** is needed. The **else** part is likewise optional. (The words **else** and **endif** must appear at the beginning of input lines; the **if** must appear alone on its input line or after an **else**.)

jobs
jobs -l
Lists the active jobs; given the **-l** options lists process IDs and execution cluster site name, in addition to the normal information.

kill %job
kill -sig % job...
kill pid
kill -sig pid...
kill -l
Sends either the **TERM** (terminate) signal or the specified signal to the specified jobs or processes. Signals are either given by number

or by names (as given in **/usr/include/signal.h**, stripped of the prefix **SIG**). The signal names are listed by **kill -l**. There is no default, saying just **kill** does not send a signal to the current job. If the signal being sent is **TERM** (terminate) or **HUP** (hangup), the job or process will be sent a **CONT** (continue) signal as well.

limit
limit resource
limit resource maximum-use
limit -h
limit -h resource
limit -h resource maximum-use

Limits the consumption by the current process and each process it creates to not individually exceed **maximum-use** on the specified **resource**. If no **maximum-use** is given, the current limit is printed; if no **resource** is given, all limitations are given. If the **-h** flag is given, the hard limits are used instead of the current limits. The hard limits impose a ceiling on the values of the current limits. Only the superuser may raise the hard limits, but a user may lower or raise the current limits within the legal range.

Resources controllable currently include **cputime** (the maximum number of CPU-seconds to be used by each process), **filesize** (the largest single file which can be created), **datasize** (the maximum growth of the data region via **sbrk** (see **sbrk** in *AIX Operating System Technical Reference*) beyond the end of the program text), **stacksize** (the maximum size of the automatically-extended stack region), and **coredumpsize** (the size of the largest core dump that will be created).

Maximum-use may be given as a (floating-point or integer) number followed by a scale factor. For all limits other than **cputime**, the default scale is **k** or **kilobytes** (1024 bytes); a scale factor of **m** or **megabytes** may also be used. For **cputime**, the default scaling is **seconds**, **m** for minutes or **h** for hours, or a time of the form **mm:ss** giving minutes and seconds, may be used.

For both **resource** names and scale factors, unambiguous prefixes of the names suffice.

logout

Terminates a login shell. Especially useful if **ignoreeof** is set.

migrate pid
migrate -site pid
migrate % job
migrate -site % job
migrate -site

Moves a process or job between sites. In the first and third forms (without a **-site** flag) the process or job is migrated to the site on which the **csh** is running. In the second and fourth forms (that include a **-site** flag), the process or job is migrated to the specified site. In the fifth form, the **csh** itself will migrate to the specified site. In all cases, the site to which the process or job is to migrate must be of the same CPU type as the site on which the process or job is running, or no migration will occur.

When the shell is migrated by **migrate -site** (or **migrate -site** $$), the shell will perform a **setlocal** system call to set the <LOCAL> to the default local for the new site. (See the **setlocal** system call in *AIX Operating System Technical Reference* and "printlocal" in topic 1.1.327).

nice
nice +number
nice command
nice +number command

The first form sets the scheduling priority for this shell to 4. The second form sets the priority to the given number. The final two forms run the specified command at priority 4 and **number** respectively. The greater the number, the less CPU the process will get. The superuser may specify negative priority by using **nice -number. command** is always executed in a subshell, and the restrictions placed on commands in simple **if** statements apply.

nohup
nohup command

The first form can be used in shell scripts to cause hangups to be ignored for the remainder of the script. The second form causes the specified **command** to be run with hangups ignored. All processes detached with **&** are effectively **nohup**'ed.

notify
notify %job...

Causes the shell to notify the user asynchronously when the status of the current or specified **jobs** changes; normally, notification is presented before a prompt. This is automatic if the shell variable **notify** is set.

onsite [-v] sitename command [arg...]
onsite [-v] sitenumber command [arg...]
onsite [-v] sitetype command [arg...]

The **command** is run on the specified site or on a site of the specified **sitetype**. If a **sitename** or **sitenumber** is given, the command is run on the specified site if it is accessible in the current partition. If a **sitetype** is given, the command is run on a site of the specified type if any are available.

The possible **sitetype** values are:

i386	An AIX PS/2 system.
i370	An AIX/370 system.
xa370	An XA AIX/370 system.
s370	A non-XA AIX/370 system.

The command is run on the chosen site with the <LOCAL> alias set to refer to that site. The **onsite** prefix can precede built-in commands, which still execute locally but has the <LOCAL> alias set to the specified site. The **arg...** is passed as arguments to **command**.

Input-output redirection is done before the execution site is set, so that the <LOCAL> alias used for input-output redirection may be different than the one that the executed command will use. This is done for compatibility with shells that don't have the **onsite** command built in and so execute the separate **onsite** command (see "onsite, on" in topic 1.1.306), but may cause some confusion. For example, if the **csh** is running on site atlas, then **onsite polaris cat < /tmp/foo** will display the contents of **/atlas/tmp/foo**, while **onsite polaris cat**

/tmp/foo will display the contents of **/polaris/tmp/foo**, since in the first case the shell opened **/tmp/foo** using its <LOCAL> alias, while in the second the **cat** command opened **/tmp/foo** with its <LOCAL> alias set to polaris' local. If the **-v** option is given, the shell will tell the user which site is being used in the command.

onintr
onintr -
onintr label
controls the action of the shell on interrupts. The first form restores the default action of the shell on interrupts, which is to terminate shell scripts or to return to the terminal command input level. The second form, **onintr -,** causes all interrupts to be ignored. The final form causes the shell to execute a **goto label** when an interrupt is received or a child process terminates because it was interrupted.

In any case, if the shell is running detached and interrupts are being ignored, all forms of **onintr** have no meaning, and interrupts continue to be ignored by the shell and all invoked commands.

popd
popd +n
Pops the directory stack, returning to the new top directory. With an argument **+n** discards the **n**th entry in the stack. The directory stack elements are numbered from 0 starting at the top.

pushd
pushd name
pushd +n
With no arguments, **pushd** exchanges the top two elements of the directory stack. Given a **name** argument, **pushd** changes to the new directory (as with **cd**) and pushes the old current working directory (as in **cwd**) onto the directory stack. With a numeric argument, rotates the **n**th argument of the directory stack around to be the top element and changes to it. The members of the directory stack are numbered from the top starting at 0.

rehash
Causes the internal hash table of the contents of the directories in the **path** variable to be recomputed. This is needed if new commands are added to directories in the **path** while you are logged in. This should only be necessary if you add commands to one of your own directories, or if a systems programmer changes the contents of one of the system directories.

repeat count command
The specified **command**, which is subject to the same restrictions as the **command** in the one line **if** statement above, is executed **count** times. I/O redirections occur exactly once, even if **count** is 0.

set
set name
set name=word
set name[index]=word
set name=(wordlist)
The first form of the command shows the value of all shell variables. Variables which have other than a single word as value print as a parenthesized word list. The second form sets **name** to the null string. The third form sets **name** to the single **word**. The fourth form

sets the **index'ed** component of **name** to **word**; this component must already exist. The final form sets **name** to the list of words in **wordlist**. In all cases, the value is command and file-name expanded.

These arguments may be repeated to set multiple values in a single set command. Note, however, that variable expansion happens for all arguments before any setting occurs.

setenv
setenv name value
setenv name
The first form lists all current environment variables. The second form sets the value of environment variable **name** to be **value**, a single string. The last form sets **name** to an empty string. The most commonly used environment variable **USER**, **TERM**, and **PATH** are automatically imported to and exported from the **csh** variables **user**, **term**, and **path**; there is no need to use **setenv** for these.

If you modify the environment variables **LANG**, **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGE**, **LC_TIME**, or **LC_MONETARY**, the current international character support environment and collating sequences are changed as specified for subsequent commands executed from the shell.

setspath [LOCAL | site | cpu] ...
Sets the site path. Sites may be specified by name or by number. CPU types may be specified by the same names which are used in hidden directories. If an argument is specified as **LOCAL**, a **nullsite** site path entry is created (see **setspath** in the *AIX Operating System Technical Reference*). The arguments to this command are file name and command substituted.

Use this command with caution. A poorly formed site path can make it difficult to execute any commands.

setxvers
setxvers string
The first form removes any experimental-version prefix. The second form sets the experimental-version prefix to **string**. The prefix is used when searching hidden directories for executable files. A hidden directory component whose name begins with the prefix is selected for execution in preference to a component whose name does not begin with **string**. The experimental-version prefix is inherited by child processes. See *AIX Operating System Technical Reference* for more information on hidden directories and executable hidden directory components.

shift
shift variable
The members of **argv** are shifted to the left, discarding **argv[1]**. It is an error for **argv** not to be set or to have less than one word as value. The second form performs the same function on the specified **variable**.

source name
source -h name
The shell reads commands from **name**. **source** commands may be nested; if they are nested too deeply, the shell may run out of file descriptors. An error in a **source** at any level terminates all nested **source** commands. Normally input during **source** commands is not placed on the history list; the **-h** option causes the commands to be placed in the

history list without being executed.

stop
stop %job...
Stops the current or specified job which is executing in the background.

suspend
Causes the shell to stop in its tracks, much as if it had been sent a stop signal with ^**Z**. This is most often used to stop shells started by **su**. (See "su" in topic 1.1.449.)

switch (string)
case str1:
...
breaksw
...
default:
...
breaksw
endsw
Each case label is successively matched, against the specified **string**, which is first command and file-name expanded. The file metacharacters *****, **?**, and, **[...]** may be used in the case labels, which are variable expanded. If none of the labels match before a **default** label is found, execution begins after the **default** label. Each **case** label and the **default** label must appear at the beginning of a line. The command **breaksw** causes execution to continue after the **endsw**. Otherwise, control may fall through **case** labels and **default** labels, as in C. If no label matches and there is no **default**, execution continues after the **endsw**.

time
time command
With no argument, a summary of time used by this shell and its children is printed. If arguments are given, the specified simple **command** is timed, and a time summary as described under the **time** variable is printed. If necessary, an extra shell is created to print the time statistic when the **command** completes. A sample output of **time command** is as follows:

```
  0.5u 0.4s O:04.3 24% 18+125k 20+0io 14pf+0w
```

0.5u	User-mode CPU time consumed by the **command**.
0.4s	CPT time consumed by the system running on behalf of the command.
18+125k	Time-averaged integral of the (shared) text space used by the command, in units of kilobytes-times-seconds.
0:04.3	Prints the elapsed time to tenths of a second.
24%	Percentage of CPU utilization of this command, computed as the sum of user and system CPU time divided by the elapsed time.
20+0io	Prints the number of block-I/O reads performed on behalf of the process.

14pf+0w Prints number of swaps occurring during execution of the command.

umask
umask value
The file creation mask is displayed (first form) or set to the specified **value** (second form). The mask is given in octal. Common values for the mask are 002, giving all access to the group and read and execute access to others, or 022, giving all access except no write access for users in the group or others.

unalias pattern
All aliases whose names match the specified **pattern** are discarded. Thus all aliases are removed by **unalias ***. It is not an error for nothing to be **unaliased**.

unhash
Use of the internal hash table to speed location of executed programs is disabled.

unlimit
unlimit resource
unlimit -h
unlimit -h resource
Removes the limitation on **resource**. If no **resource** is specified, all **resource** limitations are removed. If **-h** is given, the corresponding hard limits are removed. Only the superuser may do this.

unset pattern
All variables whose names match the specified **pattern** are removed. Thus all variables are removed by **unset ***; this has noticeably distasteful side-effects. It is not an error for nothing to be **unset**.

unsetenv pattern
Removes all variables whose name match the specified **pattern** from the environment. See also the **setenv** command.

wait
All background jobs are waited for. If the shell is interactive, an INTERRUPT can disrupt the wait, at which time the shell prints names and job numbers of all jobs known to be outstanding.

while (expr)
...
end
While the specified expression evaluates nonzero, the commands between the **while** and the matching end are evaluated. **break** and **continue** may be used to terminate or continue the loop prematurely. (The **while** and **end** must appear alone on their input lines.) Prompting occurs here the first time through the loop as for the **foreach** statement if the input is a terminal.

%job
Brings the specified **job** into the foreground.

%job &
Continues the specified **job** in the background.

@
@ **name = expr**

@ name[index] = expr

The first form prints the values of all the shell variables. The second form sets the specified **name** to the value of **expr**. If the expression contains **<**, **>**, **&**, or **|**, then at least this part of the expression must be placed within **( )**. The third form assigns the value of **expr** to the **indexed** argument of **name**. Both **name** and its **indexed** component must already exist.

The operators ***=**, **+=**, etc. are available, as in C. The space separating the name from the assignment operator is optional. Spaces are, however, mandatory in separating components of **expr**, which would otherwise be single words.

Special postfix **++** and **- -** operators increment and decrement **name** respectively, for example, @ **i++**.

Predefined and Environment Variables

The following variables have special meaning to the **csh** shell. Of these, **argv**, **cwd**, **home**, **path**, **prompt**, **shell**, and **status** are always set by the shell. Except for **cwd** and **status**, this setting occurs only at initialization; these variables will not then be modified unless this is done explicitly by the user.

The **csh** shell copies the environment variable **USER** into the variable **user**, **TERM** into **term**, and **HOME** into **home**, and copies these back into the environment whenever the normal shell variables are reset. The environment variable **PATH** is likewise handled; it is not necessary to worry about its setting other than in the file **.cshrc**, as inferior **csh** processes will import the definition of **path** from the environment, and re-export it if you change it.

argv	Set to the arguments to the shell, it is from this variable that positional parameters are substituted, that is, **$1** is replaced by **$argv[1]**, etc.
cdpath	Gives a list of alternate directories searched to find subdirectories in **chdir** commands.
cwd	The full path name of the current directory.
echo	Set when the **-x** command line option is given. Causes each command and its arguments to be echoed just before it is executed. For non-built-in commands, all expansions occur before echoing. Built-in commands are echoed before command and file-name substitution, since these substitutions are then done selectively.
filec	Enable file-name completion.
histchars	Can be given a string value to change the characters used in history substitution. The first character of its value is used as the history substitution character, replacing the default character **!**. The second character of its value replaces the character **^** in quick substitutions.
history	Can be given a numeric value to control the size of the history list. Any command which has been referenced within this many events will not be discarded. Too large values of **history** may run the shell out of memory. The last

executed command is always saved on the history list.

home	The home directory of the invoker, initialized from the environment. The file name expansion of **~** refers to this variable.
ignoreeof	If set, the shell ignores end-of-file from input devices which are terminals. This prevents shells from accidentally being killed by **Ctrl-D**s.
mail	The files where the shell checks for mail. This is done after each command completion, which will result in a prompt if a specified interval has elapsed. The shell says You have new mail if the file exists with an access time not greater than its modify time. If the first word of the value of **mail** is numeric, it specifies a different mail checking interval, in seconds, than the default, which is 10 minutes. If multiple mail files are specified, the shell says New mail in **name** when there is mail in the file **name**.
noclobber	As described in the section on **"Input/output"**, restrictions are placed on output redirection to insure that files are not accidentally destroyed and that **>>** redirections refer to existing files.
noglob	If set, file name expansion is inhibited. This is most useful in shell scripts which are not dealing with file names, or after a list of file names has been obtained and further expansions are not desirable.
nonomatch	If set, it is not an error for a file-name expansion to not match any existing files; rather, the primitive pattern is returned. It is still an error for the primitive pattern to be malformed, for example, **echo [** still gives an error.
notify	If set, the shell notifies asynchronously of job completions. The default is rather to present job completions just before printing a prompt.
path	Each word of the **path** variable specifies a directory in which commands are to be sought for execution. A null word specifies the current directory. If there is no **path** variable, only full path names will execute. The usual search path is **.**, **/bin**, and **/usr/bin**, but this may vary from system to system. For the superuser, the default search path is **/etc**, **/bin**, and **/usr/bin**. A shell which is given neither the **-c** nor the **-t** option will normally hash the contents of the directories in the **path** variable after reading **.cshrc** and each time the **path** variable is reset. If new commands are added to these directories while the shell is active, it may be necessary to give the **rehash**, or the commands may not be found.
prompt	The string which is printed before each command is read from an interactive terminal input. If a **!** appears in the string and ONLY if the **!** is preceded by a **** will it be replaced by the current event number. Default is **%**, or **#**

for the superuser.

savehist Is given a numeric value to control the number of entries of the history list that are saved in **~/.history** when the user logs out. Any command which has been referenced within this many events will be saved. During startup the shell sources **~/.history** into the history list, enabling history to be saved across logins. Too large values of **savehist** will slow down the shell during startup.

shell The file in which the shell resides. This is used in forking shells to interpret files which have execute bits set but which are not executable by the system. (See the description of **"Nonbuilt-in Command Execution"** below.) Initialized to the (system-dependent) home of the shell.

status The status returned by the last command. If it terminated abnormally, 0200 is added to the status. Built-in commands which fail return exit status **1**, all other built-in commands set status **0**.

time Controls automatic timing of commands. If the first or only word of its value is numeric, then any command that takes more than this many CPU seconds will cause a line showing the resources utilized by the command and all its children to be printed. If the value of the **time** variable has two words, the second is used as a format string controlling the printing of the resource information. The string is interpreted similarly to the format string used by **printf** (see **printf** in *AIX Technical Reference*). The % character identifies characters that will be replaced by certain resource information. If the value of the **time** variable has only one word, the string used is "%Uu %Ss %E %P %X+%Dk %I+%Oio %Fpf+%Ww". The following characters can be used in the format string:

U Prints the user-mode CPU time consumed by the command.

S Prints the CPU time consumed by the system running on behalf of the command.

E Prints the elapsed time to tenths of a second.

P Prints the percentage CPU utilization of this command. This is computed as the sum of user and system CPU time divided by the elapsed time.

W Prints number of swaps suffered by the command.

X Prints the time-averaged integral of the (shared) text space used by the command, in units of kbytes times seconds.

D Prints the time-averaged integral of the (unshared) data and stack space used by the command, in units of kbytes times seconds.

K Prints the time-averaged integral of the total space used by the command, in units of kbytes times seconds.

M Prints the maximum amount of memory in use by the command at any time, in units of kbytes.

F Prints the number of "major" page faults suffered by the command. These are page faults where the page had to be brought in from disk.

R Prints the number of "minor" page faults suffered by the command.

I Prints the number of block-I/O reads performed on behalf of the process.

verbose Set by the **-v** command line option, causes the words of each command to be printed after history substitution.

Nonbuilt-in Command Execution

When a command to be executed is found to not be a built-in command, the shell attempts to execute the command via **execve**. Each word in the variable **path** names a directory from which the shell will attempt to execute the command. If it is given neither a **-c** nor a **-t** option, the shell will hash the names in these directories into an internal table so that it will only try an **exec** in a directory if there is a possibility that the command resides there. This greatly speeds command location when a large number of directories are present in the search path. If this mechanism has been turned off (via **unhash**) or if the shell was given a **-c** or **-t** argument (and in any case for each directory component of **path** which does not begin with a **/**), the shell concatenates with the given command name to form a path name of a file, which it then attempts to execute.

Parenthesized commands are always executed in a subshell. Thus **(cd ; pwd) ; pwd** prints the **home** directory; leaving you where you were (printing this after the home directory); while **cd ; pwd** leaves you in the **home** directory. Parenthesized commands are most often used to prevent **chdir** from affecting the current shell.

If the file has execute permissions but is not an executable binary to the system, it is assumed to be a file containing shell commands, and a new shell is spawned to read it.

If there is an **alias** for **shell**, the words of the alias will be prepended to the argument list to form the shell command. The first word of the **alias** should be the full path name of the shell (for example, **$shell**). This is a special, late-occurring case of **alias** substitution and only allows words to be prepended to the argument list without modification.

Argument List Processing

If argument 0 to the shell is **-**, this is a login shell. The flag arguments are interpreted as follows:

-b This flag forces a break from option processing, causing any further shell arguments to be treated as nonoption arguments. The remaining arguments will not be interpreted as shell options. This may be used to pass options to a shell script without confusion or possible subterfuge. The shell will not run a set-user-ID script without this option.

-c Commands are read from the (single) following argument, which must be present. Any remaining arguments are placed in **argv**.

-e The shell exits if any invoked command (other than built-in commands) terminates abnormally or yields a nonzero exit status. Without this flag, the shell does not exit if any invoked command returns a nonzero exit status.

Note: The shell always exits if any built-in command returns a nonzero exit status, even if the **-e** flag is set.

-f The shell will start faster, because it will neither search for nor execute commands from the file **.cshrc** in the invoker's home directory.

-i The shell is interactive and prompts for its top-level input, even if it appears not to be a terminal. Shells are interactive without this option if their inputs and outputs are terminals.

-n Commands are parsed but not executed. This aids in syntactic checking of shell scripts.

-s Command input is taken from the standard input.

-t A single line of input is read and executed. A **** can be used to escape the new-line character at the end of this line and continue onto another line.

-v Causes the **verbose** variable to be set, with the effect that command input is echoed after history substitution.

-V Causes the **verbose** variable to be set even before **.cshrc** is executed.

-x Causes the **echo** variable to be set, so that commands are echoed immediately before execution.

-X Causes the **echo** variable to be set even before **.cshrc** is executed.

After processing of flag arguments, if arguments remain but none of the **-c**, **-i**, **-s**, or **-t** options was given, the first argument is taken as the name of a file of commands to be executed. The shell opens this file and saves its name for possible re-substitution by **$0**.

If the first characters of the shell procedure are **#!shell_path name**, **csh** runs the specified shell to process the procedure. Otherwise, **csh** runs the standard shell (**sh**). Remaining parameters initialize the **arg** variable. For more information on the **#!shell_path name** comment, see the **exec** system call in *AIX Operating System Technical Reference*.

Signal Handling

The shell normally ignores **QUIT** signals. Jobs running detached (either by **&** or the **bg** or **%...&** commands) are immune to signals generated from the keyboard, including **HANGUPS**. Other signals have the values which the shell inherited from its parent. The shell's handling of **INTERRUPTS** and **TERMINATE** signals in shell scripts can be controlled by **onintr**. Login shells catch the **TERMINATE** signal; otherwise, this signal is passed on to children from the state in the shell's parent. In no case are **INTERRUPTS** allowed when a login shell is reading the file **.logout**.

Files

~/.cshrc	Read at beginning of execution by each shell.
~/.login	Read by login shell, after **.cshrc** at login.

~/.logout	Read by login shell, at logout.
/bin/sh	Standard shell.
/tmp/sh*	Temporary file for **<<**.
/etc/passwd	Source of home directories for **~name**.

Limitations

Words can be no longer than 1024 characters. The system limits argument lists to a maximum of 10240 characters. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, these character limits may be reduced by as much as 50%, depending on the character code set being used.

The number of arguments to a command which involves file name expansion is limited to 1/6th the number of characters allowed in an argument list. Command substitutions may substitute no more characters than are allowed in an argument list. To detect looping, the shell restricts the number of **alias** substitutions on a single line to 20.

Notes

When a command is restarted from a stop, the shell prints the directory it started in, if this is different from the current directory. This can be misleading (that is, wrong), as the job may have changed directories internally.

Shell built-in functions are not stoppable and restartable, nor are command sequences of the form **command1; command2; command3**. If you suspend **command2**, the shell immediately executes **command3**. This is especially noticeable if this expansion results from an **alias**. It suffices to place the sequence of commands in parentheses, **(command1; command2; command3)**, to force it to a subshell.

Related Information

See the following commands: "cd" in topic 1.1.53, "make" in topic 1.1.254, "pr" in topic 1.1.322, and "sh, Rsh" in topic 1.1.420.

See the **access**, **exec**, **fork**, **pipe**, **umask**, and **wait** system calls, the **a.out** and **environ** files, and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See "Overview of International Character Support" in *Managing the AIX Operating System*.

1.1.101 csplit

Purpose
Splits files by context.

Syntax

```
          +---------------+
csplit ---¦ +-----------+ +-- file -- parm --¦
          +-¦ -f prefix +-+              ¦
            ¦ -k        ¦¦        +------+
           ¦¦ -s        ¦¦
           ¦+-----------+¦
           +-------------+
```

Note: This command does not have MBCS support.

Description
The **csplit** command reads a **file** and separates it into segments defined by the specified parameters (**parm**...). By default, the **csplit** command writes these segments to files **xx00**...**xxn**, where **n** is the number of parameters (**parm**) listed on the command line (**n** may not be greater than 99). These new files get the following pieces of **file**:

00: From the start of **file** up to, but not including, the line referenced by the first **parm**.
01: From the line referenced by the first **parm** up to the line referenced by the second **parm**.
.
.
.
n+1: From the line referenced by the last **parm** to the end of **file**.

The **csplit** command does not alter the original **file**.

The specified **parm**s can be a combination of the following:

/pattern/ Creates a file that contains the segment from the current line up to (but not including) the line containing **pattern**, which becomes the current line.

%pattern% Makes the line containing **pattern** the current line but does not create a file for the segment.

+num

-num Moves forward or backward the specified number of lines from the line matched by an immediately preceding **pattern** parameter (for example, **/Page/-5**).

linenum Creates a file containing the segment from the current line up to (but not including) **linenum**, which becomes the current line.

{number} Repeats the preceding argument the specified **number** of times. This number can follow any of the **pattern** or **linenum** parameters. If it follows a **pattern** parameter, the **csplit** command reuses that **pattern** the specified number of times. If it follows a **linenum** parameter, the **csplit** command splits the file from that point every **linenum** of lines for the specified number of times.

Quote all **pattern** parameters that contain blanks or other characters special to the shell. Patterns may not contain embedded new-line characters. In an expression such as **[a-z]**, the minus means "through" according to the current collating sequence. A collating sequence may define **equivalence classes** for use in character ranges. See "Overview of International Character Support" in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

Flags

-f prefix	Specifies the **prefix** name for the created file segments. **xx** is the default **prefix**.
-k	Leaves created file segments intact in the event of an error.
-s	Suppresses the display of character counts.

Examples

1. To split the text of a book into a separate file for each chapter:

   ```
   csplit book "/^Chapter *[1-9]/" {8}
   ```

   This command creates files named **xx00**, **xx01**, **xx02**,...,**xx09**, which contain individual chapters of the file **book**. Each chapter begins with a line that contains only the word **Chapter** and the chapter number. The file **xx00** contains the front matter that comes before the first chapter. The repeat argument "{8}" after the pattern allows up to nine chapters.

2. To specify the prefix for the created file names:

   ```
   csplit -f chap book "/^Chapter *[1-9]/" {8}
   ```

   This command splits **book** into files named **chap00**, **chap01**, **chap02**,...,**chap09**.

Related Information

See the following commands: "ed, red" in topic 1.1.147, "sh, Rsh" in topic 1.1.420, and "regcmp" in topic 1.1.367.

See the **regxp** file in *AIX Operating System Technical Reference*.

See "Overview of International Character Support" in *Managing the AIX Operating System*.

1.1.102 ct

Purpose
Dials an attached terminal and issues a login process.

Syntax

```
      +-------+   +-------+   +------+   +------+   +------------+
ct ---¦       +---¦       +---¦      +---¦      +---¦            +--- telno --
      +- -wn -+   +- -xn -+   +- -h -+   +- -v -+   +- -s speed -+ +---------
```

Description

The Basic Networking Utilities (BNU) command **ct** enables a user on a remote ASCII terminal, such as an IBM 3161 or a DEC VT100, to communicate with an AIX system over a telephone line attached to a modem at each end of the connection. The user on the remote terminal can then log in and work on the AIX system.

A user on the local system issues **ct** with the appropriate telephone number, **telno**, to call the modem attached to the remote terminal. The **telno** may include the digits 0 - 9, minus signs (-) representing delays, equal signs (=) representing secondary dial tones, asterisks (*), and pound/number signs (#). The phone number may contain a maximum of 31 characters. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the 31-character limit may be reduced by as much as 50%, depending on the character code set being used.

When the connection is established, **ct** issues an AIX login prompt that is displayed on the remote terminal. The user on the remote terminal enters an AIX login name at the prompt, and AIX opens a new shell. The user at the remote terminal then proceeds to work on the AIX system just like a local user.

The **ct** command is useful in the following situations:

When a user working offsite needs to communicate with a local syste under strictly supervised conditions. Because the local system contacts the remote terminal, the remote user does not need to know the phone number of the local system.

When the cost of the connection should be charged either to the loca site, or to a specific account on the calling AIX system. If the remote user has the appropriate access permission and can make outgoing calls on the attached modem, that user can make the equivalent of a collect call. The remote user calls the specified local system, logs in, and issues the phone number of the remote terminal **with** the **-h** flag. The local system hangs up the initial link so that the remote terminal is free for an incoming call, and then calls back to the modem attached to the remote terminal.

The **ct** command is not as flexible as the BNU command **cu**. For example, the user can not issue AIX commands on the local system while connected to a remote system via **ct**. However, the **ct** command does have two features not available with **cu**:

The user can instruct **ct** to continue dialing the specified number until the connection is established or a set amount of time has elapsed.

The user can specify more than one telephone number at a time t instruct **ct** to continue dialing each modem until a connection is established over one of the lines.

If the user specifies alternate dialing paths by entering more than one number on the command line, **ct** tries each line listed in the file **/usr/adm/uucp/Devices** until it finds an available line with appropriate attributes, or runs out of entries. If there are no free lines, **ct** asks if it should wait for one, and if so, for how many minutes. The **ct** command continues to try to open the dialers at 1-minute intervals until the specified time is exceeded. The user can override this prompt by specifying a time with the **-wn** flag when entering the command.

After the user logs out, **ct** prompts the user on the remote terminal with a reconnect option; the system can either display a new login prompt or drop the line.

Notes:

1. In order to establish a **ct** connection, the remote user generally contacts a local user (with a regular phone call) and asks the local user to issue the command.

2. Before issuing the **ct** command, be certain that the remote terminal is attached to a modem that can answer the telephone.

Flags

-wn Allows the dialogue to be overridden by specifying **n** as the maximum number of minutes that **ct** is to wait for a line. The command then dials the remote modem at 1-minute intervals until the connection is established or the specified time has elapsed.

-xn Used for debugging. Produces detailed information about the command's execution on standard error output on the local system. The debugging level, **n**, is a single digit between 0 and 9. The recommended default is 9.

-h Allows **ct** to hang up the current line to answer a return call.

-v Allows **ct** to send a running narrative to standard error output.

-sspeed
Sets the data rate where **speed** is expressed in baud. The default is 1200.

Examples

1. To connect to a modem with an internal number 4-1589 (the - is optional):

```
ct 41589
```

The system responds:

```
Allocated dialer at 1200 baud
Confirm hang_up? (y to hang_up)
```

2. To dial a modem connected to a local telephone number (dialing 9 for an outside line and specifying a 3-minute wait time):

```
ct -w3 9=2453017
```

3. To dial a long-distance number (specifying an outside line and a 5-minute wait):

```
ct -w5 9=15026647003
```

Files

/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.

Related Information

See the following commands: "cu" in topic 1.1.105 and "login" in topic 1.1.241.

1.1.103 ctab

Purpose
Produces a collating table.

Syntax

```
       +- -i ctab.in -+   +- -o ctab.out -+   +---------------+
ctab ---¦              +---¦               +---¦               +--¦
       +- -i infile --+   +- -o outfile --+   +- -c cputype --+
```

Description
The **ctab** command takes an input file (by default, named **ctab.in** and found in the current directory) and produces a binary file (by default, named **ctab.out**) containing a collating table. These output files are stored in a conventional directory. Programs that need the current collating and case information use the **NLCTAB** environment variable to access that information.

The following conventions are used to make it easier to set up a table file:

One line of information is present for each character explicitl named.

A line beginning with the word **option** serves to change one or more of the default conditions or metacharacters built into the **ctab** command. An **option** line contains a set of name/value pairs, with each half of each pair delimited by tab or space characters. The following is a list of recognized names:

eclass	Turns the use of equivalence classes on or off globally. The assigned value must be **on** (the default) or **off**.
sep	Uses the assigned value as the field separator character. The default value is **:** (colon).
trans	Uses the assigned value of the "translate" indicator in subject character fields. The default character is | (vertical bar).
repeat	Uses the assigned value as the "same as last line" indicator in subject character field. The default value is **and** (circumflex).
comment	Uses the assigned value as the comment character. The default value is the # character.

The order of the per-character input lines specifies the collatin sequence.

By default, fields on a line are separated by colons. Tabs or space may surround fields or separators. You can change the separator character with an option line.

Use an octal or hex escape sequence to name a non-printable character An octal escape sequence is preceded by a **** (backslash) or **\0**, and a hex escape sequence is preceded by a **\0x** or **\0X**. A backslash character that does not form part of a valid escape sequence serves to strip the following character, including a second backslash, of any special meaning it otherwise would have. For example, to include the colon character in the collating sequence, use the following line:

```
\::
```

The input file format includes a comment convention, namely that the remainder of the line following a # character is ignored. The comment character can be changed with an **option** line.

Note: The input file must be pre-processed by the **dd** command prior to its use in the **ctab** command.

Input File Specification
Use the following rules to build **infile**, entering field information for each line:

1. The first field on a line contains the subject character, a character to be inserted into the collating sequence at that point.

   This subject character definition can include a translation mechanism:

   - Instead of a single character, this field may contain two or more characters that are to be collated as a single unit, or

   - The single subject character may be followed by a (|) vertical bar and a single- or multiple-character string. The vertical bar indicates that the first character will be translated to the second string before being collated.

     For example, to treat an "é" (e acute) as equivalent to the character "e", use the following line:

     ```
     é|e
     ```

   - One restriction is placed on the translation mechanism: the subject character cannot be contained in the translated string of characters. For example, the following line is illegal:

     ```
     o|oe
     ```

   Any form of the first field may contain a trailing circumflex (**and**) to indicate that the current character is to collate to the same value as the preceding one. However, a circumflex following a translation string is illegal because the subject character to be translated has no inherent collating value.

   If the subject field contains a string of multiple characters (to collate as a unit), its first character must be declared elsewhere to establish the default collating sequence of that character.

   The translate and collating no-change characters can be changed with **option** lines.

2. The second and third fields specify whether a character is alphabetic and what its lowercase and uppercase equivalents are:

   If a subject character is to be treated as a lowercase alphabetic, the second field on its line is its uppercase equivalent, and the third field must be **l** or **L**.

If a subject character is to be treated as a uppercase alphabetic, the second field on its line is its lowercase equivalent, and the third field must be **u** or **U**.

If a subject character is to be treated as a control character or a space character, the third field must be **c**, **C**, **s**, or **S**.

Each explicitly named character whose line contains a non-null second field is considered alphabetic (that is, matched by **isalpha**). Characters that do not have an uppercase or lowercase equivalent (that is, that have a null second field) but that you wish to be considered alphabetic should contain a third field that is **l**, **L**, **u**, or **u**.

3. The fourth field on a line is used explicitly to specify the first character in the equivalence class of the subject character. The members of one equivalence class must be consecutively listed in the input file.

   There cannot be any gaps within a particular equivalence class. For example, the following lines put the characters **a**, **b**, and **c** in the same equivalence class:

   ```
     a:A:l:a
     b:B:l:a
     c:C:l:a
   ```

   As a convenience, if the fourth field is not specified, the group of consecutive characters with blank fourth fields, provided that they are all based on the same Roman alphabetic character, are placed in the same equivalence class. To reiterate, only characters with the same base are placed into the same equivalence class by default. If you wish to have many characters from different bases belong to one equivalence class, as in the example above, the first character of the equivalence class has to be specified in the fourth field for every character specified.

   It is illegal to specify an equivalence character that comes later in the collating sequence. The fourth field can refer only to characters that have already been mentioned.

   All international character support characters not based on Roman alphabetic characters by default are the sole members of their equivalence class.

Characters not named in the table file that have an ordinal value (that is, a value as a **char**) below the ordinal value of the lowest-valued character named are put into the collating sequence below the first character in the table file. All other characters not named in the table file are put into the collating sequence above the last character in the table file.

The standard characters for decimal and hexadecimal digits are always marked as digits (to be matched by **isdigit** and **isxdigit**). All other printable characters not marked as alphabetic are marked as punctuation.

Flags

-c cputype	Specifies the CPU type of the output file. In clusters with both i386 and i370 machines, the output file is created as a hidden directory. In clusters with one CPU type, the output is a file.
-i infile	Specifies the name of the input file (**ctab.in**, by default).
-o outfile	Specifies the name of the output file (**ctab.out**, by default).

Files

/usr/lib/mbcs/*.ctab	Contains the various default and locale-dependent collation table sources and binaries. Collation source files have the **ctab** extension.

Related Information

See the **isalpha**, **isdigit**, **isxdigit**, **nls**, and **getenv** subroutines in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.104 ctags

Purpose
Makes a file of tags to help locate objects in source files.

Syntax

```
        +--- -F ----+    +--------------+  +------ file -------+
ctags ---¦ +-------+ +----¦              +--¦                  +-¦
        +-¦ -B -u +-+    +- -f tagsfile -+  +- <list_of_files  -+
          ¦ -F -v ¦¦
         ¦¦ -a -w ¦¦
         ¦¦ -t -x ¦¦
         ¦+-------+¦
         +---------+
```

Description
The **ctags** command makes a tags file for the **ex** and **vi** editors from the C, Pascal, and FORTRAN source files specified in the **list_of_files** parameter. A tags file gives the locations of specified objects (in this case, functions) in a group of files. Each line of the tags file contains the object name, the file in which it is defined, and an address specification for the object definition. Functions are searched with a pattern. Specifiers are given in separate fields on the line, separated by blanks or tabs. Using the tags file, the **ex** and **vi** editors can quickly find these object
definitions.

If a file name ends in **.c** or **.h**, it is assumed to be a C source file and is searched for C routine and macro definitions. Other files first are examined to see if they contain any Pascal or FORTRAN routine definitions; if not, they are processed again for C definitions.

The tag **main** is treated specially in C programs. The tag formed is created by prefixing **M** to the file name, removing a trailing **.c** (if any), and removing the leading path name components. This makes the use of the **ctags** command practical in directories with more than one program.

Notes:

1. Recognition of the keywords **function**, **subroutine**, and **procedure** in FORTRAN and Pascal code is performed in a simple manner. No attempt is made to deal with block structures; if you have two Pascal procedures with the same name but in different blocks, the **ctags** command may yield inadequate results.

2. The **ctags** command does not recognize the **#ifdef** statement.

Flags

-a Appends to the tags file.

-B Uses backward searching patterns (?...?).

-f tagsfile
Creates a file of the specified name **tagsfile** instead of the default **tags**.

-F Uses forward searching patterns (/.../).

-t	Creates tags for typedefs, structs, unions, and enums.
-u	Updates the specified files in tags; that is, all references to them are deleted, and the new values are appended to the file. This process may be slow. (It is usually faster to simply rebuild the tags file.)
-v	Verbose option.
-w	Suppresses warning diagnostics.
-x	Causes the **ctags** command to display a list of object names, the line number and file name on which each is defined, as well as the text of that line. This flag provides a simple index. If you specify this flag, the **ctags** command does not build a tags file.

Files

tags Output tags file.

Related Information

See the following commands: "ex" in topic 1.1.158 and "vi, vedit, view" in topic 1.1.522.

1.1.105 cu

Purpose
Connects directly or indirectly to another UNIX system.

Syntax

```
     +-----------+  +----------+   +-----+   +-----+   +-----+
cu ---¦           +---¦          +---¦     +---¦     +---¦     +---
     +- -s speed -+  +- -l line -+   +- -h -+   +- -t -+   +- -d -+

     +-----+
   +-¦     +-+  +- telno --+
 ---¦ +- -o -+ +---¦        +---¦
   ¦ +-----+ ¦  +-- -n ----+
   +-¦     +-+
     +- -e -+

                                                 +-----+
     +-----------+  +-----+   +-----+   +-¦     +-+
cu ---¦           +---¦     +---¦     +---¦ +- -o -+ +-- -l line --¦
     +- -s speed -+  +- -h -+   +- -d -+   ¦ +-----+ ¦
                                         +-¦     +-+
                                           +- -e -+

                            +-----+
     +-----+  +-----+  +-¦     +-+
cu ---¦     +---¦     +---¦ +- -o -+ +-- system name --¦
     +- -h -+  +- -d -+  ¦ +-----+ ¦
                          +-¦     +-+
                            +- -e -+
```

Description

The Basic Networking Utilities (BNU) command **cu** connects one system to another UNIX system, to a terminal connected to a UNIX system, or, if the proper hardware and software are installed, to a non-UNIX system. The connection can be established over a hard-wired line, or over a telephone line via a modem.

Once the connection is established, a user can be logged in on both systems at the same time, executing commands on either one without dropping the BNU communication link. If the remote computer is also running under UNIX, the user can transfer ASCII files between the two systems.

When using a modem, you can specify the **telno** argument (telephone number) with appropriately placed equal signs for secondary dial tones, or minus signs for delays of 4 seconds. Or you can specify the **system_name** argument (a **uucp** system name) instead; in this case, the **cu** command obtains an appropriate hard-wired line or telephone number from the file **/usr/adm/uucp/Systems**.

After issuing the **cu** command from the local system, press the **Enter** key (carriage return) and then log in to the remote system. When the connection is made, the **cu** command runs as two concurrent processes: the transmit process reads data from standard input and, except for lines beginning with a ~ (tilde), passes that data to the remote terminal. The receive process accepts data from the remote system and, except for lines beginning with a ~, passes it to standard output. To control input from the remote system so the buffer is not overrun, the **cu** command uses an automatic DC3/DC1 (**Ctrl-Q/Ctrl-S**) protocol.

In addition to issuing regular AIX commands on the remote system, the user can also issue special **cu** local commands, which are preceded by a ~. Use these commands to issue AIX commands on the local system and to perform tasks such as transferring files between two UNIX systems.

Notes:

1. The system must be configured to use the **cu** command before you issue this command. Refer to *Managing the AIX Operating System* for details about this configuration.

2. Do not use the **system_name** flag in conjunction with the **-1** and **-s** flags. If you do, the **cu** command connects to the first available line for the requested system name, ignoring the specified line and speed.

Subtopics

1.1.105.1 Local Commands

The transmit process interprets lines beginning with a tilde in the following ways:

~.	Logs the user off the remote computer and terminates the remote connection.
~!	Returns the user to an interactive shell on the local system. Toggle between the local and remote systems by using ~! (remote to local) and pressing **Ctrl-D** (local to remote).
~!cmd...	Executes the command denoted by **cmd** on the local system via **sh -c**.
~$cmd...	Runs the command denoted by **cmd** locally and sends its output to the remote system for execution.
~%cd	Changes the directory on the local system.
~%take from [to]	Copies the **from** file on the remote system to the **to** file on the local system. If **to** is omitted, the remote file is copied to the local system under the same file name. As each block of the file is transferred, consecutive single digits are displayed on the terminal screen.
~%put from [to]	Copies the **from** file on the local system to the **to** file on the remote system. If **to** is omitted, the local file is copied to the remote system under the same file name. As each block of the file is transferred, consecutive single digits are displayed on the terminal screen.
~~line	Sends the string denoted by ~**line** to the remote system.
~%break	Transmits a **BREAK** to the remote system. The **BREAK** can also be specified as **~%b**.
~%debug	Toggles the **-debug** flag on or off; this can also be specified as **~%d**.
~t	Prints the values of the **TERMIO** structure variables for the user's terminal. This is useful for debugging.
~l	Prints the values of the **TERMIO** structure variables for the remote communication line. This is useful for debugging.
~%nostop	Toggles between DC3/DC1 input control protocol and no input control. This is useful in case the remote system is one that does not respond properly to the DC3 and DC1 characters.

Note: As soon as the user enters **~!**,**~%**, **~$**, **~t**, or **~l**, the system displays the name of the local computer in the a format such as the following:

```
~[system_name]!/%
```

The user then enters the command to be executed on the local computer.

1.1.105.2 Additional Information

The receive process normally copies data from the remote system to th local system's standard output. Internally, the program accomplishes this by initiating an output diversion to a file when a line from the remote system begins with ~>.

Data from the remote system is diverted to **file** on the local system. The trailing ~> marks the end of the diversion.

The use of **~%put** requires **stty** and **cat** on the remote system. It also requires that the current erase and kill characters on the remote system be identical to these current control characters on the local system. Backslashes are inserted at appropriate places.

The use of **~%take** requires **echo** and **cat** on the remote system. Also, **stty tabs** mode should be set on the remote system if tabs are to be copied without expansion to spaces.

The **cu** command can be used to connect multiple systems, and commands can then be executed on any of the connected systems. For example, issue the **cu** command on system X to connect to system Y, and then issue the **cu** command on system Y to connect to system Z. System X is then the local computer, and systems Y and Z are remote computers.

The user can execute commands on system Z by logging on and issuing the command. Commands can be executed on system X by prefixing the command with a single tilde (~**cmd**), and on system Y by prefixing the command with two tildes (~~**cmd**). In general, one tilde causes the specified command to be executed on the original local computer, and two tildes cause the command to be executed on the next system on which the **cu** command was issued.

For example, once the multiple systems are connected, the user can execute the **uname** command with the **-n** flag (to display the node name) on Z, X, and Y as follows:

```
$ uname -n
Z
$ ~!uname -n
X
$ ~~!uname -n
Y
```

Notes:

1. After executing the **cu** command, the user must log in to the remote system and press **Enter** (carriage return).

2. The **cu** command does not do integrity checking on data it transfers.

3. Data fields with special **cu** command characters may not be transmitted properly.

4. Depending on the interconnection hardware, it may be necessary to use a **~.** to terminate the conversation even if the normal logout sequence has been used.

5. There is an artificial slowing of transmission by the **cu** command during the **~%put** operation so that loss of data is unlikely.

6. The exit code is 0 for normal exit, otherwise, the code is -1.

Flags

-sspeed Specifies the transmission speed (300, 1200, 2400, 4800, 9600). The default value is **Any** speed, which instructs the system to use the rate appropriate for the default (or specified) transmission line. (The order of the transmission lines is specified in the **/usr/adm/uucp/Devices** file.) Most modems operate at 300, 1200, or 2400 baud, while most hard-wired lines are set to 1200 baud or higher.

-lline Specifies a device name to use as the communication line. This flag can be used to override the search that would otherwise take place for the first available line with the right speed. When the **-l** flag is used without the **-s** flag, the speed of a line is taken from the **/usr/adm/uucp/Devices** file. When the **-l** and **-s** flags are used together, the **cu** command searches the **/usr/adm/uucp/Devices** file to check whether the requested speed is available for the specified line. If so, the connection is made at the requested speed; otherwise, an error message is printed, and the call is not made.

The specified device is generally a hard-wired asynchronous line (for example, **/dev/tty2**), in which case a telephone number (**telno**) is not required. If the specified device is associated with a modem, a telephone number must be provided. Using this flag with **system_name** rather than with **telno** does not give the desired result (see **system_name**).

Note: Under ordinary circumstances, the user should not have to specify the transmission speed or a line/device. The defaults set when BNU is installed should be sufficient. Refer to *Managing the AIX Operating System* for information about setting defaults.

-h Emulates local echo, supporting calls to other systems that expect terminals to be set to half-duplex mode.

-t Used to dial an ASCII terminal that has been set to auto answer. Appropriate mapping of carriage-return to carriage-return line-feed pairs is set.

-d Prints diagnostic traces.

-o Designates that odd parity is to be generated for data sent to the remote system.

-e Designates that even parity is to be generated for data sent to the remote system.

-n For added security, prompts the user to provide the telephone number to be dialed, rather than taking it from the command line.

Examples

1. To connect to a remote system using a system name:

```
cu hera
```

2. To dial a remote system whose telephone number is 1-201-555-1212, where dialing 9 is required to get an outside dial tone and the baud rate is 1200:

   ```
   cu -s 1200 9=12015551212
   ```

   If the speed is not specified, **Any** is the default value.

3. To log in to a system connected by a hard-wired line:

   ```
   cu -l /dev/tty2
   ```

   or

   ```
   cu -l tty2
   ```

4. To dial a remote system with the specified line and a specific speed:

   ```
   cu -s 1200  -l tty3
   ```

5. To dial a remote system using a specific line associated with a modem:

   ```
   cu -l cul4  9=12015551212
   ```

6. To copy a file from the local system to the remote system (after logging in to the remote system):

   ```
   ~%put /u/amy/file
   ```

   or

   ```
   ~%put /u/amy/file /u/amy/tmpfile
   ```

Files

/etc/locks/LCK..(tty-device)	Prevents multiple use of device.
/usr/adm/uucp/Devices	Information about available links.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.

Related Information

See the following commands: "cat" in topic 1.1.49, "ct" in topic 1.1.102, "echo" in topic 1.1.146, "stty, STTY" in topic 1.1.447, "uuname" in topic 1.1.509, and "uucp" in topic 1.1.506.

1.1.106 custboot

Purpose
Produces a customized boot diskette set.

Syntax

```
custboot ---¦
```

Note: This command is for the PS/2 only.

Description
The **custboot** command is used to create a customized boot diskette set. This allows the user to build a set of the boot diskettes with a driver that has been added to the kernel by the user, and one that is not currently on the standard AIX PS/2 boot diskette(s). The customized boot diskette set may be from 1 to 9 diskettes. The procedure to create and to use the "custboot" customized boot diskette set is described in *Installing and Customizing the AIX PS/2 Operating System*.

The procedure for building a customized boot diskette(s) using **custboot** is described below:

1. Prior to running **custboot** to create your custom boot diskette(s) make sure the following items are available:

   Standard AIX boot diskette #1
   Several formatted high density diskettes
   A copy of **aunix**
   A copy of **nvramcfg.info**
   A copy of **fs2fd**, which includes the file **/usr/sys/bin/ldminit**

2. Run **custboot** by typing **custboot**.

3. Select one of the options from the main menu:

   To select new disk drive
   To select new kernel location
   To select new sizes of build file systems
   To keep default values for above options

4. Follow the instructions provided by the **custboot** program.

5. Boot the kernel (or customized kernel) from the reboot customized boot diskette(s) as follows:

   Reboot the system with the "custboot" diskette #1
   Select the "boot from Diskette" menu item from the AIX PS/2 bootstrap diskette.

6. Follow the instructions provided by the IBM AIX PS/2 bootstrap diskette.

Files
/usr/sys/bin/ldminit
"**aunix**"
"**nvramcfg.info**"
"**fs2fd**"

Related Information

Refer to "Creating Customized Boot Diskette Set" and "Booting the Kernel from the Boot Diskettes," in the *Installing and Customizing the AIX PS/2 Operating System* manual.

1.1.107 cut

Purpose
Writes out selected fields from each line of a file.

Syntax

```
      +- -clist ----------------+   +--------+
cut ---¦         +------------+ +---¦        +---¦
      +- -flist -¦ +---------+ +-+   +- file -+
                +-¦ -dchar ¦+-+             ¦
                  ¦ -s     ¦¦          +------+
                 ¦+---------+¦
                 +-----------+
```

¦ The default **char** is a tab.

Description
The **cut** command cuts out columns from a table or fields from each line of a file and writes these columns or fields to standard output. If you do not specify a **file**, the **cut** command reads standard input.

You must specify either the **-c** or **-f** flag. The **list** parameter is a comma-separated and/or minus-separated list of integer field numbers (in increasing order). The minus separator indicates ranges. Some sample **list**s are **1,4,7**; **1-3,8**; **-5,10** (short for **1-5,10**); and **3-** (short for third through last field). The fields specified by **list** can be a fixed number of character positions, or the length can vary from line to line and be marked with a field delimiter character, such as a tab character.

You can also use the **grep** command to make horizontal cuts through a file and the **paste** command to put the files back together. To change the order of columns in a file, use the **cut** and **paste** commands.

Flags

-clist	Specifies character positions. For example, if you specify **-c1-72**, the **cut** command writes out the first 72 characters in each line of the file. There is no space between **-c** and **list**.
-dchar	Uses the specified **char**acter as the field delimiter when you specify the **-f** flag. You must quote characters with special meaning to the shell, such as the space character.
-flist	Specifies a list of fields assumed to be separated in the file by a delimiter character, by default the tab character. For example, if you specify **-f1,7**, the **cut** command writes out only the first and seventh fields of each line. If a line contains no field delimiters, the **cut** command passes them through intact (useful for table subheadings), unless you specify the **-s** flag.
-s	Suppresses lines that do not contain delimiter characters (use only with the **-f** flag).

Example
To display several fields of each line of a file:

```
  cut -f1,5 -d: /etc/passwd
```

This displays the login name and full user name fields of the system password file. These are the first and fifth fields (**-f1,5**) separated by colons (**-d:**).

So, if the **/etc/passwd** file looks like this:

```
  su:UHuj9Pgdvz0J":0:0:User with special privileges:/:/bin/sh
  daemon:*:1:1::/etc:
  bin:*:2:2::/bin:
  sys:*:3:3::/usr/src:
  adm:*:4:4:System Administrator:/usr/adm:/bin/sh
  pierre:boodwqT3irHFE:200:200:Pierre Harper:/u/pierre:/bin/sh
  joan:wijBNaYpCZuL.:202:200:Joan Brown:/u/joan:/bin/sh
```

then the **cut** command produces:

```
  su:User with special privileges
  daemon:
  bin:
  sys:
  adm:System Administrator
  pierre:Pierre Harper
  joan:Joan Brown
```

Related Information
See the following commands: "grep, egrep, fgrep" in topic 1.1.193 and "paste" in topic 1.1.313.

1.1.108 cw, checkcw

Purpose
Prepares constant-width text for **troff** files.

Syntax

```
      +-- +t --+   +--- -f3 ----+   +-----------------------+   +--------+
cw ---¦ one of +---¦ +--------+ +---¦                       +---¦        +---
      ¦ +----+ ¦   +-¦ -d     +-+   +- -l delim -- -r delim -+   +- file -+
      +-¦ +t +-+     ¦ -ffon  ¦¦                                         ¦
        ¦ -t ¦      ¦+--------+¦                                  +------+
        +----+      +----------+

           +-----------------------+
checkcw ---¦                       +-- file --¦
           +- -l delim -- -r delim -+        ¦
                                     +------+
```

Note: This command does not have MBCS support.

Description
The **cw** command preprocesses **troff file**s containing text to be typeset in the constant-width (**CW**) font. The **cw** command reads standard input if you do not specify a **file** or if you specify a - (minus) as one of the input file names. It writes its output to standard output.

Since the text that is typeset by the **cw** command resembles the output of line printers and work stations, it can be used to typeset examples of programs and computer output in user manuals and programming texts. It has been designed to be distinctive when used with the Times Roman font.

Because the **CW** font contains a "nonstandard" set of characters and because text typeset with it requires different character and inter-word spacing than is used for "standard fonts", you must use the **cw** command to preprocess documents that use the **CW** font.

The **CW** font contains the 94 printing ASCII characters:

```
  abcdefghijklmnopqrstuvwxyz
  ABCDEFGHIJKLMNOPQRSTUVWXYZ
  0123456789
  !$%&()`'*+@.,/:;=?[]¦-_^~"<>{}#\
```

plus eight non-ASCII characters represented by four-character **troff** strings (in some cases attaching these strings to "nonstandard" graphics).

Table 1-3. Non-ASCII Characters in CW Font

Character	**Symbol**	**Troff Name**
"Cents" sign	¢	\(ct
EBCDIC "not" sign	¬	\(no

Left arrow		\(<-
Right arrow		\(->
Down arrow		\(da
Vertical single quote	'	\(fm
Control-shift sign	&mee	. \(dg
Visible space sign		\(sq
Hyphen	-	\(hy
Up arrow		\(ua
Home arrow		\(lh

The **cw** command recognizes five request lines, as well as user-defined delimiters. The request lines look like **troff** macro requests. The **cw** command copies them in their entirety onto the output. Thus, you can define them as **troff** macros; in fact, the **.CW** and **.CN** macros **should** be so defined. The five requests are:

.CW Marks the start of text to be set in the **CW** font. This request causes a break. It can take the same flags (in the same format) as those available on the **cw** command line.

.CN Marks the end of text to be set in the **CW** font. This request causes a break. It can take the same flags (in the same format) as those available on the **cw** command line.

.CD Changes the delimiters and/or settings of other flags. It can take the same flags (in the same format) as those available on the **cw** command line. The purpose of this request is to allow the changing of flags other than at the beginning of a document.

.CP argument-list Concatenates all the arguments (delimited like **troff** macro arguments), with the odd-numbered arguments set in the **CW** font and the even-numbered ones in the prevailing font.

.PC argument-list Acts the same as the **.CP** request, except the even-numbered (rather than odd-numbered) arguments are set in **CW** font.

The **.CW** and **.CN** requests should bracket text that is to be typeset in the **CW** font as is. Normally, the **cw** command operates in the transparent mode. In that mode, every character between **.CW** and **.CN** request lines represents itself, except for the **.CD** request and the special four-character names listed previously. In particular, the **cw** command arranges for all periods (.) and apostrophes (') at the beginning of lines, and all backslashes (\) and ligatures (fi, ff, and so on) to be hidden from **troff**. The transparent mode can be turned off by using the **-t** flag, in which case normal **troff** rules apply. In either case, the **cw** command hides from the user the effect of the font changes generated by the **.CW** and **.CN** requests.

You can also use the **-l** and **-r** flags to define delimiters with the same function as the **.CW** and **.CN** requests. They are meant to enclose words or phrases that are to be set in **CW** font in the running text. The **cw** command treats text between delimiters as it does text bracketed by **.CW/.CN** pairs, with one exception. Spaces within **.CW/.CN** pairs have the same width as other **CW** characters, while spaces within delimited text are half as wide, so they have the same width as spaces in the prevailing text. Delimiters have no special meaning inside **.CW/.CN** pairs.

The **checkcw** command checks that left and right delimiters and the **.CW/.CN** pairs are properly balanced. It prints out all lines in the section with the unmatched delimiters.

Notes:

1. It is unwise to use . (period) or \ (backslash) as delimiter characters.

2. Certain **CW** characters do not combine well with certain Times Roman characters, for example, the spacing between a **CW &** (ampersand) followed by a Times Roman comma (,). In such cases, using **troff** half- and quarter-space requests can help.

3. The **troff** code produced by the **cw** command is difficult to read.

4. The **mm** and **mv** macro packages contain definitions of **.CW** and **.CN** macros that are adequate for most use. If you define your own, make sure that the **.CW** macro invokes the **troff** no-fill (**.nf**) mode, and the **.CN** macro restores the fill mode (**.fi**), if appropriate.

5. When set in running text, the **CW** font is meant to be set in the same point size as the rest of the text. In displayed matter, on the other hand, it can often be profitably set one point smaller than the prevailing point size. The **CW** font is sized so that, when it is set in 9-point, there are 12 characters per inch.

6. Documents that contain **CW** text may also contain tables and equations. If this is the case, the order of preprocessing must be **cw**, **tbl**, and **eqn**. Usually, the tables will not contain any **CW** text, although it is possible to have elements in the table set in the **CW** font. Care must be taken that the **cw** command does not modify the **tbl** format information. Attempts to set equations in the **CW** font are not likely to be pleasing or successful.

7. In the **CW** font, overstriking is most easily accomplished with backspaces. Because spaces (and therefore backspaces) are half as wide between delimiters as inside **.CW/.CN** pairs, two backspaces are required for each overstrike between delimiters.

Flags

-d	Displays the current flag settings on the standard error output in the form of **troff** comment lines. This flag is meant for debugging.
-ffont	Replaces **font** with the **cw** font (default=3, replacing the bold

	font). The **-f5** flag is commonly used for formatters that allow more than four simultaneous fonts. This flag is useful only on the command line.
-ldelim	Sets the left delimiter as the one- or two-character string **delim**. The left delimiter is undefined by default.
-rdelim	Set the right delimiter as **delim**. The right delimiter is undefined by default. The left and right delimiters may (but need not) be different.
-t	Turns the transparent mode **off**.
+t	Turns the transparent mode **on** (this is the default).

Files

/usr/lib/font/ftCW	CW font-width table.

Related Information

See the following commands: "eqn, neqn, checkeq" in topic 1.1.152, "mmt, mant, mvt" in topic 1.1.275, "tbl" in topic 1.1.463, and "troff" in topic 1.1.302.2.

See the **mm** and **mv** miscellaneous facilities in *AIX Operating System Technical Reference*.

1.1.109 cxref

Purpose
Creates a C program cross-reference listing.

Syntax

```
         +-------------+   +----- -w80 -----+
cxref ---¦ +---------+ +---¦      +- 80 --+ +-- file --¦
         +-¦ -c      +-+   +- -w -¦¦      +-+        ¦
           ¦ -o file ¦¦           +- num -+   +------+
          ¦¦ -s      ¦¦
          ¦¦ -t      ¦¦
          ¦+---------+¦
          +-----------+
```

¦ Do not put a space between these items.

Note: This command does not have MBCS support.

Description
The **cxref** command analyzes C program **file**s and creates a cross-reference table, using a version of the **cpp** command to include **#define** directives in its symbol table. It writes to standard output a listing of all symbols in each file processed, either separately or in combination (see the **-c** flag on page 1.1.109). When a reference to a symbol is that symbol's declaration, an * (asterisk) precedes it. You can also use the **-D**, **-I**, and **-U** flags from the **cpp** command.

Flags

-c	Displays a combined listing of the cross-references in all input files.
-o file	Directs the output to the specified **file**.
-s	Does not display the input file names.
-t	Makes the listing 80 columns wide.
-w[num]	Makes the listing **num** columns wide, where **num** is a decimal integer greater than or equal to 51. If you do not specify **num** or if **num** is less than 51, the listing will be 80 columns wide.

Related Information

See the following commands: "cc" in topic 1.1.52 and "cpp" in topic 1.1.94.

See discussion of **cxref** in *AIX Operating System Programming Tools and Interfaces*.

1.1.110 date

Purpose
Displays or sets the date.

Syntax

1.1.110.1 Operating with Superuser Authority

```
        +--------+   +------------------------+        +----------------+
date ---¦ +----+ +---¦ +---------------+      +-- mm --¦¦      +------+ +---¦
        +-¦ -n +-+   +-¦ +------+     ¦+- hh -+        +- .ss -¦¦     +-+
          ¦ -u ¦¦      +-¦     ¦+- dd -+                       +- yy -+
         ¦+----+¦        +- MM -+
         +------+

        +--------+   +---------------------------------+          +-------+
date ---¦ +----+ +---¦ +------------------------+      +--- mm ---¦       +--
        +-¦ -n +-+   +-¦ +---------------+      +- hh -+          +- .ss -+
          ¦ -  ¦¦      +-¦ +------+      +- dd -+
         ¦+----+¦        +-¦      +- MM -+
         +------+          +- yy -+
```

1.1.110.2 Operating without Superuser Authority

```
        +------+   +--------------+
date ---¦      +---¦              +---¦
        +- -u -+   +- + "string" -+
```

```
-----------------
¦ Do not put a blank between these items.
```

Description

Warning: Do not change the date while the system is running with more than one user.

If called with no flags or with a flag list that begins with a + (plus sign), the **date** command writes the current date and time to standard output. Otherwise, it sets the current date. Only a user operating with superuser authority can change the date and time. The ordering of the day and month numbers in the date specification is locale-specific. The default orders are **MMddhhmm.ssyy** and **yyMMddhhmm.ss** where:

MM is the month number
dd is the number of the day in the month
hh is the hour in the day (using a 24-hour clock)
mm is the minute number
ss is the number of seconds
yy is the last two numbers of the year.

The alternative orderings are **ddMMhhmm.ssyy**, and **yyddMMhhmm.ss.**

Notes:

1. If the format **yyMMddhhmm.ss** is specified, the value of **yy** must be 88 to 99.

2. When the **.ssyy** format is used, if **yy** is specified as less than 70, the next century is assumed. For example, 32 is interpreted as 2032.

3. When the alternate formats are used, if the month is specified, the day must also be specified.

The current month, day, hour, and year are default values. The system operates in Greenwich Mean Time (GMT). The **date** command takes care of the conversion to and from local standard and daylight time as specified in the **TZ** environmental variable. For more information see the environment miscellaneous facility in the *AIX Operating System Technical Reference*.

Note: The **date** command is needed to set the clock correctly on a PS/2 after a time change.

When the system administrator uses the **date** command to set the time or date from any machine on a TCP/IP network which is using the time daemon (**/etc/timed**), the date is reset on all machines in the network. The time daemon on one machine in a network runs as the master and all others as slaves. The master performs the task of computing clock differences and sends correction values to the slaves. If the **date** command is set on a machine in a slave state, it notifies the master of the needed change. For information on the function of time daemons, see the **timed** command in the *AIX TCP/IP Guide*.

If you follow **date** with a + and a string containing field descriptors, you can control the output of the command. You must precede each field descriptor with a percent sign (**%**). The system replaces the field descriptor with the specified value. Enter a literal **%** as **%%**. The **date** command copies any other characters to the output without change. The **date** command always ends the string with a new-line character. Output fields are fixed size (zero padded if necessary).

Flags

-n	Do not set the time globally on all machines in a local area network that have their clocks synchronized.
-u	Display/set time in GMT.

Field Descriptors

a	Displays the abbreviated day of the week (**Sun** to **Sat** or the non-English equivalent).
d	Displays the day of month (**01** to **31**).
D	Displays the date as **mm/dd/yy** (the default) or as **dd/mm/yy**. This format is locale-specific.
h	Displays the abbreviated month (**Jan** to **Dec** or the non-English equivalent).
H	Displays the hour (**00** to **23**).
j	Displays the day of year (**001** to **366**).
m	Displays the month of year (**01** to **12**).
M	Displays the minute (**00** to **59**)
n	Inserts a new-line character.
r	Displays the time in AM/PM notation (or the non-English equivalent).
S	Displays the second (**00** to **59**).
t	Inserts a tab character.
T	Displays the time as **hh:mm:ss** (the default) or as **mm:hh:ss**. This format is locale-specific.
w	Displays the day of the week numerically (Sunday = **0**).
y	Displays the last two numbers of year (**00** to **99**).

Examples

1. To display current date and time:

   ```
   date
   ```

2. To set the date and time:

```
date  02171425.45
```

This command sets the date and time to 14:25:45 (45 seconds after 2:25 p.m.) February 17 of the current year.

3. To display the date and time in a specified format:

```
date  +"%r  %a  %d  %h  %y  (Julian  Date:  %j)"
```

This command displays the date (assume current year is 1984) shown in Example 2 as:

```
02:25:03  PM  Fri  17  Feb  84  (Julian  Date:  048)
```

Files

/dev/kmem	Default system image file.

Related Information

See the **time**, **stime**, and **strftime** system calls and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See the **timed** command in the *AIX TCP/IP User's Guide*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.111 dbx, mdbx

Purpose
Provides a tool to debug and run programs under the AIX operating System.

Syntax

```
             +---------------------------------------------------------------+
dbx, mdbx ---¦ +---------------------------+   +--------------------------+ +--
             +-¦ +---------------------+ +---¦            +-------------+ +-+
               +-¦ -c file             +-+   +- objfile -¦             +-+
                 ¦ -I dir              ¦¦                +- corefile -+
                ¦¦ -r                  ¦¦
                ¦¦ -x dbx.name         ¦¦
                ¦¦ -X in,out,error,pipe ¦¦
                ¦+---------------------+¦
                +-------------------------+
```

Note: This command does not have MBCS support.

Description
The **dbx** command is the debugger for the C Language compiler. The **mdbx** command is the debugger for the Extended C Language compilers. The debuggers function the same, but must be used on the correct files. Unless otherwise specified, the following information about **dbx** refers to both debuggers.

Note: AIX assemblers are intended to support the compilers and may not have the full functionality of other assemblers specifically used for assembler language programming. Although **dbx** can disassemble compiler-generated code properly, it uses the **as** assembler, so some restrictions apply to code not generated by a compiler.

The **dbx** command provides a symbolic debugger to be used with AIX Operating System C, VS Pascal, and VS FORTRAN programs. With this command, you can examine object and core files and provide a controlled environment for running a program. You can set breakpoints at selected statements or run the program one line at a time. You can debug using symbolic variables and instruct the **dbx** command to display them in their correct format.

The **objfile** is an object (executable) file produced by a compiler. Use the **-g** (generate symbolic information) flag when you compile your program to produce the information the **dbx** command needs. The symbolic information is stored in the object file along with the executable code. If you have not compiled the object file using the **-g** flag or if it is not executable (because of compiler or loader errors), the symbolic capabilities of the **dbx** command are severely limited.

When the **dbx** command is started, it checks for the file **.dbxinit** in the user's current directory. If it cannot find the file there, it checks the user's **$HOME** directory. If the file **.dbxinit** exists, the subcommands in it are run at the beginning of the debug session. Use an editor to create a **.dbxinit** file.

If the file **core** exists in the current directory or a **corefile** is specified, you can use the **dbx** command to examine the state of the program when it faulted.

When printing variables and expressions, the static scope of the current function is first used to resolve names. If the name is not defined in the first scope, the dynamic scope is then used. If static and dynamic searches do not yield a result, an arbitrary symbol by that name is chosen, and **dbx** prints the message **[*.using module.variable]**, where **module.variable** is the name of an identifier qualified with a block name. You can override the name resolution procedure by qualifying an identifier with a block name. Source files are treated as modules named by the file name without the language suffix (such as the **.f** suffix on a FORTRAN program, the **.p** suffix on a Pascal program, or the **.c** suffix on a C Language program).

Expressions are specified with a subset of C and Pascal syntax. Indirection can be denoted by using either a prefix ***** (asterisk) or a postfix **^** (circumflex). Array expressions are subscripted by **[ ]** (brackets) or **( )** (parentheses). The field reference operator **.** (period) can be used with pointers as well as records, making the C operator &arrow. (->) unnecessary (although it is supported).

Types of expressions are checked. The type of an expression can be overridden by using **type-name (expression)**. When there is no corresponding named type, the special construct **&type-name** can be used to represent a pointer to the named type and the construct **.$$tagname** can be used to represent a pointer to a **C** language **enum**, **struct**, or **union** tag.

The following operators are valid in expressions:

Algebraic	**+**, **-**, *****, **/** (floating), **div** (integral), **mod**, **exp** (exponentiation)
Bitwise	**|**, **bitand**, **xor**, **~**, **<<**, **>>**
Logical	**or**, **and**, **not**
Comparison	**<**, **>**, **<=**, **>=**, **<>** or **!=**, **=** or **==**
Other	**sizeof**

Flags

-c file Runs the **dbx** commands in the file before reading from standard input.

-I dir Includes **dir** in the list of directories that are searched for source files. The default is to look for source files in the current directory and in the directory where the object file is located. The search path can also be set with the **use** subcommand.

-r Runs the object file immediately. If it ends successfully, exit the **dbx** command. Otherwise, enter the debugger and report the reason for termination.

-x dbxname Name of this **dbx** executable.

-X in, out, error, pipe
File descriptor for **in, out, error**, and **pipe** for an X-window environment.

Note: Unless the **-r** flag is specified, the **dbx** command prompts the user and waits for a command.

Subcommands

Subtopics

1.1.111.1 Run and Trace Subcommands

call proc (**params**)	Executes the object code associated with the named procedure or function. You can use the **print proc** (**params**) subcommand to perform the same function, but with a return code of the function printed.
catch **catch signum** **catch signame** **ignore** **ignore signum** **ignore signame**	Starts or stops trapping a signal before it is sent to the program. These subcommands are useful when a program being debugged handles signals such as interrupts. A signal may be specified by number or by a name. Signal names are case insensitive and the **SIG** prefix in names is optional. By default all signals are trapped except **SIGHUP**, **SIGCLD**, **SIGALRM**, and **SIGKILL**.
clear sline	Removes all stops at a given source line. The **sline** can be either an integer or a file name string followed by a **:** (colon) and an integer.
cont **cont signum** **cont signame**	Continues execution from the current stopping point until the program finishes or another until break point is encountered. If a signal is specified, the process continues as though it received the signal. Otherwise, the process is continued as though it had not been stopped.
delete num ...	Removes the traces and stops corresponding to the specified numbers. The numbers associated by the **dbx** command with a trace or stop can be displayed with the **status** subcommand.
delete all	Removes all active traces and stops.
goto sline	Makes the specified source line the next line to be executed. The **sline** must be in the same function as the current source line. To override this restriction, set **$unsafegoto**.
goto "filename":num	Makes the source line specified by **num** in **"filename"** the next line to be executed.
next [num]	Runs the program up to the next source line. The difference between this and the **step** subcommand is that if the line contains a call to a procedure or function, the **step** command stops at the beginning of that block, while the **next** command executes the call and stops at the source line following the call. The **next** command ignores the breakpoints set in the called procedure or function. If a number is specified, that number of

	next commands is executed.
print proc (**params**)	Executes the object code associated with the named procedure or function. You can use **call proc** (**params**) subcommand to perform the same function, but without a return code of function called.
return [proc]	Continues until a return to **proc** is executed, or until the current procedure returns if none is specified.
run [args] [< infile] [> outfile] [>> outfile] [2> errfile] [2>> errfile] [>& outerrfile] [>>& outerrfile]	
rerun [args] [< infile] [> outfile] [>> outfile] [2> errfile] [2>> errfile] [>& outerrfile] [>>& outerrfile]	Starts running the object **file**, passing **args** as command line arguments. The **<**, **>**, or **2>** redirects input, output, or standard error, respectively. The **>>** appends redirected output and **2>>** appends redirected standard error. The **>&** redirects both output and standard error to the same file. The **>>&** appends the redirected output and standard error to the same file. When the **rerun** command is used without any arguments, the previous argument list is passed.
skip num	Continues execution from the current stopping point until **num**+1 breakpoints are encountered or the program finishes.
status [> file]	Displays the currently active trace and stop commands.
step [num]	Runs one source line. If a number is specified, that number of source lines are executed.
stop if cond **stop at sline [if cond]** **stop in proc [if cond]** **stop var [in proc] [if cond]**	Stops the program when the condition is true, when the source line number is reached, when the procedure (or function) is called, or when the variable is changed. A condition can also be specified for the source line, procedure, or variable stops in which case, the program only stops if the specified condition is true. You may need to use the qualified name to get the actual variable stop. Use the **whereis** command to display the qualified name. The debugger associates numbers with each **stop** subcommand. Use the **status** subcommand to display these numbers. Use the **delete** or **clear** subcommand to turn stopping off.
trace **trace in proc [if cond]** **trace sline [if cond]** **trace proc [in proc] [if cond]** **trace expr at sline [if cond]**	

trace var [in proc] [if cond]	Prints the tracing information for the specified procedure (or function), source line, expression, or variable when the program runs. A condition may be specified. The debugger associates numbers with each **trace** subcommand. Use the **status** subcommand to display these numbers. Use the **delete** subcommand to turn tracing off.

1.1.111.2 Subcommands for Examining Program Data

assign var = expr	Assigns the value of **expr** to **var**.
assign addr = expr	Loads the value of **expr** into memory at address **addr**.

case [default]
case [mixed]
case [lower]
case [upper]
	Changes the way in which the debugger interprets symbols. The default handling of symbols is based upon the current language. Symbols are folded to lowercase unless C is the current language. This command should be used if a symbol needs to be interpreted in a way not consistent with the current language. Entering this command with no parameters displays the current case mode.

dump [proc] [> file]	Displays or puts in a file the names and values of variables in **proc**. If the procedure specified is **.** (period), all active variables are dumped. The default is the current procedure.
dump module_name	Displays the names and values of all variables in **module_name**.
print expr [,expr...]	Displays the values of the expressions.
whatis name	Prints the declaration of **name**, where **name** is a variable, procedure, or function name. In the case of multiple symbol occurrences, full qualifiers may be necessary.
where [> file]	Displays a list of the active procedures and functions.
whereis identifier	Prints the full qualification of all the symbols whose name matches **identifier**. The order in which the symbols are printed is not significant.
which identifier	Prints the full qualification of **identifier** (the outer blocks with which the identifier is associated).
up [count] **down [count]**	Moves the current function, which is used for resolving names, up or down the stack **count** levels. The default is 1.

1.1.111.3 Subcommands for Accessing Source Files

/regular expression[/]	Searches forward in the current source file for the specified pattern.
?regular expression[?]	Searches backward in the current source file for the specified pattern.
edit [file]	
edit proc	Invokes an editor with **file** or the current source file if none is specified. If a procedure (or function) **proc** is specified, the editor is invoked on the file that contains it. The default editor is **vi**. The default can be overridden by resetting the environment variable **EDITOR** to the name of the desired editor. **Note:** If the procedure has the same name as any other file in the same directory, the editor invokes the other file, not the file that contains the procedure.
file [file]	Changes the current source file to **file**. If none is specified, prints the name of the current source file.
func [proc]	Changes the current function to the specified procedure or function. If none is specified, prints the current function. Changing the current function implicitly changes the current source file to the one that contains the function; it also changes the current scope used for name resolution.
list [proc]	If the name of a procedure or function is given, lists lines **f-n** to **f+m** where **f** is the first statement in the procedure or function, **n** is a small number, and **m** is the number of lines remaining that fit in the default list window. Use **set $listwindow=value** to set or change the number of lines displayed in the list window.
list [sline [, sline]]	Lists the source lines in the current source file from the first line number to the second inclusive. If no lines are specified, lists the next 10 lines or **$listwindow** lines. An **sline** of **$** specifies the current line of execution. An **sline** of @ specifies the next line to be listed. Each source line can also be specified as **sline+num** or **sline-num**.
move sline	Changes the next line to be displayed to **sline**.
listi [proc]	Lists instructions from the specified procedure or function. The number of instructions displayed is controlled by the

	$listwindow value.
listi at sline	Lists instructions beginning with the source line specified.
listi [address [, address]]	Lists instructions from the first address to the second address inclusive. If no lines are specified, the next **$listwindow** instructions are listed.
use dir [**dir** ...]	Sets the list of directories to be searched when looking for source files.

1.1.111.4 Machine Level Subcommands

address,address/[mode][>file]

address/[count][mode][>file] Prints the contents of memory starting at the first **address** and continuing up to the second **address** or until **count** items are printed. If **address** is **.** (period), the address following the one printed most recently is used. The **mode** specifies how memory is to be printed; if it is omitted, the previous mode specified is used. The initial mode is **X**. The following modes are supported:

b	Prints a byte in octal.
c	Prints a byte as a character.
d	Prints a short word in decimal.
D	Prints a long word in decimal.
f	Prints a single-precision real number.
g	Prints a double-precision real number.
h	Prints a byte in hexadecimal.
i	Prints the machine instruction.
o	Prints a short word in octal.
O	Prints a long word in octal.
s	Prints a string of characters terminated by a null byte.
x	Prints a short word in hexadecimal.
X	Prints a long word in hexadecimal.

Symbolic addresses are specified by preceding the name with an **&**. Addresses can be expressions made up of other addresses and the operators **+**, **-**, and ***** (indirection). Also, any expression enclosed in parentheses is interpreted as an address.

cleari addr Removes all breakpoints at a specified address.

gotoi addr Changes the program counter to **addr**.

registers [**>file**] Displays the values of all general-purpose registers, system-control registers, floating-point registers, and the current instruction register. General-purpose registers are denoted by **$name**, where **name** is the extended register name. Floating point registers are denoted by **$frn**.

stepi [num]

nexti [num] Runs a single step as in **step** or **next**, but runs a single instruction rather than source line. If **num** is specified, a single step is run that number of times.

tracei [addr or var][if cond]

stopi [addr or var][if cond] Traces or sets a stop when the contents of **addr** change.

tracei [at addr][if cond]

stopi at addr [if cond] Turns on tracing or sets a stop at a machine instruction address.

stopi if cond [at addr] Turns on tracing or sets a stop if **cond** is true.

1.1.111.5 Subcommand Aliases and Variables

alias	Displays aliases for subcommands.
alias name name	
alias name "string"	
alias name (params) "string"	When subcommands are processed, the **dbx** command first checks to see if the word is an alias for either a subcommand or a string. If it is an alias, the **dbx** command treats the input as though the corresponding string (with values substituted for any parameters) had been entered.
set var [= expr]	The **set** subcommand defines a value (expression) for a debugger variable. The name of the variable cannot conflict with names in the program being debugged. A variable is expanded to the corresponding expression within other commands. The following variables have a special meaning:

$frame [=address]	Setting this variable to **address** causes **dbx** to use the stack frame pointed to by **address** for doing stack traces and accessing local variables. This facility is of particular use for kernel debugging.
$expandunions	Setting this variable causes the **dbx** command to display values of each part of unions or variant records.
$hexchars	
$hexints	
$hexstrings	When set, the **dbx** command prints characters, integers, or character pointers, respectively, in hexadecimal.
$hexin	When set, the **dbx** command interprets integers as hexadecimal.
$listwindow[=n]	The value of this variable specifies the number of lines, **n**, to list around a function or when the list command is given without any parameters. Its default value is 10.

$mapaddrs	Setting (unsetting) this variable causes the **dbx** command to start (stop) mapping addresses. As with **$frame**, this variable is useful for kernel debugging.
$octin	When set, the **dbx** command interprets integers as octal.
$octints	When set, the **dbx** command displays integers in octal.
$noargs	When set, the **dbx** command omits the arguments from commands that walk the stack (**where**, **up**, **down**, **dump**).
$noflargs	When set, the **dbx** command does not display floating-point registers from the **registers** command.

Note: The following variables should be used only with great care because they severely limit the usefulness of the **dbx** command for detecting errors.

$unsafeassign	When **$unsafeassign** is set, strict type checking between the two sides of an assign statement is turned off.
$unsafebounds	When set, subscript checking on arrays is disabled.
$unsafecall	When **$unsafecall** is set, strict type checking is turned off for arguments to subroutine or function calls.
$unsafegoto	When set, **goto** destination checking is disabled.

unalias name	Removes the alias with the given **name**.
unset name	Deletes the debugger variable associated with **name**.

1.1.111.6 Vector Subcommands (mdbx Only)

vregisters [<vreg number>]	Prints the contents of all vector registers or the register designated by <**vreg number**>. The format of the display is controlled by internal debugger variables, as follows: If the **$intvectors** variable is set, the contents of the vector elements are displayed as integers. If the **$doublevectors** variable is set, the contents of the vector elements are displayed as double-precision floating point numbers. When specifying **vregisters** with a single register, the <**vreg number**> parameter must be even. These debugger variables are mutually exclusive. If one is set, the other is automatically cleared.
vstate	Prints the contents of the vector status register, vector mask register, and vector activity-count register.

1.1.111.7 Debugger Variables (mdbx Only)

The variables below are special variables known to the debugger. They may be displayed with the **print** subcommand or modified with the **assign** subcommand.

vrN	References vector register **N** (where 0 <= **N** <= 31) as an integer. The reference must be further qualified to specify a particular component of the register. For example, **$vr5[13]** references the fourteenth word of the sixth vector register (vector registers and components within registers are numbered starting with 0).
$vfrN	References vector register **N** (where 0 <= **N** <= 31) as a single-precision floating point number. A particular component within the register must be specified.
$vdrN	References vector register N (where 0 <= **N** <= 30, even) as a double-precision floating point number. A particular component within the register must be specified.
$vstatus	References the vector status register. The display appears as two 32-bit words in hexadecimal format. The two halves of the register can be referenced separately using the subscript notation.
$vac	References the vector activity-count register. The display appears as two 32-bit words in hexadecimal format. The two halves of the register can be referenced separately using the subscript notation.
$vmask	References the vector mask register. The display appears as a number of 32-bit words displayed in hexadecimal format. Individual words can be referenced separately using the subscript notation. The number of words displayed depends on the current section size. For example, if the section size is 128, four 32-bit words are displayed to show 128 bits, one bit per vector element.
vcount	Displays the vector count field of the vector status register in decimal. The field is a 16-bit unsigned binary integer.
$vmaskflag	Displays the vector mask mode bit of the vector status register.
$vstate	Controls whether status information from the vector processor is displayed as part of the output of the **registers** subcommand. If **$vstate** is set, the contents of the vector status register, the vector activity-count register, and the vector mask register (in addition to the normal output of the **registers** command) is displayed in hexadecimal. (The **registers** subcommand should not be confused with the **vregistsers** subcommand, which applies strictly to the vector processor. The **registers** subcommand displays the values of general purpose registers and system control registers.)

1.1.111.8 Other Useful Subcommands

help	Prints a synopsis of common **dbx** commands.
prompt [string]	Changes the **dbx** command prompt to **string**. Without the **string** argument, the **prompt** subcommand displays the current prompt.
quit	Quits the **dbx** command.
screen	Opens a virtual terminal for **dbx** command interaction. The user continues to operate in the window in which the process originated. Also, the user must be using a console that supports virtual terminals.
sh command	Passes the command line to the shell for execution. The **SHELL** environment variable determines which shell is used.
source file	Reads **dbx** commands from the given file.

Files

a.out	Contains object code; object file.
core	Contains core dump information.
.dbxinit	Contains initial commands.

Related Information
See the following command: "cc" in topic 1.1.52.

See the **a.out** and **core** files in *AIX Operating System Technical Reference*.

See "Debugging Programs" in *AIX Operating System Programming Tools and Interfaces*.

See the compiler chapters in *AIX Operating System VS FORTRAN, VS Pascal* and *C User's Guides*.

1.1.112 dc

Purpose
Provides an interactive desk calculator for doing arbitrary-precision integer arithmetic.

Syntax

```
      +--------+
dc ---¦        +---¦
      +- file -+
```

Description
The **dc** command is an arbitrary precision arithmetic calculator. This command takes its input from **file** or standard input until it reads an end-of-file character. It then writes to standard output. The **dc** command operates on decimal integers, but you may specify an input base, output base, and a number of fractional digits to be maintained. The command is structured overall as a stacking, reverse Polish calculator.

The **bc** command (see page 1.1.44) is a preprocessor for the **dc** command. It provides infix notation and a syntax similar to the C language, which implements functions and reasonable control structures for programs.

Subcommands

number	Pushes the specified value onto the stack. A **number** is an unbroken string of the digits 0-9. To specify a negative number, precede it with _ (underscore). A number may contain a decimal point.
+ - / * % ^	Adds (+), subtracts (-), multiplies (*), divides (/), remainders (%), or exponentiates (^) the top two values on the stack. The **dc** command pops the top two entries off the stack and pushes the result on the stack in their place. The command ignores fractional parts of an exponent.
sx	Pops the top of the stack and stores it in a register named **x**, where **x** may be any character.
Sx	Treats **x** as a stack. It pops the top of the main stack and pushes that value onto stack **x**.
lx	Pushes the value in register **x** on the stack. The register **x** is not changed. All registers start with zero value.
Lx	Treats **x** as a stack and pops its top value onto the main stack.
d	Duplicates the top value on the stack.
p	Displays the top value on the stack. The top value remains unchanged. The **p** interprets the top of the stack as an ASCII string, removes it, and displays it.
P	Interprets the top of the stack as a string, removes it, and displays it.

f	Displays all values on the stack.
q	Exits the program. If the **dc** command is executing a string, it pops the recursion level by two.
Q	Pops the top value on the stack and the string execution level by that value.
x	Treats the top element of the stack as a character string and executes it as a string of **dc** commands.
X	Replaces the number on the top of the stack with its scale factor.
[string]	Puts the bracketed **string** onto the top of the stack.
<x **>x** **=x**	Pops the top two elements of the stack and compares them. Evaluates register **x** as if it obeys the stated relation.
v	Replaces the top element on the stack by its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.
!	Interprets the rest of the line as an AIX command.
c	Cleans the stack: by popping all values on the stack.
i	Pops the top value on the stack and uses that value as the number radix for further input.
I	Pushes the input base on the top of the stack.
o	Pops the top value on the stack and uses that value as the number radix for further output.
O	Pushes the output base on the top of the stack.
k	Pops the top of the stack, and uses that value as a non-negative scale factor. The appropriate number of places displays on output and is maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base is reasonable if all are changed together.
K	Pushes the non-negative scale pointer on the top of the stack.
Y	Displays addresses of all variables pushed onto the stack.
z	Pushes the number of elements in the stack onto the stack.
Z	Replaces the top number in the stack with the number

of digits in that number.

?	Gets and runs a line of input.
;:	The **dc** command uses these characters for array operations.

Examples

1. To use **dc** as a calculator:

   Enter:

   ```
   1 4 / p
   ```

   The system responds:

   ```
   0
   ```

   Enter:

   ```
   1 k       [ Keep 1 decimal place  ]s.
   1 4 / p
   ```

   The system responds:

   ```
   0.2
   ```

   Enter:

   ```
   3 k       [ Keep 3 decimal places ]s.
   1 4 / p
   ```

   The system responds:

   ```
   0.250
   ```

   Enter:

   ```
   16 63 5 / + p
   ```

   The system responds:

   ```
   28.600
   ```

   Enter:

   ```
   16 63 5 + / p
   ```

   The system responds:

   ```
   0.235
   ```

   You may type the comments, which are enclosed in **[ ]s.**, but they are provided only for your information.

   When you enter **dc** expressions directly from the keyboard, press **Ctrl-D** to end the **bc** session and return to the shell command line.

2. To load and run a **dc** program file:

Enter:

```
dc prog.dc
5 lf x p   [  5 factorial ]s.
```

The system responds:

```
120
```

Enter:

```
10 lf x p  [ 10 factorial ]s.
```

The system responds:

```
3628800
```

These commands interpret the **dc** program saved in the file **prog.dc**, then read from the work station keyboard. The **lf x** evaluates the function stored in register **f**, which could be defined in the program file **prog.dc** as:

```
[ f: compute the factorial of n ]s.
[    (n = the top of the stack) ]s.

[ If 1>n do b;  If 1<n do r ]s.
   [d 1 >b d 1 <r] sf

[ Return f(n) = 1           ]s.
   [d - 1 +] sb

[ Return f(n) = n * f(n-1)  ]s.
   [d 1 - lf x *] sr
```

You can create **dc** program files with a text editor or with the **-c** (compile) flag of the **bc** command. When you enter **dc** expressions directly from the keyboard, press **Ctrl-D** to end the **bc** session and return to the shell command line.

Related Information

See the following command: "bc" in topic 1.1.36.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.113 dcopy

Purpose
Copies file systems for the best access time.

Syntax

```
         +----------------+   +-- -a7 --+   +-------------------+
dcopy ---¦ +------------+ +---¦         +---+-     -ffsize    ¦-+-- oldfs --
         +-¦ -d         +-+   +- -anum -+   +- -ffsize : isize -+
           ¦ -v         ¦¦
          ¦¦ -sCyl:skip ¦¦
          ¦+------------+¦
          +--------------+
```

¦ If not specified, the values from **oldfs** are used.

Description
Warning: **oldfs** and **newfs** must not refer to the same minidisk. Doing so will destroy the old file system.

The **dcopy** command copies an existing file system **oldfs** to a new file system **newfs**, appropriately sized to hold the reorganized results. For best results, **oldfs** should be the raw device, and **newfs** should be the block device. If **oldfs** or **newfs** is a file system name, **dcopy** uses the corresponding block device given in **/etc/filesystems**. You should run **dcopy** on unmounted file systems (in the case of the root file system, copy to a new minidisk).

If you do not specify any flags, **dcopy** copies files from **oldfs**, compressing directories by removing vacant entries and spacing consecutive blocks in a file by the optimal rotational gap.

The **dcopy** command makes **newfs** identical to **oldfs** and preserves the pack and volume labels. Thus, to compress a file system without moving it, use the **dcopy** command to copy the file to another file system and the **dd** command to copy the file back.

The **dcopy** command catches **INTERRUPT** and **QUIT** signals and reports on its progress. To end **dcopy**, send a **QUIT** signal (**Ctrl-**) and **dcopy** no longer catches **INTERRUPT** or **QUIT**. **dcopy** also attempts to modify its command line arguments so that its progress can be monitored with the **ps** command.

Note: If the Transparent Computing Facility is installed and **dcopy** is run on a non-primary copy of a replicated file system, **dcopy** will not perform directory compression. Directory compression on replicated file systems is accomplished by running **dcopy** on the primary copy of the file system and allowing the modified directories to propagate to the other copies of the file system in the normal manner. Also, when **dcopy** is performed on a replicated file system the original inode numbers are preserved to avoid unnecessary propagation. On a non-replicated file system inode numbers are changed to use the lowest set of inodes.

Warning: When run on a mounted primary copy of a replicated file system, it is important to guarantee that no updates to the file system are made while the **dcopy** operation is in progress or until the copied file system replaces the original. Failure to insure that updates do not occur can result in inconsistencies between the new primary copy of the file system and its other non-primary copies. For this reason, it is highly advised that a **dcopy** on a primary copy of a file system that cannot be unmounted be done while the system is in single-user state with cluster communications disabled (see the **clusterstop** command) and absence of updates can be reasonably guaranteed.

Flags

-anum	Places files not accessed in the specified number of days after the free blocks of the destination file system. The default value of **num** is 7. If you do not specify **num**, no files are moved.
-d	Leaves the order of directory entries as is. If you do not specify this flag, **dcopy** moves subdirectories to the beginning of directories.
-ffsize[:isize]	Specifies the file system and inode list sizes (in blocks). If not specified, the value from **oldfs** is used.
-scyl:skip	Supplies device information for creating the best organization of blocks in a file, where **cyl** is the number of block per cylinder and **skip** is the number of blocks to skip.
-v	Reports how many files were processed and how big the source and destination free lists are.

Related Information
See the following commands: "fsck, dfsck" in topic 1.1.177 and "mkfs" in topic 1.1.269.

1.1.114 dd

Purpose
Converts and copies a file.

Syntax

```
                             +- ibs=512 -+    +- obs=512 -+
      +-----------------+   +-¦           +---¦           +-+
dd ---¦ +------------+ +---¦ +- ibs=num -+    +- obs=num -+ +---
      +-¦¦           +-+    +----------- bs=num ------------+
         ¦ if=infile  ¦¦
        ¦¦ of=outfile ¦¦
        ¦¦ cbs=num    ¦¦
        ¦¦ fskip=num  ¦¦
        ¦¦ skip=num   ¦¦
        ¦¦ seek=num   ¦¦
        ¦¦ count=num  ¦¦
        ¦+------------+¦
        +--------------+

    +------------------------------+
 ---¦        4+------------------+ +---¦
    +- conv= -¦ ascii ²  unblock +-+
               ¦ ebcdic   lcase ¦ ¦¦
              ¦¦ iblock   ucase   ¦¦
              ¦¦ ibm      swab    ¦¦
              ¦¦ oblock   noerror ¦¦
              ¦¦ block    sync    ¦¦
              ¦¦ tonls    fromnls ¦¦
              ¦¦ ibm      flatten ¦¦
              ¦+------------------+¦
              ¦          4         ¦
              +-------- , ---------+
```

¦ **infile** and **outfile** default to standard input and standard output.
[2] Use only one of **ascii** and **ebcdic**.
¦ Use only one of **lcase** and **ucase**.
4 Do not use blanks around equal sign and commas.

Description

Note: If you are using a system that supports a multibyte character set, use the **iconv** command (see page 1.1.204) to convert data encoded in one file code into another file code.

The **dd** command reads the specified **infile** or standard input, does the specified conversions, and copies it to the specified **outfile** or standard output. The input and output block size may be specified to take advantage of raw physical I/O. The terms ***block*** and ***record*** refer to the quantity of data read or written by **dd** in one operation and are not necessarily the same size as a disk block.

Where sizes are specified, a number of bytes is expected. A number may end with **w**, **b**, or **k** to specify multiplication by 2, 512, or 1024 respectively; a pair of numbers can be separated by an **x** to indicate a product.

The conversion requested by **conv=fromnls** translates each extended character in a text file to a printable ASCII escape sequence that uniquely identifies the extended character. The complementary conversion, provided by **conv=tonls**, translates ASCII escape sequences to the corresponding extended character. The conversion requested by **conv=flatten** translates an extended character to the single ASCII character most resembling it in appearance or to a **?** (question mark) if no ASCII characters resemble that extended character.

The character set mappings associated with **conv=ASCII** and **conv=ebcdic** are complementary operations, described in the **ebcdic** file in *AIX Operating System Technical Reference*. These attempt to map between ASCII and the subset of EBCDIC that is found on most terminals and keypunches.

The **cbs** specification is used if one of the following conversions is specified: **ASCII**, **unblock**, **ebcdic**, **ibm**, or **block**. For the first two conversions, **dd** places characters in a conversion buffer of size **cbs**, converts these characters to ASCII, trims trailing blanks and adds new-line characters before sending data specified output. For the latter three cases, **dd** places ASCII characters in the conversion buffer, converts these characters to EBCDIC, adds trailing blanks to create records of size **cbs**.

After it finishes, **dd** reports the number of whole and partial input and output blocks.

Notes:

1. Normally, you need only write access to the output file. However, when the output file is not on a direct access device and you use the **seek** parameter, you also need read access to the file.

2. The **dd** command inserts new-line characters only when converting to ASCII; it pads only when converting to EBCDIC.

3. Use the **backup**, **tar**, or **cpio** commands instead of the **dd** command whenever possible to copy files to tape. These commands are designed for use with tape devices.

4. If you need to use **dd** to copy to a streaming tape and the data is an odd length (not a multiple of 512 bytes), you must use the **conv=sync** option to fill the last record. Streaming tape devices permit only multiples of 512 bytes.

5. When using **dd** to copy to tape, consult the hardware documentation for that tape drive to determind block size.

6. Compared to disk drives, tape drives generally require a smaller block size.

Parameters

if=infile	Specifies the input file name; standard input is the default.
of=outfile	Specifies the output file name; standard output is the default.
ibs=num	Specifies the input block size in bytes; the default is 512.
obs=num	Specifies the output block size in bytes; the default is 512.

bs=num	Specifies both the input and output block size, superseding **ibs** and **obs**.
cbs=num	Specifies the conversion buffer size.
skip=num	Skip **num** input records before starting copy.
seek=num	Seek to the **num**th record from the beginning of output file before copying.
fskip=num	Skip past **num** end-of-file characters before starting copy; this parameter is useful for positioning on multi-file magnetic tapes.
count=num	Copies only **num** input blocks. The default block size is 512 bytes (see the **ibs** parameter).

conv=spec[,spec...]
Specifies one of the following conversions:

ascii	Converts EBCDIC to ASCII.
ebcdic	Converts ASCII to EBCDIC.
ibm	Slightly different map of ASCII to EBCDIC
block	Convert variable length records to fixed length
unblock	Convert fixed length records to variable length
tonls	Converts ASCII escape sequences to extended characters.
fromnls	Converts extended characters to ASCII escape sequences.
flatten	Converts extended characters to the ASCII character most resembling it or to a **?** (question mark).

And one or more of the following conversions:

iblock **oblock**	Minimizes data loss resulting from a read or write error on direct access devices. If you specify **iblock** and an error occurs during a block read (where the block size is 512 or the size specified by **ibs=num**), **dd** attempts to reread the data block in smaller size units. If **dd** can determine the sector size of the input device, it reads the bad record one sector at a time. Otherwise, it reads it 512 bytes at a time. The input block size (**ibs**) must be a multiple of this "retry size". This allows you to maximize disk input efficiency while ensuring that data loss associated with a read error is confined to a single sector. The **oblock** conversion works similarly on output.
lcase **ucase**	Makes all alphabetic characters lowercase or uppercase.
swab	Swaps every pair of bytes.
noerror	Does not stop processing on an error.
sync	Pads every input record to **ibs**.
... , ...	Several comma-separated conversions.

Example

1. To convert an ASCII text file to EBCDIC:

```
dd if=text.ascii of=text.ebcdic conv=ebcdic
```

This converts **text.ASCII** to EBCDIC representation, storing the EBCDIC version in **text.ebcdic**.

Note: When you specify **conv=ebcdic**, **dd** converts the ASCII ^ (circumflex) character to an unused EBCDIC character (9A hexadecimal), and ASCII ~ (tilde) to EBCDIC ¬ (NOT symbol).

2. To use **dd** as a filter:

```
li -l ¦ dd conv=ucase
```

   This displays a long listing of the current directory (**li -l**) in uppercase.

3. To copy an XA370 file system on the first partition to a 3480 cartridge tape with a maximum block size:

```
dd if=/dev/chd00033 of=/dev/rmt0rh bs=65535
```

Related Information

See the following commands: "cp, copy" in topic 1.1.91, "tr" in topic 1.1.479 and "iconv" in topic 1.1.204.

See the **ebcdic** and **tape** files in *AIX Operating System Technical Reference.*

1.1.115 defkey

Purpose
Redefines keyboard keys.

Syntax

```
          +------+   +--------+
defkey ---¦      +---¦        +---¦
          +- -? -+   +- file -+
```

Note: This command is for the PS/2 only.

Description

The **defkey** command lets you redefine the keyboard keys on the active virtual terminal. Input to the **defkey** command comes either interactively from the keyboard or from a redirected file. Key assignments can be a single character, non-spacing characters, or strings.

If you specify a **file** that does not exist, the **defkey** command creates and opens the file; if **file** exists, the command opens the file. It then displays a menu that prompts you for input. This **file** can then be used as redirected input to the **defkey** command.

Flags

-? Provides help information. Since -? is treated as a wild card character in the shells, this argument should be quoted, for example:

```
defkey -\?
```

Examples

1. To redefine a key or keys and to create or add to a keyboard definition file:

   ```
   defkey mykeys
   ```

   This command creates the file **mykeys** and prompts for input. When the **defkey** command ends, the keys that you specify are redefined on the active virtual terminal. You can also use the file **mykeys** to redefine the keyboard on another virtual terminal with the command:

   ```
   defkey  < mykeys
   ```

2. To interactively redefine one or more keyboard keys for the active virtual terminal:

   ```
   defkey
   ```

Related Information

See the **hft** and **display** symbols sections of *AIX Operating System Technical Reference*.

1.1.116 del

Purpose
Deletes files.

Syntax

```
       +-----+
del ---¦     +-- file --¦
       +- - -+        ¦
               +------+
```

Description

The **del** command is provided in AIX for compatibility with other systems. A similar function is provided by "rm i". The **del** command displays the list of specified **file** names and asks you to confirm your request to delete the group of files. To delete the files, press the **Enter** key or enter a line beginning with **y**.

Since pressing the **Enter** key by itself is the same as answering "yes", be careful not to delete files accidentally. Any other response specifies **no** (do not delete the files).

The **del** command does not delete directories. See "rmdir" in topic 1.1.378 for information about deleting directories.

Warning: The **del** command ignores file protection, allowing the owner of a file to delete a write-protected file. To to delete a file, you must have write permission in the directory that the file exists in.

Flag

- Requests confirmation to delete each specified **file** rather than to delete the entire group.

Examples

1. To delete a file:

   ```
   del  chap1.bak
   ```

   This command displays the message:

   ```
   delete chap1.bak? (y)
   ```

   asking you to confirm deletion **chap1.bak**. To delete the file, press the **Enter** key or enter **y**.

2. To use the **del** command with pattern-matching characters:

   ```
   del  *.bak
   ```

   Before passing the command line to the **del** command, the shell replaces the pattern ***.bak** with the names of all the files in the current directory that end with **.bak**. (This is known as ***file-name expansion***.) The **del** command asks you to confirm your request before deleting a group of files.

3. To interactively select files to be deleted:

```
del  -  *
```

   This command displays one at a time, the name of each file in the current directory, allowing you to select the files you want to delete.

Related Information

See "rmdir" in topic 1.1.378 and "rm, delete" in topic 1.1.375.

1.1.117 delta

Purpose
Creates a delta in a Source Code Control System (SCCS) file.

Syntax

```
                            +-----------------------------------+
                          +-¦ +- -y --------+   +- -m -------+ +-- file --+
         +------------+   ¦ +-¦             +---¦            +-+        ¦ ¦
delta ---¦ +--------+ +---¦   +- -ycomment -+   +- -mmrlist -+   +------+ +---
         +-¦ -glist +-+   ¦ +- -y --------+   +- -m -------+              ¦
           ¦ -n     ¦¦  +-¦             +---¦            +----- - ------+
          ¦¦ -p     ¦¦    +- -ycomment -+   +- -mmrlist -+
          ¦¦ -rSID  ¦¦
          ¦¦ -s     ¦¦
          ¦+--------+¦
          +----------+
```

Description
The **delta** command is used to introduce into the named Source Code Control System (**SCCS**) **file** any changes that were made to the file version retrieved by a **get -e** command.

The **delta** command reads the **g-files** that correspond to the specified **file**s (see "SCCS Files" in topic 1.1.186.1) and creates a new delta.

If you specify a directory in place of **file**, the **delta** command performs the requested actions on all SCCS files within that directory (that is, on all files with the **s.** prefix). If you specify a **-** (minus) in place of **file**, the **delta** command reads standard input and interprets each line as the name of an SCCS file. When the command reads standard input, you must supply the **-y** flag. You must also supply the **-m** flag if the **v** header flag is set. (For more information on header flags, see the discussion under the **admin** command on page 1.1.16.3.) The **delta** command reads standard input until it reaches END OF FILE (**Ctrl-D**).

If you are not familiar with the delta numbering system, see *AIX Operating System Programming Tools and Interfaces* for more information.

Note: The SOH ASCII character (binary 001) has special meaning to SCCS. Therefore, lines beginning with an SOH ASCII character (binary 001) cannot be placed in the SCCS file unless the SOH is quoted using a \ (backslash). See the **sccsfile** file in *AIX Operating System Technical Reference*.

A **get** of many SCCS files, followed by **delta** of those files, should be avoided when the **get** command generates a large amount of data. Instead, you should alternate the use of the **get** and **delta** commands.

Flags

-glist Specifies a list of **SID**s (deltas) that are to be ignored when the **get** command creates the **g-file**. After you use this flag, the **get** command ignores this SID if it is one that it should not include when it builds the **g-file**.

-m[mrlist] If the SCCS file has the **v** header flag set, then a

Modification Request (MR) number must be supplied as the reason for creating the new delta.

If you do not specify the **-m** flag and the **v** header flag is set, the **delta** command reads MRs from the standard input. If standard input is a work station, the **delta** command prompts you for the MRs. The command continues to take input until it reads END OF FILE (**Ctrl-D**). It always reads MRs before reading the comments (see the **-y** flag). You can use blanks, tab characters, or both to separate MRs in a list.

If the **v** header flag has a value, it is interpreted as the name of a program that validates the MR numbers. If the **delta** command returns a nonzero exit value from the MR validation program, the **delta** command assumes some of the MR numbers are invalid and stops running.

-n Retains the **g-file**, which is normally removed at completion of **delta** command processing.

-p Writes to standard output (in the format of the **diff** command) the SCCS file differences before and after the delta is applied. See "diff" in topic 1.1.124 for an explanation of the format.

-rSID Specifies which delta is to be made to the SCCS file. You must use this flag only if two or more outstanding **get -e** commands were run on the same SCCS file by the same person. The **SID** can be either the SID specified on the **get** command line or the SID to be made, as reported by the **get** command (see Table 1-4 in topic 1.1.186.2 for additional information). An error results if the specified SID cannot be uniquely identified, or if an SID must be specified but is not.

-s Suppresses the information normally written to standard output on normal completion of the **delta** command.

-y[comment] Specifies text used to describe the reason for making the delta. A null string is considered a valid **comment**. If your comment line includes special characters or blanks, the line must be enclosed in single or double quotation marks.

If you do not specify the **-y** flag, the **delta** command reads comments from standard input until it reads a blank line or END OF FILE (**Ctrl-D**). If input is from the keyboard, the **delta** command prompts for the comments. If the last character of a line is a backslash, it is ignored. Comments must be no longer than 512 characters. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the 512-character limit may be reduced by as much as 50%, depending on the character code set being used.

Example

To record changes you have made to an SCCS file:

```
  delta  s.prog.c
```

This command adds a delta to the SCCS file **s.prog.c**, recording the changes made by editing the file **s.prog.c**. The **delta** command then asks you for a comment that summarizes the changes you make. Enter the comment, then press END OF FILE (**Ctrl-D**), or press the **Enter** key twice to indicate that you have finished the comment.

Related Information

See the following commands: "admin" in topic 1.1.16, "bdiff" in topic 1.1.37, "cdc" in topic 1.1.54, "get" in topic 1.1.186, "sccshelp" in topic 1.1.411, "prs" in topic 1.1.336, and "rmdel" in topic 1.1.377.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.118 deroff

Purpose
Removes **nroff, troff, tbl**, and **eqn** constructs from files.

Syntax

```
          +---------+   +--------+   +--------+   +------------+
deroff ---¦ one of  +---¦ one of +---¦ +----+ +---¦            +---¦
          ¦ +-----+ ¦   ¦ +----+ ¦   +-¦ -k +-+   +--- file ---+
          +-¦ -ma +-+   +-¦ -i +-+     ¦ -p ¦               ¦
            ¦ -me ¦       ¦ -l ¦       ¦ -u ¦       +---------+
            ¦ -ml ¦       +----+       ¦ -w ¦
            ¦ -mm ¦                    +----+
            ¦ -ms ¦
            +-----+
```

Note: This command does not have MBCS support.

Description
The **deroff** command reads the specified files (standard input, by default); removes all **troff** requests, macro calls, backslash constructs, **eqn** constructs (between **.EQ** and **.EN** lines and between delimiters), and **tbl** descriptions; and writes the remainder of the file to standard output.

The **deroff** command normally follows chains of included files (**.so** and **.nx troff** commands). If a file has already been included, a **.so** command naming the file is ignored and a **.nx** command naming the file ends execution. Note that the **deroff** command is not a complete **troff** interpreter and can be confused by subtle constructs. Most errors result in too much rather than too little output.

Flags

-i	Suppresses processing of included files.
-k	Keeps (that is, does not remove) a block of text. The default is to remove blocks of text (for example, **.ne** constructs).
-l	Suppresses processing of included files whose names begin with **/usr/lib**, such as macro files in the directory **/usr/lib/tmac**.
-ma	Remove ma (man) macros from text so that only running text is output.
-me	Removes me macros from text so that only running text is output (default macro).
-ml	Removes mm macros from text and also deletes mm list structures. The **-ml** flag does not handle nested lists.
-mm	Removes mm and ms macros from text so that only running text is output (no text from macro lines is included).
-ms	Removes ms macros in text so that only running text is output.
-p	Processes special paragraphs.
-u	Removes ASCII underline and boldface control sequences (_\b and \b).

The **-u** flag automatically sets the **-w** flag.

-w Makes the output a word list, with one word per line and all other characters deleted. In text, a word is defined as any string containing at least two letters, composed of letters, digits, ampersands (**&**), and apostrophes ('). In a macro call, a word is defined as a string beginning with at least two letters and containing at least three letters. Delimiters are any characters other than letters, digits, apostrophes, and ampersands. Trailing apostrophes and ampersands are removed from words.

Related Information

See the following commands: "eqn, neqn, checkeq" in topic 1.1.152, "nroff, troff" in topic 1.1.301, and "tbl" in topic 1.1.463.

1.1.119 devices

Purpose
Adds, deletes, changes, and displays device information.

Syntax

```
devices ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **devices** command lets you add, delete, change, or examine information about devices on the system. To use the **devices** command, you must be a member of the system group or have superuser authority.

The **devices** command is an interactive, menu-driven program. See *Installing and Customizing the AIX Operating System*.

Files

/etc/filesystems	Descriptions of mountable file systems.
/etc/predefined	Listing of all devices available to the system.
/etc/master	Default master configuration file.
/etc/system	Default system configuration file.
/etc/ports	System terminal characteristics file.
/etc/qconfig	Printer configuration file.

Related Information

In the *AIX Commands Reference*, see "pstart, penable, pshare, pdelay" in topic 1.1.338.

See the discussion of **devices** in *Installing and Customizing the AIX Operating System*.

1.1.120 devnm

Purpose
Names a device.

Syntax

```
devnm -- path --¦
                ¦
       +------+
```

Description
The **devnm** command reads **path**, identifies the special file associated with the mounted file system where **path** resides, and writes the special file name to standard output. Each **path** must be a full path name and must name a file in a file system mounted on the local cluster site.

The most common use of the **devnm** command is to construct a mount table entry for the root device during system initialization.

Examples

1. To identify the device on which a file resides:

   ```
   devnm /diskette0/bob/textfile
   ```

   This command displays the name of the special device file on which **/diskette0/bob/textfile** resides. If a diskette is mounted as **/diskette0**, the **devnm** command displays:

   ```
   fd0  /diskette0/bob/textfile
   ```

   This means that **/diskette0/bob/textfile** resides on the diskette drive **/dev/fd0.**

2. To identify the device on which a file system resides:

   ```
   devnm  /
   ```

   This command displays the name of the device on which the root file system (/) resides. The following list appears on the screen:

   ```
   hd1  /
   ```

   This means that / resides on **/dev/hd1**.

Files

/dev	Directory containing device special files.
/etc/filesystems	

1.1.121 df

Purpose
Reports the number of available disk blocks.

Syntax

```
      +------+   +------------------+
df ---+- -I -+---¦                  ¦+---¦
      +- -L -¦   +--- filesystem ---+
      +- -M -¦                   ¦
      +- -g -¦     +-------------+
      +- -i -¦
      +- -p -¦
      +- -s -¦
      +- -v -+
```

¦ The default action is to provide information for each file system on each site in the current cluster partition for which the **/etc/filesystems** entry has the attribute free=true.

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **df** command writes to standard output information about total space and available space on the specified file systems. All space values are given in kilobytes. If the **filesystem** argument is used, **df** reports on the specified file system. The **filesystem** argument can be the name of the device on which the file system resides or the directory on which it is mounted.

Normally, the **df** command uses free counts maintained in the superblock. Under certain exceptional circumstances, these counts can be in error. If a file system is being actively modified when **df** is run, the free count can be inaccurate.

The **df** command fails if the file system is unmounted but attached to a site whose CPU type has a different byte ordering than the machine on which the command is run.

Flags

-g Displays the global file system number for each file system.

-i Reports the number of free and used inodes. (If the primary site is down, the **df -i** command reports the number of used inodes is 1 since, without the primary site, this number is unknown.) The **-i** flag is the default for the **df** command.

-I Lists file system, total kilobytes used and free, percentage used, and **Mounted on** columns.

-L If used without a **filesystem** argument, **df** reports on all file systems mounted on the site on which the command is run.

-M Displays **Mounted on** in the second column.

-p Lists the pack number with each file system.

-s Checks for count errors by forcing **df** to fully search the free lists of the local file systems to verify the counts. The **df** command requires considerably more processing time when the **-s** flag is specified. Using the **-s** flag automatically invokes the **-L** flag so that **df** reports on file systems for the site processing the **df** command.

-v Lists all fields (verbose output).

Examples

1. To list information about all default file systems on all sites:

   ```
   df
   ```

   If the cluster contained two sites (**atlas** and **procyon**), each with a mounted **/tmp** directory in addition to the root and local, the output from the **df** command looks similar to this:

   ```
   Filesystem   Total KB   free   %used   iused   %iused   Mounted on
   /dev/hd02     25692     2812     89%   6053      61%    /
   /dev/hd03     11220     1628     85%    669      33%    /atlas
   /dev/hd04     22912     5396     76%   2335      56%    /atlas/tmp
   /dev/hd02     23600      704     97%   6053      61%    /
   /dev/hd03     10092     1192     88%    753      25%    /procyon
   /dev/hd01     11232     8346     25%    562      28%    /procyon/tmp
   ```

2. To list information about the file system on a diskette:

   ```
   df /dev/fd0
   ```

3. To list information about the file system normally mounted as **/diskette0**:

   ```
   df /diskette0
   ```

Files

/etc/filesystems	Lists the known file systems and defines their characteristics.

Related Information

See the following commands: "fsck, dfsck" in topic 1.1.177.

See the **filesystem** file in *AIX Operating System Technical Reference*.

See discussion of the **df** command in *Managing the AIX Operating System*.

1.1.122 dfmt

Purpose
Writes a label, a vtoc and a boot block on a 370 disk.

Syntax

```
        +------ - ------+
dfmt ---¦ +-----------+ +--- special label ---¦
        +-¦ -C -bfile +-+
          ¦ -d -      ¦¦
         ¦¦ -D -R     ¦¦
         ¦¦ -F -vfile ¦¦
         ¦¦ -S        ¦¦
         ¦¦ -l        ¦¦
         ¦+-----------+¦
         +-------------+
```

Description

The **dfmt** command is a 370-specific program used in installation to write a label, vtoc, and a boot block on a 370 disk. By default, **dfmt** writes a label, an empty boot block, and empty vtoc, and a boot record that goes into a wait state.

CKD disks must be formatted with a 4K block size using the CMS format command. The **dfmt** command does not format disks at the hardware level.

Flags

-d	Print progress reports while running.
-D	Only print progress reports, do not write to disk.
-C	Simulate a ckd disk on a normal file.
-F	Simulate a fba disk on a normal file.
-S	Assume default answers on all questions.
-l	Rewrite the label.
-bfile	Install a new boot block from **file**.
-R	Reformat the entire disk.
-0	Reformat only track 0.
-vfile	Write a new vtoc described by **file**.

file may be specified a "-" to standard input. **file** contains lines of the following format:

```
  ID_number startblk endblk
```

Use a negative ID_number to specify a boot partition.

1.1.123 diction, explain

Purpose
Identifies and prints wordy sentences.

Syntax

```
           +--------------+
diction ---¦ +----------+ +--- file ---¦
           +-¦ -ml      +-+
             ¦ -m       ¦¦
            ¦¦ -f pfile ¦¦
            ¦¦ -n       ¦¦
            ¦+----------+¦
            +------------+

explain ---¦
```

Note: This command does not have MBCS support.

Description

The **diction** command finds all sentences in a document that contain phrases from a data base of bad or wordy diction. Each phrase is bracketed with [ ]. Because **diction** runs **deroff** before looking at the text, formatting header files should be included as part of the input.

The **explain** command is an interactive thesaurus for the phrases found by **diction**.

Notes:

1. These commands do not have National Language Support.

2. Use of non-standard formatting macros can cause incorrect sentence breaks. In particular, **diction** does not understand **-me**.

Flags

-ml	Causes **deroff** to skip lists; used if a document contains many lists of non-sentences.
-mm	Overrides the default **-ms** macro package.
-f pfile	Names a user pattern file to be used in addition to the default file.
-n	Used with **-f**, suppresses use of the default file; only the user pattern file is used.

Related Information
The following command: "deroff" in topic 1.1.118.

1.1.124 diff

Purpose
Compares text files.

Syntax

```
        +--------+   +---------------+
diff ---¦        +---¦ +-----------+ +--- file1 --- file2 ---¦
        ¦ +----+ ¦   +-¦ -b        +-+
        +-¦ -e +-+     ¦ -         ¦¦
          ¦ -f ¦      ¦¦ -D string ¦¦
          ¦ -h ¦      ¦¦ -l        ¦¦
          ¦ -n ¦      ¦¦ -r        ¦¦
          +----+      ¦¦ -s        ¦¦
                      ¦¦ -S name   ¦¦
                      ¦¦ -i        ¦¦
                      ¦¦ -t        ¦¦
                      ¦¦ -w        ¦¦
                      ¦+-----------+¦
                      +-------------+
```

Description
The **diff** command compares **file1** and **file2** and writes to standard output information about what changes must be made to make them identical. If you specify a **-** (minus) for **file1** or **file2**, the **diff** command reads standard input. If **file1** is a directory, the **diff** command uses a file in that directory with the name **file2**. If **file2** is a directory, the **diff** command uses a file in that directory with the name **file1**. If both **file1** and **file2** are directories, the contents of the directories are sorted and the **diff** command processes all text files that differ in the directories. Binary files that differ, common subdirectories, and files that appear in only one directory are listed.

The typical output contains lines in these forms:

Lines Affected in *file1*	**Action**	**Lines Affected in** *file2*
num1	**a**	**num2**[,**num3**]
num1[,**num2**]	**d**	**num3**
num1[,**num2**]	**c**	**num3**[,**num4**]

These lines resemble **ed** subcommands to convert **file1** into **file2**. The numbers before the action letters pertain to **file1**; those after pertain to **file2**. Thus, by exchanging **a** for **d** and reading backward, you can also tell how to convert **file2** into **file1**. As in **ed**, identical pairs (where **num1** = **num2**) are abbreviated as a single number.

Following each of these lines, the **diff** command displays all lines affected in the first file preceded by a <, then all lines affected in the second file preceded by a >.

Except in rare circumstances, the **diff** command finds a smallest sufficient set of file differences. An exit value of 0 indicates no differences, 1 indicates differences found, and 2 indicates an error.

Note: Editing scripts produced by the **-e** or **-f** flags cannot create lines consisting of a single period (.).

Flags

-b Ignores trailing spaces and tab characters and considers other strings of blanks to compare as equal.

-c Produces a diff with lines of context. The default is to present three lines of context. You can change the default by specifying a number after the **c**. With this option, the output format is modified slightly. The output begins with identification of the files involved and their creation dates, and then each change is separated by a line with 12 asterisks. The lines removed from **file1** are marked with "-", and those added to **file2** are marked with "+". Lines that are changed from one file to the other are marked in both files with "!".

-D string
Causes the **diff** command to create a merged version of **file1** and **file2** on the standard output, with C preprocessor controls included so that a compilation of the result without defining string is equivalent to compiling **file1**, while defining string will yield **file2**.

-e Produces output in a form suitable for use with the **ed** command to convert **file1** to **file2**.

-f Produces output in a form not suitable for use with **ed**, showing the modifications necessary to convert **file1** to **file2** in the reverse order of that produced under the **-e** flag.

-h Performs a faster comparison. This flag only works when the changed sections are short and well separated, but it does work on files of any length. The **-c**, **-Dstring**, **-e**, and **-f** flags are not available when you use the **-h** flag.

-i Ignore case.

-l If both files specified are directories, this flag produces a long output format. Each text file the **diff** command prints is piped through **pr(1)** to paginate it, and other differences are summarized after all text file differences are reported.

-n Produces output in a form not suitable for use with the **ed** command, showing the modifications necessary to convert **file1** to **file2**.

-r If both files specified are directories, this option causes the **diff** command to be applied recursively through all common subdirectories involved.

-s If both files specified are directories, this option causes the **diff** command to report files which are the same. Normally, these files are not mentioned.

-S name
If both files specified are directories, this option starts the directory comparison after encountering the file name.

-t Expands tabs on output.

-w Ignores blanks.

Examples

1. To compare two files:

   ```
   diff chap1.bak chap1
   ```

   This command displays the differences between the files **chap1.bak** and **chap1**.

2. To compare two files, ignoring differences in the number of blanks:

   ```
   diff  -b  prog.c.bak  prog.c
   ```

   If two lines differ only in the number of blanks and tabs between words, the **diff** command considers them to be the same.

3. To create a file containing commands that the **ed** command can use to reconstruct one file from another:

   ```
   diff  -e  chap2  chap2.old  >new.to.old.ed
   ```

   This command creates a file named **new.to.old.ed** that contains the **ed** commands used to change **chap2** back into the version of the text found in **chap2.old**. In most cases, the file **new.to.old.ed** is a much smaller than the file **chap2.old**. You can save disk space by deleting the file **chap2.old**, but you can reconstruct the file at any time by entering:

   ```
   (cat new.to.old.ed ; echo  '1,$p') | ed  -  chap2  >chap2.old
   ```

   The commands in parentheses add **1,$p** to the end of the list of editing commands sent to the **ed** command. The **1,$p** causes the **ed** command to write the file to standard output after editing it. This modified command sequence is then piped to the **ed** command (**| ed**), and the editor reads it as standard input. The **-** flag causes the **ed** command not to display the file size and other extra information since these would be mixed with the text of the file **chap2.old**. (See page 1.1.420.22 for details about grouping commands with parentheses.)

Files

/tmp/d?????	Temporary files.
/usr/lib/diffh	For the **-h** flag.

Related Information

See the following commands: "bdiff" in topic 1.1.37 "cmp" in topic 1.1.76, "comm" in topic 1.1.83, "ed, red" in topic 1.1.147, and "sdiff" in topic 1.1.413.

1.1.125 diffmk

Purpose
Identifies differences between two versions of a file.

Syntax

```
          +---------------------------+                       +---------+
diffmk ---¦ +-----------------------+ +--- file1 --- file2 ---¦         +---
          +-¦ -b                    +-+                       +- file3 -+
            ¦ -abstringX  -aestringX ¦¦
           ¦¦ -cbstringX  -cestringX ¦¦
           ¦¦ -dbstringX  -destringX ¦¦
           ¦+-----------------------+¦
           +-------------------------+
```

Note: This command does not have MBCS support.

Description
The **diffmk** command compares **file1** and **file2** and creates a third file that includes **change mark commands** for the **nroff** and **troff** commands. The old and new versions of the file are **file1** and **file2**, respectively. The **diffmk** command writes the newly created file to **file3**, if specified, or to standard output. This file contains the lines of **file2** with formatter change mark (**.mc**) requests inserted as appropriate. When **file3** is formatted, the changed or inserted text is marked with a | (vertical bar) in the right margin of each line. An ***** (asterisk) in the margin indicates that a line was deleted.

If the environment parameter DIFFMARK is defined, it names a command string that the **diffmk** command uses to compare the files. (Normally, the **diffmk** command uses the **diff** command.) For example, you might set DIFFMARK to **diff -h** in order to better handle extremely large files.

Flags

-b	Ignores differences that are changes to tabs or spaces on a line.
-abstringX -aestringX	Uses **X** (a string) to mark where added lines begin and end.
-cbstringX -cestringX	Uses **X** (a string) to mark the beginning and end of a changed line.
-dbstringX -destringX	Uses **X** (a string) to mark where deleted lines begin and end.

Examples

1. To identify differences between two versions of a text file:

   ```
   diffmk  chap1.old  chap1  > chap1.nroff
   ```

   This command produces a copy of the file **chap1**, which contains **nroff/troff** change mark commands that identify text that has been added to, changed in, or deleted from the file **chap1.old**. This copy is saved in the file **chap1.nroff**.

2. To identify differences between files without issuing **nroff/troff** messages:

   ```
   diffmk -ab'>>New:' -ae'<<End New' chap1.old  chap1  >chap1.nroff
   ```

   This command causes the **diffmk** command to write the message **>>New:** on the line preceding a section of new lines that have been added to the file **chap1** and to write the message **<<End New** on the line following the added lines. Changes and deletions still generate **nroff/troff** commands to put a | or * in the margin.

3. To use different **nroff/troff** marking commands and ignore changes in white space:

   ```
   diffmk  -b  -cb'.mc  %'  -ce' ' chap1.old  chap1  > chap1.nroff
   ```

   This command imbeds commands that mark changes to lines with **%**, additions with |, and deletions with *****. It does not mark changes that only involve a different number of spaces or tabs between words (**-b**).

Related Information

See the following commands: "diff" in topic 1.1.124 and "nroff, troff" in topic 1.1.301.

1.1.126 diff3

Purpose
Compares three files.

Syntax

```
         +--------+
diff3 ---¦ one of +-- file1 -- file2 -- file3 --¦
         ¦ +----+ ¦
         +-¦ -e +-+
           ¦ -x ¦
           ¦ -3 ¦
           +----+
```

Description
The **diff3** command reads three versions of a file and writes to standard output the ranges of text that differ, flagged with the following codes:

====	All three files differ.
====1	**file1** differs.
====2	**file2** differs.
====3	**file3** differs.

The type of change needed to convert a given range of a given file to match another file is indicated in one of these two ways in the output:

file : n1 a Where **file** is **1**, **2**, or **3**, text is to be added after line **n1**.

file : n1[,n2] c Where **file** is **file 1**, **2**, or **3**, text between line **n1** and line **n2** is to be changed. If **n1** = **n2**, the range may be abbreviated to **n1**.

The original contents of the range follows immediately after a **c** indication. When the contents of two files are identical, the **diff3** command does not show the contents of the lower-numbered file, although it shows the location of the identical lines for each.

Notes:

1. Editing scripts produced by the **-e** flag cannot create lines consisting only of a single period (**.**).

2. The **diff3** command does not work on files longer than 64K bytes.

Flags

-e Creates an edit script for use with the **ed** command to incorporate into **file1** all changes between **file2** and **file3** (that is, the changes that normally would be flagged ==== and ====3).

-x Produces an edit script to incorporate only changes flagged ====.

-3 Produces an edit script to incorporate only changes flagged ====3.

Example

The following table illustrates the contents of three files **fruit.a, fruit.b**, and **fruit.c**:

fruit.a	**fruit.b**	**fruit.c**
banana	apple	grape
grape	banana	grapefruit
kiwi	grapefruit	kiwi
lemon	kiwi	lemon
mango	orange	mango
orange	peach	orange
peach	pear	peach
pare		pear

To list the differences among the three files:

```
  diff3  fruit.a  fruit.b  fruit.c
```

The output from the **diff3** command shows the differences as follows. (Comments to the right do not appear in the output.)

```
====     All three files are different.
1:1,2c    - Lines 1 and 2 of the first file, fruit.a
  banana
  grape
2:1,3c    - Lines 1 through 3 of the second file, fruit.b
  apple
  banana
  grapefruit
3:1,2c    - Lines 1 and 2 of the third file, fruit.c
  grape
  grapefruit
====2    The second file, fruit.b, is different.
1:4,5c    - Lines 4 and 5 are the same in fruit.a and fruit.c.
2:4a      - To make fruit.b the same as fruit.a  and fruit.c, add text after
3:4,5c
  lemon
  mango
====1    The first file, fruit.a, is different from files fruit.b and file.c
1:8c
  pare
2:7c      - Line 7 of fruit.b and line 8 of fruit.c are the same
3:8c
  pear
```

Files

/tmp/d3*	Temporary files for **diff3**.
/usr/lib/diff3prog	**diff3** executable program.

Related Information

See the following command: "diff" in topic 1.1.124.

1.1.127 dircmp

Purpose
Compares two directories and the contents of their common files.

Syntax

```
          +--------+
dircmp ---¦ +----+ +-- directory1 -- directory2 --¦
          +-¦ -d +-+
            ¦ -  ¦¦
           ¦+----+¦
           +------+
```

Description
The **dircmp** command reads **directory1** and **directory2** and writes information about their contents to standard output. First, the **dircmp** command compares the file names in each directory. When the same file names appear in both, the command compares the contents of those files. The **dircmp** command lists the files unique to each directory in the output. It then lists the files with identical names in both directories, but with different contents. If entered with no flag, it also lists files that have identical contents as well as identical names in both directories.

Flags

-d Displays for each common file name both versions of the differing file lines. The display format is the same as that of the "diff" in topic 1.1.124 command.

-s Does not list the names of identical files.

Examples

1. To summarize the differences between the files in two directories:

   ```
   dircmp  proj.ver1  proj.ver2
   ```

   This command displays a summary of the differences between the directories **proj.ver1** and **proj.ver2**. The summary lists separately the files found only in one directory or the other, and those found in both. If a file is found in both directories, the **dircmp** commands notes whether the two files are identical.

2. To show the details of the differences between files:

   ```
   dircmp  -d  -s  proj.ver1  proj.ver2
   ```

   The **-d** flag displays a **diff** listing for each of the differing files found in both directories.

   The **-s** flag suppresses information about identical files.

Related Information

See the following commands: "cmp" in topic 1.1.76 and "diff" in topic 1.1.124.

1.1.128 dis

Purpose
Produces PS/2 Assembler language listing from compiled programs.

Syntax

```
       +-------------------------------------------------------------
dis ---¦ +-----------------+   +------------------------+   +- object file --
       +-¦ +-------------+ +---¦         one of         +---¦                +
         +-¦ -o filename +-+   ¦ +---------------------+ ¦   +- executable ---
           ¦ -           ¦¦    +-¦ -e entry-point name +-+
          ¦+-------------+¦      ¦ -r                  ¦
          +---------------+      +---------------------+
```

Note: This command is for the PS/2 only.

Description

The **dis** command provides a disassembler to be used with AIX C, VS Pascal and VS FORTRAN programs on the PS/2. The disassembler can be executed from the command line or through the menu system. It can operate on either the object file **(file.o)** or the executable file **(file)**. Use the **d+** flag when you compile your program to produce the symbolic information the disassembler needs. Although the symbolic information is not necessary, the function of the disassembler becomes limited without it. It uses this information to generate user type declarations, user variable location information, user statement labels, and unique symbol labels.

When disassembling executable files made from separately compiled modules, full symbolic information exists only for the entry points contained in the modules compiled with the **d+** flag.

Note: **dis** works on C files compiled with both Extended C and C compilers.

Flags

-e entry-point	Specifies the entry point to be disassembled. An entry point is the name of a function, procedure, subroutine, program, or FORTRAN entry statement in the user's program. The entry point name is case sensitive. All Pascal and FORTRAN entry point names must be specified in lowercase. This option cannot be used with the **-r** option.
-o file name	Specifies the output file for disassembled code.
-r	Specifies that the output of the disassembled program can be reassembled using the assembler (see the **as** command). This option cannot be used with executable files; it is used with object files only. The absence of the symbolic information prohibits the use of this option from the command line. This option cannot be used with the **-e** option.
-w	Suppresses the display of warning messages.

Files

/usr/lib/msg/vsdismsg.inc Message file, which **must** be present.

Related Information

See the following commands: "as" in topic 1.1.25 and "cc" in topic 1.1.52.

See the discussion of the disassembler in *AIX Operating System Programming Tools and Interfaces*.

1.1.129 diskusg

Purpose
Generates disk accounting data by userid.

Syntax

```
                              +---------------+   +- -p/etc/passwd -+   +----------
/usr/lib/acct/diskusg ---¦ +-----------+ +---¦                  +---¦   one of
                              +-¦ -s        +-+   +---- -pfile -----+   ¦ +--------
                                ¦ -ifnmlist ¦¦                          +-¦ -u file
                               ¦+-----------+¦                            ¦ -v
                               +-------------+                            +--------
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **diskusg** command generates intermediate disk accounting information from data in **file**s or from standard input, if you do not specify any files. The **diskusg** command writes lines to standard output, one per user, in the following format:

uid login #blocks

where:

uid	is the numerical user ID of the user
login	is the login name of the user
#blocks	is the total number of disk blocks allocated to the user.

The **diskusg** command normally reads only the inodes of file systems for disk accounting. In this case, **file**s are the special file names of these devices.

AIX uses a disk block allocation strategy such that no blocks are allocated for files of 384 or fewer bytes. The data for such files is held in the inode. **diskusg** reports these files as using 0 disk blocks.

Note: This command is for local devices only.

Flags

-ifnmlist	Ignores the data on those file systems with a file system name in **fnmlist**. The **fnmlist** is a list of file system names. The **diskusg** compares each name in this list with the file system name stored in the volume ID.
-pfile	Uses **file** as the name of the password file to generate login names. The file **/etc/passwd** is used by default.
-s	Combines all lines for a single user into a single line. (The input date is already in **diskusg** command output format.)
-u file	Writes records to **file** of files that are charged to no one. Records consist of the special file name,

	the inode number, and the user ID.
-v	Writes a list to standard error of all files that are charged to no one.

The output of the **diskusg** command is normally the input to the **acctdisk** command, which generates total accounting records that can be merged with other accounting records. The **diskusg** command is normally run in **dodisk** (see "acct/*" in topic 1.1.6).

Examples

The following sequence of commands generates daily disk accounting information:

```
  for  i  in  /dev/hd0  /dev/hd1  /dev/hd2  /dev/hd3
  do
        diskusg  $i  > dtmp.'basename $i'  &
  done
  wait
  diskusg  -s  dtmp.*  | sort  +On  +1  |  acctdisk  > dacct
```

Files

/etc/passwd	Used for user-ID-to-login-name conversions.

Related Information
See the following commands: "acct/*" in topic 1.1.6, "acctcms" in topic 1.1.7, "acctcom" in topic 1.1.8, "acctcon1, acctcon2" in topic 1.1.9, "acctmerg" in topic 1.1.11, and "runacct" in topic 1.1.398.

See the **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

See the discussion of accounting in *Managing the AIX Operating System*.

1.1.130 display

Purpose
Sets colors and fonts on the current virtual terminal.

Syntax

```
             +---------------+
display ---¦                 +---
             +- displayname -+

    +----------------------+   +----------------------+   +-------------------
 ---¦ +----+   +---------+ +---¦      +-------------+ +---¦      +------------
    +-¦ -b +---¦         +-+   +- -p -¦             +-+   +- -t -¦¦
  ¦   ¦ -f ¦   +- color -+            +- colorfile -+            +--- font ---
  ¦   +----+                                                                 ¦
  ¦                                                               +--------+
  +---------------------------------------------------------------------------
```

¦ Do not put a blank between these items.

Note: The **display** command is for the PS/2 only.

Flags

displayname Can be one of the following:

1 First display
2 Second display
3 Third display
4 Fourth display

If no **displayname** is specified, the user is presented with a list of attached displays and asked which one to change to -. If an invalid display number is entered, an error message is displayed. These flags change the physical display used by the standard out terminal and/or changes the default display for subsequent opens is changed to the specified display.

-b Specifies the current background color.

-f Specifies the current foreground color.

color Specifies the active color number (1 - 16). If no **color** is specified with either the **-f** or **-b** options, the user is presented with a display of the first 16 active colors to choose from.

-p Clears the screen.

colorfile Specifies the path name of a file that contains the ASCII values of the colors to be put into the active color pallet. The format of the file is:

```
colorva11
colorva12
.
```

```
  .
  .
  colorval**n**
```

If no **colorfile** is specified, the default **colorfile** for the current display is used. The default path name is: **/etc/vtm/palxxxx** where **xxxx** is the display's physical device ID.

-t Specifies the active current and/or alternate fonts.

font Lists the font IDs to be changed. The first one is the primary (current) font; the second, the first alternate font; the third, the second alternate font, and so on. If no font IDs are specified, the user is presented with a list of all the fonts available for the current display and is prompted to choose the current and alternate fonts.

1.1.131 dist

Purpose
Redistributes a message to additional addresses.

Syntax

```
       +-----------+   +------- cur -------+   +-- -noannotate --+
dist----¦           +---¦                   +---¦                 +---
       +- +folder -+   ¦       one of       ¦   ¦     one of      ¦
                       ¦ +---------------+  ¦   ¦ +-------------+ ¦
                       +-¦ num      cur  +--+   +-¦ -annotate   +-+
                         ¦ sequence .    ¦        ¦ -noannotate ¦
                         ¦ first    next ¦        +-------------+
                         ¦ prev     last ¦
                         +---------------+

   +------------------------------------------------------------------+
---¦ +--- -nodraftfolder -------------------------------------------+ +---
   ¦ ¦                                                              ¦ ¦
   +-¦                one of                  +------- new -------+ +-+
     ¦ +------------------------------------+ ¦                   ¦ ¦
     ¦ ¦ -draftfolder +folder -draftmessage ¦ ¦      one of       ¦ ¦
     +-¦ -draftfolder +folder               +---¦ +---------------+ +-+
       ¦ -draftmessage                      ¦ +-¦ num        .    +-+
       +------------------------------------+   ¦ sequence next ¦
                                                ¦ first    last ¦
                                                ¦ prev     new  ¦
                                                ¦ cur           ¦
                                                +---------------+

   +--------------+   +-----------------+   +-- -noinplace --+
---¦               +---¦     one of      +---¦                +---
   +- -form file -+   ¦ +-------------+ ¦   ¦     one of      ¦
                      +-¦ -editor cmd +-+   ¦ +------------+  ¦
                        ¦ -noedit     ¦     +-¦ -inplace   +--+
                        +-------------+       ¦ -noinplace ¦
                                              +------------+

   +----------------------------+
---¦           one of            +---¦
   ¦ +------------------------+ ¦
   +-¦ -whatnowproc cmdstring +-+
     ¦ -nowhatnowproc         ¦
     +------------------------+

dist --- -help ---¦
```

Description

The **dist** command is used to redistribute messages to a new list of addresses. This command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

By default, the **dist** command copies a message form to a new draft message and invokes an editor. You then can fill in the message header fields **Resent-To:** and **Subject:** and fill in or delete the other header fields (such as **Resent-cc:** and **Resent-Bcc:**). Since the body of the message is the message you are redistributing, do not fill in the body. The **dist** command does not automatically display the body of the message. When you exit the editor, the **dist** command invokes the MH command **whatnow**. You can press **Enter** to see a list of the available **whatnow** subcommands. These subcommands enable you to continue editing the message header, list the message header, direct the disposition of the message, or end the processing of the **dist** command. See "whatnow" in topic 1.1.533 for a description of the subcommands.

When you send the draft message, the recipients are sent the headers and body of the original message, which are appended to the new message. The **dist** command does not automatically store a copy of the original message with the new draft message. The draft message you create using the **dist** command consists of header fields only.

You can specify the message that you want to distribute by using the **+folder** *msg* flag. If you do not specify a message, the **dist** command redistributes the current message.

You can specify the format of the message header by using the **-form** flag. If you do not specify this flag, **dist** uses your default message format located in the file **user_mh_directory/distcomps**. If this file does not exist, **dist** uses the system default message format located in **/usr/lib/mh/distcomps**. **dist** assigns the form to the message being redistributed.

Note: The line of dashes or a blank line must be left between the header and the body of the message for the message to be identified when it is sent.

Flags

-annotate
Annotates the message being redistributed with the lines:

```
Resent: date
Resent: addrs
```

The annotation appears in the original draft message so that you can maintain a complete list of recipients with the original message. If you do not actually redistribute the message using the immediate **dist** command, the **-annotate** flag may fail to provide annotation. The **-inplace** flag forces annotation to be done in place.

-draftfolder +*folder*
Places the draft message in the specified folder. If you do not specify this flag, **dist** selects a default draft folder according to the information supplied in the MH profiles. You can define a default draft folder in **$HOME/.mh_profile**. If **-draftfolder +***folder* is followed by *msg*, *msg* represents the **-draftmessage** attribute.

-draftmessage *msg*
Specifies the draft message. You can specify one of the following message references as *msg*:

num	*sequence*	**f**
prev	**cur**	**.**
next	**last**	**n**

The default draft message is **new**. If you specify a draft message, that message becomes the current message.

-editor *cmd*
Specifies that *cmd* is the initial editor for preparing the message for distribution. If you do not specify this flag, **dist** selects a default editor or suppresses the initial edit, according to the information supplied in the MH profiles. You can define a default initial editor in **$HOME/.mh_profile**.

+*folder msg*
Redistributes the specified message in the specified folder. You can specify one of the following message references for *msg*:

num	*sequence*	**f**
prev	**cur**	**.**
next	**last**	

The default message is the current message in the current folder. If you specify a folder, that folder becomes the current folder.

-form *file*
Assigns the form contained in the specified file to the message being resent. The **dist** command treats each line in *file* as a format string.

-help
Displays help information for the command.

-inplace
Forces annotation to be done in place to preserve links to the annotated message.

-noannotate
Does not annotate the message. This flag is the default.

-nodraftfolder
Places the draft in the file **user_mh_directory/draft**.

-noedit
Suppresses the initial edit.

-noinplace
Does not perform annotation in place. This flag is the default.

-nowhatnowproc
Does not invoke a program that guides you through the distribution tasks. The **-nowhatnowproc** flag also prevents any edit from occurring.

-whatnowproc *cmdstring*
Invokes *cmdstring* as the program to guide you through the distribution tasks. See "whatnow" in topic 1.1.533 for information about the default **whatnow** program and its subcommands.

Note: If you specify **whatnow** for *cmdstring*, the **dist** command invokes an internal **whatnow** procedure rather than a program with the file name **whatnow**.

Profile Entries

Current-Folder:	Sets your default current folder.
Draft-Folder:	Sets your default folder for drafts.
Editor:	Sets your default initial editor.
fileproc:	Specifies the program used to refile messages.
Path:	Specifies your **user_mh_directory**.
whatnowproc:	Specifies the program used to prompt "What now?" questions.

Files

/usr/lib/mh/distcomps	System default message skeleton.
user_mh_directory/distcomps	User's default message skeleton. If it exists, it overrides the system default message skeleton.
$HOME/.mh_profile	MH user profile.
user_mh_directory/draft	Draft file.

Related Information

See other MH commands: "ali" in topic 1.1.17, "anno" in topic 1.1.19, "comp" in topic 1.1.85, "forw" in topic 1.1.174, "prompter" in topic 1.1.334, "refile" in topic 1.1.366, "repl" in topic 1.1.369, "send" in topic 1.1.416, and "whatnow" in topic 1.1.533.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

"See Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.132 dmesg

Purpose
Collects system diagnostic messages to form an error log.

Syntax

```
        +-----+
dmesg ---¦     +---¦
        +- - -+
```

Description

The **dmesg** command looks in a system buffer for recently printed diagnostic messages and prints them on the standard output. The messages are those printed or logged by the system when errors occur. If the **-** flag is given, then the **dmesg** command computes (incrementally) the new messages since the last time it was run and places these on the standard output.

Files

/usr/adm/msgbuf	Scratch file for memory of - option.
/dev/osm	Used to access the kernel's diagnostic message buffer.

Related Information

See the following command: "syslogd" in topic 1.1.457.

1.1.133 domainname

Purpose
Sets or displays the name of the current network information service (NIS) domain.

Syntax

```
                +--------+
domainname ---¦        +---¦
                +- name -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **domainname** command displays the name of the current NIS domain. If you have superuser authority, you can also use this command to set the name of the domain.

A **domain** is a group of host machines in the network. The name of the domain typically is set in the startup script for the **/etc/rc.nfs** file.

Files
/etc/rc.nfs

1.1.134 dosdel

Purpose
Deletes DOS files.

Syntax

```
         +-------------------+
dosdel --¦   +-----------+   +--- file --¦
         ¦   ¦ -v        ¦   ¦
         +---¦ -D device +---+
             +-----------+ ¦
           +---------------+
```

Note: This command does not have MBCS support.

Description
The **dosdel** command deletes the DOS file specified by **file**. Use the **-v** flag to obtain format information about the disk.

File-naming conventions are the same as DOS, with one exception: the **doswrite** command replaces the \ (backslash) character used to separate components of a DOS path name with the / (slash) character. The **dosdel** command converts lowercase characters in the **file** parameter to uppercase before it checks the disk. Because the file name is assumed to be the full (not relative) path name, you need not add the initial / (slash).

Flags

-D device	Specifies a device or file system to use as the DOS disk. If you do not specify this flag, the default device is **/dev/fd0**.
-v	Writes format information about the disk. Use primarily to verify the identify of a disk or file system as a DOS disk.

Related Information

See the following commands: "dosdir" in topic 1.1.135, "dosread" in topic 1.1.136, and "doswrite" in topic 1.1.137.

1.1.135 dosdir

Purpose
Lists the directory for DOS files.

Syntax

```
          +---------------+   +------------+   +- -D/dev/fd0 -+
dosdir ---¦       +------+ +---¦   +----+   +---¦              +---
          +- -l -¦      +-+   +---¦ -a +---+   +- -D device --+
                 +- -e -+         ¦ -t ¦
                                ¦ ¦ -d ¦ ¦
                                ¦ ¦ -v ¦ ¦
                                ¦ +----+ ¦
                                +--------+

   +------------------+
 --¦   +----------+   +--¦
   +---¦ file     +---+
       ¦ directory¦
     ¦ +----------+ ¦
     +--------------+
```

Note: This command does not have MBCS support.

Description
The **dosdir** command displays information about the specified DOS file or directory (the current directory by default). If you specify a directory without also specifying the **-d** flag, the **dosdir** command displays information about the files in that directory.

File-naming conventions are the same as DOS, with one exception: the **dosdir** command replaces the \ (backslash) character used to separate components of a DOS path name with a / (slash) character because the backslash can have special meaning to the AIX Operating System. The **dosdir** command converts lowercase characters in the file or directory name to uppercase before it checks the disk. Because the file name is assumed to be the full (not relative) path name, you need not add the initial / (slash).

Flags

-a	Writes information about all files. This includes hidden and system files as well as the . (dot) and .. (dot dot) files.
-d	Treats **file** as a file, even if it is a directory. If a directory is specified, information about the directory is listed rather than information about the files it contains.
-D [device]	Specifies a device or file system to use as the DOS disk. If you do not specify this flag, the default device is **/dev/fd0**.
-e	Uses the **-l** flag to write the list of clusters allocated to the file.
-l	Produces a long list that includes the creation date, size in bytes, and attributes. The size of a subdirectory is specified as 0 bytes. The attributes have the following meanings:

A	Archive: the file has not been backed up since it was last modified.
D	Directory: the file is a subdirectory, and is not included in the normal DOS directory search.
H	Hidden: the file is not included in the normal DOS directory search.
R	Read-only: the file cannot be modified.
S	System: the file is a system file, and is not included in the normal DOS directory search.

-t Lists the entire directory tree starting at the named directory.

-v Writes information about the format of the disk.

Related Information
See the following commands: "dosread" in topic 1.1.136 and "doswrite" in topic 1.1.137.

1.1.136 dosread

Purpose
Copies a DOS file.

Syntax

```
           +--------+   +- -D/dev/fdO -+          +---------+
dosread ---¦ +----+ +---¦              +-- file1 --¦         +---¦
           +-¦ -a +-+   +- -D device --+          +- file2 -+
             ¦ -v ¦¦
            ¦+----+¦
            +------+
```

Note: This command does not have MBCS support.

Description
The **dosread** command copies the specified DOS **file1** to standard output or to the specified AIX **file2** (by default the root directory). Unless otherwise specified, the **dosread** command copies as many bytes as are specified in the directory entry for **file1**. This means, in particular, that copying directories does not work, since directories by convention have a record size of 0.

File-naming conventions are the same as DOS, with one exception: the **dosread** command replaces the \ (backslash) character used to separate components of a DOS path name with a / (slash) character because the backslash can have special meaning to the AIX Operating System. The **dosread** command converts lowercase characters in the **file1** name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

Notes:

1. Pattern-matching characters (***** and **?**) are not treated in a special way by this command (although they are by the shell). If, for example, you do not specify a file-name extension, the file name is matched as if you specified a blank extension.

2. This command must be named **dosread**.

Flags

-a	Replaces the sequence CRLF (carriage return-line feed) with NL (new-line character) and interprets a **Ctrl-Z** (ASCII SUB) as the end-of-file character.
-D device	Specifies the name of the DOS device or file system. The default *device* is **/dev/fd0**. This device must have the DOS-disk format.
-v	Writes information to the standard output about the format of the disk. Use this flag to verify that a device or file system is a DOS disk.

Examples

1. To copy a text file from a DOS diskette to the AIX file system:

```
  dosread -a chap1.doc chap1
```

This command copies the DOS text file **\CHAP1.DOC** on **/dev/fd0** to the AIX file **chap1** in the current directory.

2. To copy a binary file from a fixed-disk DOS file system to the AIX file system:

   ```
     dosread -D/dev/hd1 /survey/test.dta /u/fran/testdata
   ```

   This command copies the DOS data file **\SURVEY\TEST.DTA** on **/dev/hd1** to the AIX file **/u/fran/testdata**.

Files

/dev/fd0	Device name for diskette drive.

Related Information
See the following commands: "dosdel" in topic 1.1.134, "dosdir" in topic 1.1.135, and "doswrite" in topic 1.1.137.

1.1.137 doswrite

Purpose
Copies AIX files to DOS files.

Syntax

```
            +--------+   +- -D/dev/fdO -+
doswrite ---¦ +----+ +---¦              +-- file1 -- file2 --¦
            +-¦ -a +-+   +- -D device --+
              ¦ -  ¦¦
             ¦+----+¦
             +------+
```

Note: This command does not have MBCS support.

Description
The **doswrite** command copies the specified AIX **file1** to the specified DOS **file2**. If **file2** is a multi-component name (that is, if it contains /), each intervening component must exist as a directory, and the last component (the named file) must not exist.

File-naming conventions are the same as DOS, with one exception: the **doswrite** command replaces the \ (backslash) character used to separate components of a DOS path name with the / (slash) character because the backslash can have special meaning to AIX. The **doswrite** command converts lowercase characters in the **file1** name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

Notes:

1. Pattern-matching characters (***** and **?**) are not treated in a special way by this command (although they are by the shell). If, for example, you do not specify a file-name extension, the file name is matched as if you specified a blank extension.

2. This command must be named **doswrite**.

Flags

-a	Replaces NL (new-line character) characters with the sequence CRLF (carriage return-linefeed). A **Ctrl-Z** is added to the output at end of file.
-D device	Specifies the name of the DOS device or file system. The default *device* is **/dev/fd0**. To use a low density diskette, specify **/dev/fd01**. This device must have the DOS-disk format.
-v	Writes information to the standard output about the format of the disk. Use this flag to verify that a device or file system is a DOS disk.

Examples

1. To copy a text file from the AIX file system to a DOS diskette:

```
doswrite  -a  chap1  chap1.doc
```

This command copies the AIX file **chap1** in the current directory to the DOS text file **\CHAP1.DOC** on **/dev/fd0**.

2. To copy a binary file from the AIX file system to a fixed-disk DOS file system:

```
doswrite  -D/dev/hd1  /u/fran/testdata  /survey/test.dta
```

   This copies the AIX data file **/u/fran/testdata** to the DOS file **\SURVEY\TEST.DTA** on **/dev/hd1**.

Files

/dev/fd0	Device name for the diskette drive.

Related Information
See the following commands: "dosdir" in topic 1.1.135, "dosread" in topic 1.1.136, and "dosdel" in topic 1.1.134.

1.1.138 dp

Purpose
Parses and reformats dates.

Syntax

```
                    +--------------------+   +--------------+
/usr/lib/mh/dp ---¦       one of         +---¦              +--- date ---¦
                  ¦ +----------------+ ¦   +- -width num -+          ¦
                  +-¦ -form file      +-+                    +--------+
                    ¦ -format string ¦
                    +----------------+

/usr/lib/mh/dp --- -help ---¦
```

Note: This command does not have MBCS support.

Description

The **dp** command is used to parse and reformat dates. The **dp** command is not designed to be run directly by the user; it is designed to be called by other programs. The **dp** command is typically called by its full path name. The **dp** command is part of the Message Handling (MH) package.

The **dp** command parses each string specified as a date and attempts to reformat the string. The default output format for the command is the ARPA RFC822 standard. For each string it is unable to parse, the **dp** command displays an error message.

Flags

-form *file*	Reformats the given dates into the alternate format described in *file*.
-format *string*	Reformats the given dates into the alternate format specified by *string*. The default format string is: %<(nodate{**date**})error:%{**date**}%|%(putstr(pretty{**date**}))%> Prints "error" if the date is not a parsable date, or prints a friendly rendering of the date.
-help	Displays help information for the command.
-width *num*	Sets the maximum number of columns that the **dp** command uses to display dates and error messages. The default is the width of the display.

Files

$HOME/.mh_profile	The MH user profile.

Related Information
See the following command: "ap" in topic 1.1.20.

See the **mh-format** and **mh-profile** files in *AIX Operating System Technical Reference*.

"See Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.139 dspcat

Purpose
Displays messages from a catalog.

Syntax

```
         +-------------- -g ----- catfile ------------------+
dspcat---¦              +----------------+  +-------------+  +---¦
         +-- catfile --¦                +--¦             +--+
                        +-- set-number --+  +-- msg-num --+
```

Description
The **dspcat** command displays part or all of the contents of a message catalog. Use it from the command line to visually verify the contents of a message catalog. A separate command, **dspmsg** (page 1.1.140), provides the means to display one message at a time, and is usually run from within a shell script program.

The **dspcat** command first searches the directory **/usr/lib/mbcs/msg/language.codeset/** for **catfile**. The **language.codeset** depends on the current locale. (The locale may be changed using the **LANG** environment variable.) For example, if the current locale is English, the default search directory would be **/usr/lib/mbcs/msg/En.pc850/**. If **catfile** is not found in the default search directory, **dspcat** searches the current directory for **catfile**.

The **catfile** parameter refers to the ***message catalog*** being accessed. If you specify a **set-number**, all messages within a set in the catalog are displayed. The combination of set number and message number displays only the message that is uniquely identified by the message number residing in the specified set. The format of the output is as follows:

```
  set-number:msg-number:text
```

If both message and set numbers are specified, only the text is displayed as output.

Note: Message catalog files are binary files which must be created by the **gencat** command.

Flags

-g Outputs a message text source file using the messages in the specified catalog. If you redirect the output into a file, you can edit the messages, and later process them back into a message catalog using the **gencat** command.

Example
To display the entire contents of the **mymessages.cat** message file, type:

```
  dspcat mymessages.cat
```

The contents of **mymessages.cat** will be displayed on standard output.

Related Information

In this book, see the following commands: "gencat" in topic 1.1.184 and "dspmsg" in topic 1.1.140.

See the **catopen** subroutine in the *AIX Technical Reference*.

See the "Message Catalog Generation" section of the "International Character Support" chapter of the *AIX Operating System Programming Tools and Interfaces*.

See the "Messages" chapter in the *AIX MBCS Guide*.

1.1.140 dspmsg

Purpose

Displays a message from a message catalog or a default message if the specified message cannot be accessed.

Syntax

```
          +-----------------+
dspmsg ---¦                 +--- catfile --- msg-number ---
          +- -s set-number -+

    +------------------------------------+
 ---¦                     +-----------+ +---¦
    +- 'default message' -¦           +-+
                          +--- arg ---+
                                    ¦
                            +-------+
```

Description

Use the **dspmsg** command in shell scripts to display messages from a message catalog to the person running the shell script program. It is an alternate to the message presenting capability of the **echo** command. The separate **dspcat** command (page 1.1.139) can display more than one message at a time, and is usually used from the command line.

The **dspmsg** command first searches the directory **/usr/lib/mbcs/msg/language.codeset/** for **catfile**. The **language.codeset** depends on the current locale. (The locale may be changed using the **LANG** environment variable.) For example, if the current locale is English, the default search directory would be **/usr/lib/mbcs/msg/En.pc850/**. If **catfile** is not found in the default search directory, **dspmsg** searches the current directory for **catfile**.

The **catfile** parameter specifies the catalog being accessed. The **-s set-number** parameter specifies a specific set within the message catalog. The default value for the set to be accessed is 1. The **msg-number** parameter specifies the message to be displayed from the catalog. If the message specified by the preceding parameters cannot be displayed, a default message, if supplied, is displayed.

The **arg** parameter specifies arguments that can be substituted into the message. Up to 10 arguments can be substituted. The specified **arg** parameters are inserted in place of the **%** characters within the message extracted from the catalog. Positional parameters are also supported (for information on positional parameters, see the **printf** subroutine in the *AIX Technical Reference*). The only allowable substitution is a string value; a non-string variable produces undefined results. The **arg** parameter can be included only when a default message is also included.

Note: Message catalog files are binary files which must be created by the **gencat** command.

Example
The following sh shell script fragment will display message 1 from set 2 in **/u1/stevea/errors.cat** if no parameters are supplied to the shell script. If message 1 from set 2 doesn't exist in **errors.cat**, the message

in quotes will be displayed.

```
  if [ $# -eq 0 ]
  then
        dspmsg =s 2 /ul/stevea/errors.cat 1 "Input invalid.  Please try again
  fi
```

Related Information

See the following commands: "echo" in topic 1.1.146, "dspcat" in topic 1.1.139, and "gencat" in topic 1.1.184.

See the **catopen** subroutine in the *AIX Technical Reference*.

See the "Message Catalog Generation" section of the "International Character Support" chapter of the *AIX Operating System Programming Tools and Interfaces*.

See the "Messages" chapter in the *AIX MBCS Guide*.

1.1.141 du

Purpose
Summarizes disk usage.

Syntax

```
      +-----------+   +------ . ------+
du ---¦ +-------+ +---¦ +-----------+ +---¦
      +-¦ -a -r +-+   +-¦ file      +-+
        ¦ -l -  ¦¦      ¦ directory ¦¦
       ¦+-------+¦     ¦+-----------+¦
       +---------+     +-------------+
```

Description
The **du** command gives the number of blocks in all **files** and (recursively) directories within each specified **directory**. By specifying the **-a** flag, you can also have the **du** command report the number of blocks in individual files. The block count includes the indirect blocks of each file and is in units of 1024 bytes. If you provide no **file** or **directory** name, the **du** command uses the current directory.

When encountering a symbolic link, **du** will report the size of the symbolic link itself, rather than reporting the size of the file or directory to which it points.

Notes:

1. If you do not specify the **-a** flag, the **du** command does not report on any **file**s.

2. If there are too many distinctly linked files, the **du** command counts the excess files more than once.

3. AIX allocates 4096-byte disk blocks and therefore **du** will always report file sizes multiples of 4 1024-byte blocks. For example, a file which is 2048 bytes long will be reported as using 4 blocks.

4. AIX uses a disk block allocation strategy such that no blocks are allocated for files of 384 or fewer bytes. **du** reports these files as using 0 disk blocks.

Flags

-a Displays disk use for each file.

-l Allocates blocks in files with multiple links evenly among the links. By default, a file with two or more links is counted only once.

-r Indicates inaccessible files and directories.

-s Displays only the grand total (for each of the specified files or directories given).

Examples

1. To summarize the disk usage of a directory tree and each of its subtrees:

```
du  /u/fran
```

For **/u/fran** and each of its subdirectories, this command displays the number of disk blocks that the files in the tree beneath it contain.

2. To display the disk usage of each file:

   ```
   du  -a  /u/fran
   ```

   This command displays the number of disk blocks contained in each file and subdirectory of **/u/fran**. The number beside a directory is the disk usage of that directory tree. The number beside a regular file is the disk usage of that file alone.

3. To display only the total disk usage of a directory tree:

   ```
   du  -rs  /u/fran
   ```

   This command displays only the sum total disk usage of **/u/fran** and the files it contains (**-s**). The **-r** flag tells **du** to display an error message if it cannot read a file or directory.

Related Information

See the following commands: "df" in topic 1.1.121 and "quot" in topic 1.1.349.

1.1.142 dump

Purpose
Dumps selected parts of an object file.

Syntax

```
        +------------+      +-------------------------+ +----------+
dump ---¦ -a  -g  -r +------¦            +---------+ +-¦          +---
        ¦ -b  -h  -s ¦ ¦    +- -z name --¦¦         +-+ +- +z num -+ ¦
      ¦ ¦ -c  -l  -t ¦ ¦  ¦             +- , num -+                 ¦
      ¦ ¦ -d  -L  -x ¦ ¦  +-------------------------------------------+
      ¦ ¦ -f  -o     ¦ ¦
      ¦ +------------+ ¦
      +----------------+

   +---------------+   +-------------------------+  +------------------------+
 --¦ +-----------+ +---¦         +----------+  +--¦          +-----------+ +--
   +-¦ -n sec  ² +-+   +- +t num -¦          +--+  +- +D num -¦           +-+
     ¦ -p        ¦ ¦   ¦         +- -t num -+               +- -D num -+
     ¦ -u          ¦   +-----------------------------------------------------
     ¦ -v       4 ¦
     +-----------+

 ------ file -----¦
                ¦
    +----------+
```

¦ Do not put a space between these items.
[2] Use **-n** only with **-h**, **-l**, **-r**, or **-s**.
¦ Use **-p** only with **-a**, or **-o**.
4 Do not use **-v** with **-s** or **-o**.

Description
The **dump** command dumps selected parts of the specified **file**. The command accepts object files and archive object files. It writes information in character, hexadecimal, octal, or decimal representation, as appropriate to format the information in a meaningful way.

Flags

You must use at least one of the following flags:

-a	Dumps the archive header of each member of each specified archive.
-b	Same as **-L**.
-c	Dumps the string table.
-d	Dumps the contents of the **.data** section.
-f	Dumps each file header.
-g	Dumps the global symbols in the archive symbol table.
-h	Dumps section headers.

-l	Dumps line number information.
-L	Dumps the contents of the **.lib** section. This section contains the shared library information.
-o	Dumps each optional header.
-r	Dumps relocation information.
-s	Dumps the contents of the sections.
-t	Dumps symbol table entries.
-x	Same as **-L**.

The following optional flags are also available:

-D num
Dumps the section number **num** or the range of sections starting at **num** and ending at the numbers specified by **+Dnum**.

+D num
Dumps sections in the range beginning either with the first section or beginning with the section specified by **-Dnum**.

-n sec
Dumps information pertaining only to the named section (**.text**, **.data**, **.lib**, **.init**, **.debug**, **.bss**, **comment**). This flag applies only to the **-h, -l, -r,** and **-s** flags. For example, if you want the section header information from the **.text** section, enter:

```
dump -h -n .text file
```

-p
Does not print the headers.

-t num
Dumps only the index symbol table entry specified with **num**. Use the **-t** flag with the **+t** flag to specify a range of symbol table entries.

+t num
Dumps the symbol table entry in the range that ends with **num**. The range starts at either the first symbol table entry or at the entry specified by the **-t** flag.

-u
Underlines the name of the **file**. May not work on some terminals and printers.

-v
Dumps the information in symbolic representation rather the numeric. You can use this flag with any of the above flags except **-s** or **-o**.

-z name[,num]
Dumps line number entries for **name** or a range of line number entries that starts at the specified number. You can use a blank to replace the comma that separates **name** and **num**.

+z num
Dumps all line numbers up to **num** when used with the **-z[,num]** flag.

Note: The line number information is **not** currently put in the symbol table of object files generated by the AIX compilers (C, VS FORTRAN and VS Pascal).

Related Information

See the following commands: "ar" in topic 1.1.23, "nm" in topic 1.1.298, "shlib2" in topic 1.1.422, and "size" in topic 1.1.427.

See the **a.out** and **ar** files in *AIX Operating System Technical Reference*.

1.1.143 dumpbsd

Purpose
Produces an incremental file system dump.

Syntax

```
                     +---------+
dumpbsd --- key ---¦ +-----+ +--- filesystem ---¦
                     +-¦ d s +-+
                       ¦ f W ¦
                      ¦¦ n w ¦¦
                      ¦¦ u   ¦¦
                      ¦+-----+¦
                      +-------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

Note: The **dumpbsd** command is not used to port information between AIX and BSD systems. The **tar** command is used for compatibility with BSD systems.

The **dumpbsd** command copies to magnetic tape all files changed after a certain date in the **filesystem**. The **key** specifies the date and other options about the dump. **Key** consists of characters from the set **0123456789fusdWn**.

0-9 This number is the 'dump level'. All files modified since the last date stored in the file **/etc/dumpdates** for the same file system at lesser levels will be dumped. If no date is determined by the level, the beginning of time is assumed; thus the option **0** causes the entire file system to be dumped.

f Place the dump on the next **argument** file instead of the tape. If the name of the file is "-," **dumpbsd** writes to standard output.

u If the dump completes successfully, write the date of the beginning of the dump on file **/etc/dumpdates**. This file records a separate date for each file system and each dump level. The format of **/etc/dumpdates** is readable by people, consisting of one free format record per line: file system name, increment level and **ctime**(3) format dump date. **/etc/dumpdates** may be edited to change any of the fields, if necessary.

s The size of the dump tape is specified in feet. The number of feet is taken from the next **argument**. When the specified size is reached, **dumpbsd** will wait for reels to be changed. The default tape size is 2300 feet.

d The density of the tape, expressed in BPI, is taken from the next **argument**. This is used in calculating the amount of tape used per reel. The default is 1600.

W **dumpbsd** tells the operator what file systems need to be dumped. This information is gleaned from the files **/etc/dumpdates** and **/etc/filesystems**. The **W** option causes **dumpbsd** to print out, for each file system in **/etc/dumpdates** the most recent dump date and level, and highlights those file systems that should be dumped. If the **W** option is set, all other options are ignored, and **dumpbsd** exits immediately.

w Is like **W**, but prints only those file systems which need to be dumped.

n Whenever **dumpbsd** requires operator attention, notify by means similar to a **wall**(1) all of the operators in the group "operator".

If no arguments are given, the **key** is assumed to be **9u** and a default file system is dumped to the default tape.

The **dumpbsd** command requires operator intervention on these conditions: end of tape, end of dump, tape write error, tape open error or disk read error (if there are more than a threshold of 32). In addition to alerting all operators implied by the **n** key, **dumpbsd** interacts with the operator on **dumpbsd**'s control terminal at times when **dumpbsd** can no longer proceed, or if something is grossly wrong. All questions **dumpbsd** poses **must** be answered by typing "yes" or "no", appropriately.

Since making a dump involves a lot of time and effort for full dumps, **dumpbsd** checkpoints itself at the start of each tape volume. If writing that volume fails for some reason, **dumpbsd** will, with operator permission, restart itself from the checkpoint after the old tape has been rewound and removed, and a new tape has been mounted.

The **dumpbsd** command tells the operator what is going on at periodic intervals, including usually low estimates of the number of blocks to write, the number of tapes it will take, the time to completion, and the time to the tape change. The output is verbose, so that others know that the terminal controlling **dumpbsd** is busy, and will be for some time.

To perform dumps, start with a full level 0 dump:

```
dumpbsd 0un
```

Next, dumps of active file systems are taken on a daily basis, using a modified Tower of Hanoi algorithm, with this sequence of dump levels:

3 2 5 4 7 6 9 8 9 9 ...

For the daily dumps, a set of 10 tapes per dumped file system is used on a cyclical basis. Each week, a level 1 dump is taken, and the daily Hanoi sequence repeats with 3. For weekly dumps, a set of 5 tapes per dumped file system is used, also on a cyclical basis. Each month, a level 0 dump is taken on a set of fresh tapes that is saved forever.

The operator must add the following stanzas to the file system to be dumped into **/etc/filesystems**.

mode The first stanza is the mode or type of file system described in these terms:

rw	read/write device
ro	read only device
sw	swap device
nm	file system not normally mounted
xx	ignore type

freq	The second stanza is the dump frequency in days must be supplied by giving a numerical value.

Files

/dev/rrp1g	Default file system to dump from.
/dev/rmt8	Default tape unit to dump to.
/etc/dumpdates	New format dump date record.
/etc/filesystems	Dump table: file systems and frequency.
/etc/group	To find group "operator".

Related Information
See the following commands: "restorebsd" in topic 1.1.372 and "dump" in topic 1.1.142.

1.1.144 dumpcfg

Purpose
Modify Dump space in VTOCs.

Syntax

```
                           +--------------+
/etc/dumpcfg --- device ---¦ +- delete -+ +---¦
                           +-¦          +-+
                             +-- size --+
```

Description
Dumpcfg is a utility that allows the system administrator to add, change, or delete dump partitions on a **VTOC**. The user must be root, and the device specified must exist.

If the user specifies a size value, then the device specified becomes a dump partition on the **VTOC** with the size specified. If the device specified is already an existing dump device, then the size of that partition is changed to be the size of the device specified (if possible). This is done by incorporating the free space surrounding the dump partition. If the device specified does not exist, than a new dump partition of the size specified is placed in the smallest available freespace partition possible.

If the user specifies delete, then the device specified is deleted from the **VTOC**, provided that the dump space is not currently the active dump partition in use by the kernel.

If neither a size nor delete is specified, then the **VTOC** for the disk on which the device specified resides is displayed.

On a 370 system, **dumpcfg** can also be used to add a dump partition to a previously unused physical disk. In this case, before running **dumpcfg**, you must run **dfmt** with the **-b/generic/boot1** option. This puts a valid **vtoc** and a valid boot block on the disk.

Examples
Assume a user has a dump device on chd00034. It is 20K blocks long, and it has 20K blocks free space following it. The user wants his dump space to be 30K blocks. The user should enter:

```
  dumpcfg /dev/chd00034 30000
```

Now assume a user has a dump device on hd4 on hdisk0. It is 20K blocks long, and it has 10K blocks free space following it, and no other free space on the device. Also assume the user has 40K blocks free space on another disk (hdisk1). That disk has hd33 and hd34 on it. If the user wants to increase the dump space to 40K, it must be put on hdisk1 in this case since there is not enough room on hdisk0. The user should do the following:

1. mknod a dev, **/dev/hd**35 for example, on hdisk 1

2. Enter **dumpcfg /dev/hd35 40000**

3. Edit **/etc/system** to change dumpdev in sysparms to be hd35. Reconfigure the kernel and reboot.

4. Once the system has been rebooted, run **dumpcfg /dev/hd4 delete**. Note that this could not be deleted until after the reconfiguration of the kernel because it was still the active dump device.

Related Information

See the command "minidisks" in topic 1.1.266, and the discussion of minidisk in *Installing and Customizing the AIX Operating System*.

1.1.145 dumpfs

Purpose
Dumps file system information.

Syntax

```
          +- filesystem -+
dumpfs ---¦              +---¦
          +--- device ---+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **dumpfs** command prints out information for the file system or special device specified. The listing is very long and detailed. This command is used for finding file system information such as the file system block size and minimum free space percentage.

Related Information
See the following commands: "mkfs" in topic 1.1.269 and "fsck, dfsck" in topic 1.1.177.

See **filesystems** and **fs** topics in *AIX Operating System Technical Reference.*

1.1.146 echo

Purpose
Writes its arguments to standard output.

Syntax

```
        +------+
echo ---¦      +--- string ---¦
        +- -n -+              ¦
                 +-----------+
```

Description

Note: The following description applies to **/bin/echo** and the **sh** built-in **echo** command. See page 1.1.100.5 for a description of the **csh** built-in **echo** command.

The **echo** command writes its arguments to standard output. The **string**s are separated by blanks and a new-line character follows the last **string**. Use **echo** to produce diagnostic messages in command files and to send data into a pipe.

The **echo** command recognizes the following escape conventions:

\b	Display a backspace character.
\c	Suppress the new-line character. Use as last option on command line; it ignores all characters after this option.
\f	Display a form-feed character.
\n	Display a new-line character.
\r	Display a carriage return character.
\t	Display a tab character.
\	Display a backslash character.
\num	Display an 8-bit character, whose ASCII value is the 1-, 2- or 3-digit octal number **num**. The first digit of **num** must be a zero.

Flag

-n No newline is added to the output.

Examples

1. To write a message to standard output:

```
echo Please insert diskette . . .
```

2. To display a message containing special characters:

```
echo "\n\n\nI'm at lunch.\nI'll be back at 1:00."
```

This skips three lines and displays the message:

```
I'm at lunch.
I'll be back at 1:00.
```

Note: You must quote the message if it contains escape sequences like **\n**. Otherwise, the shell treats the **** specially. See page 1.1.420.8 for details about quoting.

3. To use **echo** with pattern-matching characters:

   ```
   echo The back-up files are: *.bak
   ```

   This displays the message **The back-up files are:** followed by the file names in the current directory ending with **.bak**.

4. To add a single line of text to a file:

   ```
   echo Remember to set the shell search path to $PATH. >>notes
   ```

   This adds the message to the end of the file **notes** after the shell substitutes the value of the shell variable **PATH**.

5. To write a message to the standard error output:

   ```
   echo Error: file already exists. >&2
   ```

   Use this in shell procedures to write error messages. If the **>&2** is omitted, the message is written to the standard output. For details about this type of file redirection, see "Input and Output Redirection Using File Descriptors" in topic 1.1.420.20.

Related Information

See the following commands: "csh" in topic 1.1.100, "sh, Rsh" in topic 1.1.420 and "dspmsg" in topic 1.1.140.

1.1.147 ed, red

Purpose
Starts a line editing program.

Syntax

```
one of
+-----+   +------+   +--------+
¦  ed +---¦      +---¦        +---¦
¦ red ¦   +- -p -+   +- file -+
+-----+
```

Description

The **ed** command is a line editing program that works on only one file at a time by copying it into a temporary file buffer and making changes to that copy. The **ed** command does not alter the file itself until you use the **w** (write) subcommand. You can specify on the command line the **file** you want to edit, or you can use the **e** (edit) subcommands.

When the **ed** command reads a new file into the buffer, the contents of that file replaces the buffer's previous contents, if any.

The **red** command is a restricted verstion of the **ed** command (see "sh, Rsh" in topic 1.1.420). With the **red** command, you can edit only files that reside in the current directory, or in the **/tmp** directory, and you cannot use the **!AIX-cmd** subcommand (see page 1.1.147.2).

An **ed** subcommand consists of zero, one, or two **addresses**, which are followed by a single-character subcommand, that may be followed by parameters. These addresses specify one or more lines in the buffer. Because every subcommand has default addresses, you frequently do not need to specify addresses.

The **ed** program operates in one of two modes, **command mode** or **text mode**. In command mode, **ed** recognizes and executes subcommands. In text mode, **ed** adds text to the file buffer but does not recognize subcommands. To leave text mode, enter a . (dot) alone at the beginning of a line.

Subtopics

1.1.147.1 Pattern Matching

The **ed** command supports a limited form of special pattern-matching characters that you can use as **regular expressions** (**REs**) to construct pattern strings. You can use these patterns in addresses to specify lines and in some subcommands to specify portions of a line.

Subtopics

1.1.147.1.1 Regular Expressions (REs)

The following REs match a single character:

char Any valid character (other than one of the special pattern-matching characters) matches itself.

. A period (matches any single character except for the new-line character.

[**string**] A **string** enclosed in square brackets ([ ]) matches any one character in that string. The **string** can contain Japanese characters. Certain pattern-matching characters have special meanings within square brackets:

- **^** If the first character of **string** is a circumflex, the RE ([(^)**string**]) matches any character **except** the characters in **string** and the new-line character. A (^) has this special meaning **only** if it occurs first in the string.

- **-** You can use a - (minus) to indicate a range of consecutive ASCII characters according to the current collating sequence. For example, **[a-f]** might be equivalent to **[abcdef]** or **[aAbBcCdDeEfF]** or **[aáàbcdeéèf]**. The collating sequence is defined by the environment variable **LC_COLLATE** or **LANG**. See *Managing the AIX Operating System* for more information. A collating sequence may define equivalence classes for characters. For example, if three charactersc/-**e**, **é**, and **è**-are equivalent, the following expressions identify the same sequence of characters:

  ```
  [a-e]
  [a-è]
  ```

  The minus character loses its special meaning if it occurs first ([-**string**]), if it immediately follows an initial circumflex ([(^)-**string**]), or if it appears last ([**string**-]) in the string.

- **]** When the right square bracket (]) is the first character in the string ([]**string**]) or when it immediately follows an initial circumflex ([^]**string**]), it is treated as a part of the string rather than as the string terminator.

- **[[:charclass:]]** Character class expressions allow you to match a character type. The system interprets this type of expression according to the current class definition. The class definition depends on the current system language. You cannot use character class expressions in range expressions.

  To use a character class expression, enclose it in brackets and colons:

  [[:**charclass**:]]

The following character classes are supported:

[:upper:]	Uppercase letters
[:lower:]	Lowercase letters
[:alpha:]	Uppercase and lowercase letters
[:digit:]	Digits
[:xdigit:]	Hexadecimal digits
[:punct:]	Punctuation character (neither a control character nor alphanumeric)
[:space:]	Space, tab, carriage return, new-line, vertical tab, or form-feed character
[:print:]	Printing character

For example, to match any uppercase letter or digit, use the following expression:

```
  [[:upper:] [:digit:]]
```

[[=c=]] Equivalence class; that is, any collation element defined as having the same relative order in the current collation sequence as **c**. For example, if A and a belong to the same equivalence class, both **[[=A=]b]** and **[[=a=]b]** are equivalent to **[Aab]**.

[[.cc.]] Collating symbol. Multi-character collating elements must be represented as collating symbols to distinguish them from single-character elements. For example, if the string **ch** is a valid collating element, **[[.ch.]]** is treated as an element matching the same string of characters, while **ch** is treated as a simple list of **c** and **h**. If the string is not a valid collating element in the current collating sequence definition, the symbol is treated as an invalid expression.

\sym A \ (backslash) followed by a special pattern-matching character matches the special character itself (as a literal character). These special pattern-matching characters are:

**. * [**	Always special **except** when they appear within square brackets ([]).
^	Special at the **beginning** of an entire pattern or when it immediately follows the left bracket of a pair of brackets ([(^)....]).
$	Special at the **end** of an entire pattern.

In addition, the character used to delimit an entire pattern is special for that pattern. (For example, see how / (slash) is used in the **g** subcommand on page 1.1.147.2.) This special pattern matching is described in more detail below.

1.1.147.1.2 Forming Patterns

The following rules describe how to form patterns from REs:

1. An RE that consists of a single, ordinary character matches that same character in a string.

2. An RE followed by an ***** (asterisk) matches zero or more occurrences of the character that the RE matches. For example, the following pattern:

   ```
   ab*cd
   ```

   matches each of the following strings:

   ```
   acd
   abcd
   abbcd
   abbbcd
   ```

   but not the following string:

   ```
   abd
   ```

   If there is any choice, the longest matching leftmost string is chosen. For example, given the following string:

   ```
   122333444
   ```

   the pattern **.*** matches **122333444**, the pattern **.*3** matches **122333**, and the pattern **.*2** matches **122**.

3. An RE followed by:

\\{m\\}	Matches exactly **m** occurrences of the character matched by the RE.
\\{m,\\}	Matches at least **m** occurrences of the character matched by the RE.
\\{m,n\\}	Matches any number of occurrences of the character matched by the RE from **m** to **n** inclusive.

   m and **n** must be integers from 0 to 255, inclusive. Whenever a choice exists, this pattern matches as many occurrences as possible.

4. You can combine REs into patterns that match strings containing that same sequence of characters. For example, **AB\*CD** matches the string **AB*CD**, and **[[:alpha::]]*[[:digit:]]*** matches any string that contains any combination of alphabetic characters (including no occurrences), followed by any combination of numerals (including no occurrences).

5. The character sequence **\\(pattern\\)** marks a subpattern that matches the same string it would match if it were not enclosed.

6. The characters \num match the same string of characters that a subpattern matched earlier in the pattern (see the preceding discussion of item 5). **num** is a digit. The pattern \num matches the string matched by the **num**th subpattern, counting from left to right.

For example, the following pattern:

```
  \(A\)\(B\)C\2\1
```

matches the string **ABCBA**. You can nest subpatterns.

1.1.147.1.3 Restricting Which Patterns Match

A pattern, which can contain Japanese characters, can be restricted to match only the first segment of a line, the final segment, or both:

1. A ^ (circumflex) at the beginning of a pattern causes the pattern to match only a string that begins in the first character position on a line.

2. A **$** (dollar sign) at the end of a pattern causes that pattern to match only a string that ends with the last character (not including the new-line character) on a line.

3. The construction ^**pattern$** restricts the pattern to matching only an entire line (The pattern must begin at the first character position and end at the last character position).

The null pattern (that is, //) duplicates the previous pattern.

1.1.147.2 Addressing

There are three types of **ed** command addresses: line number addresses, addresses relative to the current line, and pattern addresses. The ***current line*** (usually the last line affected by a command) is the point of reference in the buffer. The current line is the default address for several **ed** commands. (See "Subcommands" to find out how each subcommand affects the current line.)

Following are guidelines for constructing addresses:

1. . (dot) addresses the current line.

2. **$** (dollar sign) addresses the last line of the buffer.

3. **n** addresses the **n**th line of the buffer.

4. '**x** addresses the line marked with a lowercase ASCII letter, **x**, by the **k** subcommand (see page 1.1.147.2).

5. /**pattern**/ (a pattern enclosed in slashes) addresses the next line that contains a matching string. The search begins with the line after the current line and stops when it finds a match for the pattern. If necessary, the search moves to the end of the buffer, wraps around to the beginning of the buffer, and continues until it either finds a match or returns to the current line.

6. **?pattern?** (a pattern enclosed in question marks) addresses the previous line that contains a match for the pattern. The **?pattern?** construct, like /**pattern**/, can search the entire buffer, but it does so in the opposite direction.

7. An address followed by +**n** or -**n** (a plus sign or a minus sign followed by a decimal number) specifies an address plus or minus the indicated number of lines. (The + sign is optional.)

8. An address that begins with + or - specifies a line relative to the current line. For example, **-5** is equivalent to **.-5** (five lines above the current line).

9. An address that ends with - or + specifies the line immediately before (-) or immediately after (+) the addressed line. Used alone, the - character addresses the line immediately before the current line. The + character addresses the line immediately after the current line; however, the + character is optional. The + and - characters have a cumulative effect; for example, the address -- addresses the line two lines above the current line.

10. For convenience, a , (comma) stands for the address pair **1,$** (first line through last line) and a ; (semicolon) stands for the pair **.,$** (current line through last line).

Commands that do not accept addresses regard the presence of an address as an error. Commands that do accept addresses can use either given or default addresses. When given more addresses than it accepts, a command uses the last (rightmost) one(s).

In most cases, commas (**,**) separate addresses (for example **2,8**). Semicolons (**;**) also can separate addresses. A semicolon between addresses causes **ed** to set the current line to the first address and then calculate the second address (for example, to set the starting line for a search based on guidelines 5 and 6 above). In a pair of addresses, the first must be numerically smaller than the second.

For many purposes, you may prefer to use a different editor that has different features:

"edit" in topic 1.1.149, a simple line editor for novice or casual users
"sed" in topic 1.1.415, a stream editor often used for writing programs
"ex" in topic 1.1.158, an extended (line) editor with interactive subcommand features
"vi, vedit, view" in topic 1.1.522, a visual (screen) editor that also accesses **ex** line editing features while letting you view the text.

A list of **ed** size limitations follows. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the following character limits can be reduced by as much as 50%, depending on the character code set being used.

64 characters per file name
512 characters per line (although there is currently a system-impose limit of 255 characters per line entered from the keyboard).
256 characters per global subcommand list

The maximum number of lines depends on the amount of memory available to you. The maximum file size depends on the amount of physical data storage (disk or tape drive) available or on the maximum number of lines permitted in user memory.

Subcommands

In most cases, only one **ed** subcommand can be entered on a line. The exceptions to this rule are the **p** and **l** subcommands, which can be added to any subcommand except **e**, **f**, **r**, or **w**. The **e**, **f**, **r**, and **w** subcommands accept file names as parameters. The **ed** program stores the last file name used with a subcommand as a default file name. The next **e**, **f**, **r**, or **w** subcommand given without a file name uses the default file name.

The **ed** program responds to an error condition with one of two messages: **?** (question mark) or **?file**. When **ed** receives an INTERRUPT signal (**Ctrl-C**), it displays a **?** and returns to command mode. When the **ed** program reads a file, it discards ASCII NULL characters and all characters after the last new-line character.

In the following list of **ed** subcommands, default addresses are shown in parentheses. Do not enter the parentheses. The address . (period) refers to the current line. When a . is shown in the first position on an otherwise empty line, it is the signal to return to command mode.

(.)**a**
text
.

The **a** (append) subcommand adds text to the buffer after the addressed line. The **a** subcommand sets the current line to the last inserted line, or, if no lines are inserted, to the addressed line. Address **0** causes the **a** subcommand to add text at the beginning of the buffer.

(.)**c**

text
.   The **c** (change) subcommand deletes the addressed lines, then replaces them with new input. The **c** command sets the current line to the last new line of input, or, if there are no lines, to the first line that is not deleted.

(.,.)**d** The **d** (delete) subcommand removes the addressed lines from the buffer. The line after the last line deleted becomes the current line. If the deleted lines were originally at the end of the buffer, the new last line becomes the current line.

e file The **e** (edit) subcommand first deletes any contents from the buffer. The command then loads another file into the buffer, sets the current line to the last line of the buffer, and displays the number of characters read in to the buffer. If the buffer has been changed since its contents were last saved (with the **w** subcommand), the **e** command displays **?** before it clears the buffer.

The **e** subcommand stores **file** as the default file name to be used, if necessary, by subsequent **e**, **r**, or **w** subcommands. (See the **f** subcommand.)

When the **!** character replaces **file**, the **e** subcommand takes the rest of the line as an AIX shell (**sh**) command and reads the command output. The **e** subcommand does not store the name of the shell command as a default file name.

E file The **E** (edit) subcommand works like the **e** subcommand, with one exception: **E** does not check for changes made to the buffer since the last **w** subcommand.

f [file] The **f** (filename) subcommand changes the default file name (the stored name of the last file used) to **file**, if **file** is given. If **file** is not given, the **f** subcommand prints the default file name.

(1,**$**)**g/pattern/subcmd-list** The **g** (global) subcommand first marks every line that matches the **pattern** Then, for each marked line, this subcommand sets the current line to that line and executes **subcmd-list**. A single subcommand, or the first subcommand of a list, should appear on the same line with the **g** subcommand; subsequent subcommands should appear on separate lines. Except for the last line, each line containing a subcommand should end with a \.

The **subcmd-list** can include the **a**, **i**, and **c** subcommands and their input. If the last command in **subcmd-list** would normally be the **.** (dot) that ends input mode, the **.** (dot) is optional. If there is no **subcmd-list**, the **ed** command displays the current line. The **subcmd-list** can include any subcommands except the **g**, **G**, **v**, or **V** subcommands.

Note: The **g** subcommand is similar to the **v** subcommand, which executes **subcmd-list** for every line that does not contain a match for the pattern.

(1,**$**)**G/pattern/**	The interactive **G** (Global) subcommand first marks every line that matches the pattern. The command displays the first marked line, sets the current line to that line, and waits for a subcommand. The **G** subcommand accepts any but the following **ed** subcommands: **a**, **c**, **i**, **g**, **G**, **v**, and **V**. After the subcommand finishes, **G** displays the next marked line, and so on. The **G** command takes a new-line character as a null subcommand. An & causes **G** to execute the previous subcommand again, if there was one. Subcommands executed within the **G** subcommand can address and change any lines in the buffer. The **G** subcommand can be terminated by pressing the INTERRUPT key (**Ctrl-C**). **Note:** The **G** subcommand complements the **V** subcommand, which marks lines that do not match the pattern.
h	The **h** (help) subcommand gives a short explanation (help message) for the most recent **?** diagnostic or error message.
H	The **H** (Help) subcommand causes **ed** to display the help messages for all subsequent **?** diagnostics. **H** also explains the previous **?** if there was one. **H** alternately turns this mode on and off; it is initially off.
(.)**i** **<***text***>** **.**	The **i** (Insert) subcommand inserts text before the addressed line and sets the current line to the last inserted line. If no lines are inserted, **i** sets the current line to the addressed line. This subcommand differs from the **a** subcommand only in the placement of the input text. Address 0 is not legal for this subcommand.
(.,.+1)**j**	The **j** (join) subcommand joins contiguous lines by removing the intervening new-line characters. If given only one address, **j** does nothing. (For splitting lines, see the **s** subcommand.)
(.)**kx**	The **k** (mark) subcommand marks the addressed line with name **x**, which must be a lowercase ASCII letter. The address '**x** (single quotation mark before the marking character) then addresses this line. The **k** subcommand does not change the current line.
(.,.)**l**	The **l** (list) subcommand displays the addressed lines. The **l** subcommand wraps long lines and, unlike the **p** subcommand, represents non-printing characters, either with mnemonic overstrikes or in octal notation. An **l** subcommand may be appended to any **ed** subcommand except **e**, **f**, **r**, or **w**.
(.,.)**ma**	The **m** (move) subcommand repositions the addressed lines. The first moved line follows the line addressed by **a**. Address **0** for **a** causes **m** to move the addressed lines to the beginning of the file. Address **a** cannot be one of

the lines to be moved. The **m** subcommand sets the current line to the last moved line.

(.,.)**n**	The **n** (number) subcommand displays the addressed lines, each preceded by its line number and a tab character (displayed as blank spaces); **n** leaves the current line at the last line displayed. An **n** subcommand may be appended to any **ed** subcommand except **e**, **f**, **r**, or **w**.
(.,.)**p**	The **p** (print) subcommand displays the addressed lines and sets the current line set to the last line displayed. A **p** subcommand may be appended to any **ed** subcommand except: **e**, **f**, **r**, or **w**. For example, the subcommand **dp** deletes the current line and displays the new current line.
P	The **P** subcommand turns on or off the **ed** prompt string ***** (asterisk). Initially, the **ed** prompt string is turned off.
q	The **q** (quit) subcommand ends the **ed** program. Before ending the program, **q** checks to determine whether the buffer has been written to a file since the last time it was changed. If not, **q** displays the **?** message.
Q	The **Q** (Quit) subcommand ends the **ed** program without checking for changes to the buffer since the last **w** subcommand was entered (compare with the **q** subcommand).
(.)**r file**	The **r** (read) subcommand reads a file into the buffer after the addressed line but does not delete the previous contents of the buffer. If you do not specify **file**, the command reads the default file, if any, into the buffer (see the **e** and **f** subcommands). The **r** command does not change the default file name. Address **0** causes **r** to read a file in at the beginning of the buffer. If no address is given, the file is read in after the current line. After it reads a file successfully, the **r** command displays the number of characters read into the buffer and sets the current line to the last line read. If **!** (exclamation point) replaces **file** the **r** subcommand takes the rest of the line as an AIX shell (**sh**) command whose output is to be read. The **r** subcommand does not store the names of shell commands as default file names.
(.,.)**s/pattern/replacement/** (.,.)**s/pattern/replacement/g**	The **s** (substitute) subcommand searches each addressed line for a string that matches the pattern and then replaces the string with the specified **replacement** string. Without the global indicator **g**, the **s** command replaces only the first matching string on each addressed line. With the **g** indicator, the **s** command replaces every occurrence of the matching string on each addressed line. If **s** does not find a match for the pattern, it returns the error message **?**. Any character except a space or a new-line character can separate (delimit) the **pattern** and **replacement**. The **s** subcommand sets the current line to the last line changed.

An (**&**) ampersand in the **replacement** string is a special symbol that has the same value as the **pattern** string. So, for example, the subcommand **s/are/&n't/** has the same effect as the subcommand **s/are/aren't/** and replaces **are** with **aren't** on the current line. A backslash before the ampersand (\**&**) removes this special meaning of **&** in **replacement**.

A subpattern is part of a pattern enclosed by the delimiting characters \\(and \\); the pattern works as if the enclosing characters were not present. In **replacement**, the characters \n refer to strings that match subpatterns; **n**, a decimal number, refers to the **n**th subpattern, counting from the left. (For example, **s/\\(t\\)\\(h\\) \\(e\\)/t\\1\\2ose)** replaces **the** with **those** if there is a match for the pattern **the** on the current line.) Whether subpatterns are nested or in a series, \n refers to the **n**th occurrence, counting from the left, of the delimiting characters, \\).

The **%** (percent sign) character, when used by itself as **replacement**, causes the **s** command to use the previous **replacement** again. The **%** character does not have this special meaning if it is part of a longer **replacement** or if it is preceded by a \\.

Lines may be split by substituting new-line characters into them. In **replacement**, the sequence \Enter quotes the new-line character (not displayed) and moves the cursor to the next line for the remainder of the string. New lines cannot be substituted as part of a **g** or **v** subcommand list.

(.,.)**ta** The **t**(transfer) subcommand inserts a copy of the addressed lines after address **a**. The **t** subcommand accepts address **0** (for inserting lines at the beginning of the buffer). The **t** subcommand sets the current line to the last line copied.

u The **u** (undo) subcommand restores the buffer to the state it was in before it was last modified by an **ed** subcommand. The commands that the **u** command can undo are **a**, **c**, **d**, **g**, **G**, **i**, **j**, **m**, **r**, **s**, **t**, **u**, **v**, and **V**.

(1,**$**)**v/pattern/subcmd-list** The **v** subcommand executes the subcommands in **subcmd-list** for each line that does not contain a match for the pattern. If you enter & as the subcmd-list, this executes the last substitution command entered.

Note: The **v** subcommand is a complement for the global subcommand **g**, which executes **subcmd-list** for every line that does contain a match for the pattern.

(1,$)**V/pattern//** The **V** subcommand first marks every line that does not match the pattern. The command then displays the first marked line, the current line to that line, and waits for a subcommand.

Note: The **V** subcommand complements the **G** subcommand, which marks the lines that do match the pattern.

(1,**$**)**w file**	The **w** (write) subcommand copies the addressed lines from the buffer to the file named in **file**. If no addressed lines are given, this subcommand copies the entire buffer. If the file does not exist, the **w** subcommand creates it with permission code 666 (read and write permission for everyone), unless the **umask** setting specifies another file creation mode. (For information about file permissions, see "umask" in topic 1.1.490 and "chmod" in topic 1.1.67.) The **w** subcommand does not change the default file name (unless **file** is the first file name used since you started **ed**). If you do not provide a file name, **ed** uses the default file name, if any (see the **e** and **f** subcommands). The **w** subcommand does not change the current line. If the **ed** command successfully writes the file, it displays the number of characters written. When **!** (exclamation point) replaces **file**, the **ed** program takes the rest of the line as an AIX shell (**sh**) command whose output is to be read; the **w** subcommand does not save shell command names as default file names. **Note:** Address **0** is not legal address for the **w** subcommand. Therefore, it is not possible to create an empty file with the **ed** command.
(**$**)**=**	With a numeric address, = displays the associated line number. Without an address, the = (equal sign) subcommand displays the current line number. With the address **$**, the = subcommand displays the number of the last line in the buffer. The = subcommand does not change the current line and cannot be included in a **g** or **v** subcommand list.
!AIX-cmd	The **!** (exclamation point) subcommand allows AIX commands to be run from within the **ed** command. Anything following **!** on an **ed** subcommand line is interpreted as an AIX command. Within the text of that command string, **ed** replaces the unescaped character **%** with the current file name, if there is one. When used as the first character of a shell command (after the **!** that runs a subshell) the **ed** command replaces the **!** character with the previous AIX command; for example, the command **!!** repeats the previous AIX command. If the AIX command interpreter (the **sh** command), expands the command string, **ed** echoes the expanded line. The **!** subcommand does not change the current line.
num **+num** **-num**	The **ed** command interprets a number alone on a line as an address and displays the addressed line. Addresses can be absolute (line numbers or **$**) or relative to the current line (**+num** or - **num**). Entering a new-line character (a blank line) is equivalent to entering **+1p** and is useful for stepping forward through the buffer

one line at a time.

Flags

-	Suppresses character counts that the editor displays with the **e**, **r**, and **w** subcommands; suppresses diagnostic messages for the **e** and **q** subcommands; and suppresses the **!** prompt after a **!AIX-cmd** subcommand.
-p string	Sets the editor prompt to **string**. The default for **string** is null (no prompt).

Files

/tmp/e#	Temporary file; # is the process number.
ed.hup	Work is saved in this file the terminal hangs up while the the **ed** command is running.

Related Information

See the following commands: "grep, egrep, fgrep" in topic 1.1.193, "sed" in topic 1.1.415, "sh, Rsh" in topic 1.1.420, "stty, STTY" in topic 1.1.447, and "regcmp" in topic 1.1.367.

See the **regexp** subroutine in *AIX Operating System Technical Reference*.

See the **environment** miscellaneous facility in *AIX Operating SystemTechnical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

See "Programming for an MBCS Environment" in *AIX Operating System Programming Tools and Interfaces*.

1.1.148 edconfig

Purpose
Edits values in a **sendmail** program configuration file.

Syntax

/usr/lib/edconfig --- **file** ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **edconfig** command allows you to edit a specified configuration file for the **sendmail** program. It provides a menu interface that allows you to change some of the characteristics defined in the configuration file. To change other characteristics, you must use a text editor. You must have superuser authority to edit the configuration file that the system uses (**/usr/adm/sendmail/sendmail.cf**). The **edconfig** command allows you to define or change the following types of entries in the configuration file:

The content of the **host name** class and macro

The **domain name** macro

The four classes that define the separate tokens of the domain nam

Configuration options (with help information) for

- Operational logging level
- Default delivery mode
- Alias file path
- Statistics file path
- Queue file path
- Maximum mail retention time in queue
- Queue uses of expensive mailers

Configuration file revision level

The **file** parameter provides the path name of the configuration file that you want to edit. The file must be in the format of the configuration file that is supplied with the operating system. The program searches for comment lines in that file of the form **#parameter** to locate the information in that file concerning **parameter**. For example, the comment line **#Or** precedes the line that defines the read timeout option.

To change parameters in the standard **sendmail** configuration file, enter the following command (while operating with superuser authority):

```
/usr/lib/edconfig /usr/adm/sendmail/sendmail.cf
```

The program reads the contents of the configuration file into memory. It then displays a menu to help you select what to change. All changes are made only to the copy of the file in memory until you choose to exit and write the changes to the file. You can also exit without writing changes. The program provides information with each step in the menus to help you decide how to enter the correct information. However, you should be familiar with the **sendmail** program and its use before changing information in the configuration file.

Files

/usr/lib/edconfig	The **edconfig** program.
/usr/lib/edconfig.hf	A text file containing the help information that **edconfig** program displays.
/usr/adm/sendmail/sendmail.cf	The configuration file for the **sendmail** program.

Related Information

See the following command: "sendmail, mailq, newaliases" in topic 1.1.417.

See the chapter about managing the mail system in *Managing the AIX Operating System*.

See the entry for **sendmail.cf** in *AIX Operating System Technical Reference*.

1.1.149 edit

Purpose
Starts a line editor that is designed for the new user.

Syntax

```
        +------+
edit ---¦      +-- file --¦
        +- -r -+          ¦
                 +-------+
```

Description
The **edit** command starts a line editor that is designed for beginning users. It is a simplified version of the **ex** command (see "ex" in topic 1.1.158). To edit the contents of a file, enter:

edit file

If **file** is the name of an existing file, the **edit** command copies the file to a buffer and displays the number of lines and characters in it. (Both the file and the file name can contain Japanese characters.) Then it displays a colon prompt (**:**) to show that it is ready to read subcommands from standard input. If **file** does not already exist, the **edit** command tells you this, but still stores the name as the current file name. You can give more than one **file** name, in which case the **edit** command copies the first file into its buffer and stores the remaining file names in an argument list for later use.

The **edit** command operates in one of two modes: command mode and text entry mode. In command mode, the **edit** command displays the colon prompt to show you that it is ready to accept **edit** subcommands. In text entry mode, **edit** places all input into its editing buffer. The general format of an **edit** subcommand is as follows:

[**addr]subcommand [parameters] [count]**

The **addr** can be a line number or a pattern to be matched or, in some cases, a range of line numbers or patterns. To specify a range, separate two line numbers or **pattern**s with a comma or a semicolon (for example, **1,5** or **1;5**). In a range, the second address must refer to a line that follows the first addressed line in the range. If you do not specify **addr**, the **edit** command works on the **current line**. If you add a numeric **count** to most subcommands, **edit** works on the specified number of lines.

For most subcommands, the last line affected becomes the new current line. That means, for example, that after the **edit** command reads a file into its buffer, the last line in the file becomes the current line.

Subtopics

1.1.149.1 Addressing Lines Within a File

The simplest way to address a line within a file is to use its line number. But this can be unreliable because line numbers change when you insert and delete lines. The **edit** command provides a way to search through the buffer for strings. Given the address

/pattern/

the **edit** command searches forward for **pattern**, while given

?pattern?

it searches backwards for **pattern**. (The **pattern** can contain Japanese characters.) If a forward search reaches the end of the buffer without finding **pattern**, the search continues from the beginning of the file until it reaches the current line. A backwards search does just the reverse.

The following characters have special meanings in forward and backward searches:

^	Matches the beginning of a line.
$	Matches the end of a line.

Thus, you can use **/^pattern/** to search for patterns at the beginning of a line, and **/pattern$/** to search for patterns at the end of the line.

The current line has a symbolic name, dot (**.**), and the last line in the buffer has a symbolic name, dollar sign (**$**), that you can use in addresses. These symbolic names are useful when working with a range of lines. For example,

```
  .,$print
```

displays all lines from the current line to the last line in the buffer. You can combine these symbols with numbers to reference lines, so that **$-5** refers to the fifth line from the last and **.+20** refers to the line 20 lines beyond the current line. You can also use these symbols with the **=** (equal) command to find out the line number of the current line or the last line, as follows:

```
  .=
  $=
```

To view the next line in the buffer, press the **Enter** key. To display the next half-screen of lines, press **Ctrl-D**.

Note: Do not confuse the meaning of **$** in text patterns (end of line) with its meaning in addresses (last line).

1.1.149.2 The Family of Editors

The **edit** command is part of a family of editors that includes **edit**, **ex**, and **vi**. The **edit** command starts a simple line editor that is designed for beginning users. It is a simplified version of the **ex** editor. After you become more experienced with the **edit** editor, you may want to try the advanced features of one of the other editors in the family. Because the **edit** editor is part of a family of editors, you can apply your knowledge of this editor to the other editors in the family.

The **ex** editor is a powerful interactive line editor. The **edit** subcommands work the same way in **ex**, but the editing environment is somewhat different. For example in **edit**, only the characters ^, **$**, and \ have special meanings as pattern-matching characters; however, several additional characters also have special meanings in **ex**. For more information on the **ex** editor, see "ex" in topic 1.1.158.

The **vi** editor is a display-based editor designed for experienced users who do a lot of editing at their display. It contains many of the advanced features of the **ex** editor, but focuses on the display editing portion of **ex**. The **edit** editor prevents you from accidentally entering **vi**'s two alternative modes of editing, the open mode and the visual mode. For more information on the **vi** editor, see "vi, vedit, view" in topic 1.1.522.

Flag

-r Recovers **file** after an editor or system crash.

Subcommands

You can enter most **edit** subcommands as either a complete word or an abbreviation. In the following list, a subcommand abbreviation appears in parentheses. Unless noted otherwise, all subcommands work by default on the current line. When the **edit** command displays the colon (:) prompt, you can enter these subcommands:

[addr]append (**a**)

text

. Reads the input *text* into the file being edited, placing **text** after the line at the address specified by **addr**. If you specify address 0, **edit** places the text at the beginning of the buffer. To return to command mode, enter a line with only a . (period) in the first position.

[addr1[,addr2]]change (**c**)

text

. Replaces the specified line or lines with the input *text*. If any lines are input, the last input line becomes the new current line.

[addr1[,addr2]]delete [buffer] (**d**)

Removes the specified line or lines from the editing buffer. The line following the last deleted line becomes the current line. If you specify a **buffer** by giving a letter from **a** to **z**, the specified lines are saved in that buffer or, if the letter is uppercase, appends the lines to that buffer.

edit file (**e**) Begins an editing session on a new file. The editor first checks to see if the buffer has been modified (edited) since the last **write** subcommand was entered. If it has, **edit** issues a warning and cancels the **edit** subcommand. Otherwise, it deletes the complete contents of the editor buffer, makes the named file the current file, and displays the new file name. After ensuring that this file can be edited, it reads the file into its buffer. If the **edit** command reads the file without error, it displays the number of lines and characters that it read. The last line read becomes the new current line.

file (**f**) Displays the current file name along with the following information about it:

Whether it has been modified since the last **write** subcommand was entered.
What the current line is.
How many lines are in the buffer.
What percentage of the way through the buffer the current line is.

file file Changes the name of the current file to **file**. The **edit** command considers this file not edited.

[addr1[,addr2]]global/pattern/cmds (**g**)

Marks each of the specified lines that matches **pattern**. Then the **edit** command carries out the specified subcommands (**cmds**) on each marked line.

A single **cmd** or the first **cmd** in a subcommand list appears on same line as **global**. The remaining **cmd**s must appear on separate lines, where each line (except the last) ends with a \ (backslash). The default subcommand is **print**.

The list can include the **append**, **insert**, and **change** subcommands and their associated input. In this case, if the ending period comes on the last line of the command list, you may omit it. The **undo** subcommand and the **global** subcommand itself, however, may not appear in the command list.

[addr]insert (**i**)

text

. Places the given text before the specified line. The last line input becomes the current line. Otherwise, the current line does not change.

[addr1[,addr2]]move *addr3* (**m**)

Repositions the specified line or lines to follow *addr3*. The first line moved becomes the current line.

next (**n**) Copies the next file in the command-line argument list to the buffer for editing.

[addr1[,addr2]]number (**nu**)

Displays each specified line or lines preceded by its buffer line number. The last line displayed becomes the current line.

preserve

Saves the current editor buffer as though the system had just crashed. Use this command when a **write** subcommand has resulted in an error, and you do not know how to save your work.

[addr1[,addr2]]print (**p**)

Displays the specified line or lines. The last line displayed becomes the current line.

[addr]put buffer (**pu**)

Retrieves the contents of the specified buffer and places it after **addr**. If you do not specify a buffer, the **edit** command restores the last deleted or yanked text. Thus you can use this subcommand together with the **delete** command to move lines or with the **yank** command to duplicate lines between files.

quit (**q**)

quit! (**q!**) Ends the editing session.

Note: The **quit** command does not write the editor buffer to a file. However, if you have modified the contents of the buffer since entering the last **write** command, **edit** displays a warning message and does not end the session. In this case, either use the **quit!** subcommand to discard the buffer or the **write** command to the buffer and then use the **quit** command.

recover file Recovers **file** from the system save area. Use this command after a system crash or after using a **preserve** subcommand.

[addr1[,addr2]]substitute/pattern/repl/ (**s**)

[addr1[,addr2]]substitute/pattern/repl/g

Replaces on each specified line the **first** instance of **pattern** with the replacement pattern **repl**. If you add the **g** flag, it replaces **all** instances of **pattern** on each specified line.

undo (**u**) Reverses the changes made in the buffer by the last buffer editing subcommand. Note that **global** subcommands are considered a single subcommand to an **undo** command. You cannot **undo** a **write** or an **edit** command.

[addr1,[addr2]]write file (**w**)

Writes the contents of the specified line or lines to **file**. The default range is all lines in the buffer. The **edit** command displays the number of lines and characters that it writes. If you do not specify a **file**, the **edit** command uses the current file name. If **file** does not exist, the **edit** command creates it.

[addr1,[addr2]]yank [buffer] (**ya**)

Places the specified line or lines in **buffer**, which is a single alpha character **a** through **z**.

[addr]z	Displays a screen of text, beginning with the specified line.
[addr]z-	Displays a screen of text, with the specified line at the bottom of the screen.
[addr]z.	Displays a screen of text, with the specified line in the middle of the screen.

Related Information

See the following commands: "ed, red" in topic 1.1.147, "ex" in topic 1.1.158, and "vi, vedit, view" in topic 1.1.522.

1.1.150 edquota

Purpose
Edits user quotas.

Syntax

```
           +-----------------+
edquota ---¦                 +--- users ---¦
           +- -p proto-user -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **edquota** command is a quota editor. One or more users may be specified on the command line. For each user a temporary file is created with an ASCII representation of the current disc quotas for that user and an editor is then invoked on the file. The quotas may then be modified, new quotas added, etc. Upon leaving the editor, **edquota** reads the temporary file and modifies the binary quota files to reflect the changes made.

The editor invoked is **vi** unless the environment variable **EDITOR** specifies otherwise.

Only the superuser may edit quotas.

Flags

-p	Causes **edquota** to duplicate the quotas of the prototypical user specified for each user specified. This is the normal mechanism used to initialize quotas for groups of users.

Files

quotas	At the root of each filesytem with quotas.
/etc/filesystems	To find file system names and locations.

Related Information

See the following commands: "quota" in topic 1.1.350 and "quotaon, quotaoff" in topic 1.1.352.

1.1.151 env, printenv

Purpose
Displays current environment or sets the environment for the execution of a command.

Syntax

```
       +-----------------------------------+
env ---¦ +-----+                           +---¦
       +-¦     +--- name=value --- command -+
         +- - -+                ¦
               +---------------+

            +--------+
printenv ---¦        +---¦
            +- name -+
```

Description
The **env** command lets you get and change your current environment, and then run the named **command** with the changed environment. Changes in the form **name=value** are added to the current environment before the command is run. If **-** (minus) is used, the current environment is ignored and the command runs with only the changed environment. Changes are only in effect while the named **command** is running.

If a **command** is not specified, **env** displays your current environment one **name=value** pair per line.

Printenv displays the values of the variables in the environment. If a **name** is specified, only its value is printed. If a **name** is not specified, **printenv** displays the current environment, one **name = value** per line.

If a **name** is specified and it is not defined in the environment, **printenv** returns exit status 1, else it returns status 0.

Examples

1. To add a shell variable to the environment for the duration of one command:

   ```
   TZ=MST7MDT date
   env TZ=MST7MDT date
   ```

   Each of these commands displays the current date and time in Mountain Standard Time. The two commands shown are equivalent. When **date** is finished, the previous value of **TZ** takes effect again.

2. To replace the environment with another one:

   ```
   env - PATH=$PATH IDIR=&Du./jim/include LIBDIR=&Du./jim/lib make
   ```

   This runs **make** in an environment that consists **only** of these definitions for **PATH**, **IDIR**, and **LIBDIR**. You must redefine **PATH** so that the shell can find the **make** command.

   When **make** is finished, the previous environment takes effect again.

Related Information

See the following commands: "sh, Rsh" in topic 1.1.420 and "csh" in topic 1.1.100.

See **exec** system call, the **profile** file, and the **environ** miscellaneous facility in *AIX Operating System Technical Reference.*

Compatibility Note

The **env** and the **printenv** commands without parameters provide the same function.

1.1.152 eqn, neqn, checkeq

Purpose
Formats mathematical text for the **nroff** and **troff** commands.

Syntax

```
 one of
+------+   +-----------------+   +--------+
¦ eqn  +---¦ +--------------+ +---¦        +---¦
¦ neqn ¦   +-¦ -dxy   -pnum  +-+   +- file -+
+------+     ¦ -ssize -ffont ¦¦          ¦
            ¦+--------------+¦     +------+
            +-----------------+


           +--------+
checkeq ---¦        +---¦
           +- file -+
                  ¦
            +------+
```

Description
The **eqn** command is a **troff** command preprocessor for typesetting mathematical text on a photo typesetter. The **neqn** command is used with **nroff** for other **printing devices**. The output of the **eqn** and **neqn** command is generally piped into the **troff** and **nroff** command as follows:

```
  eqn file | troff
  neqn file | nroff
```

If you do not specify any files or if you specify **-** as the last file name, the commands read standard input. A line consisting of **.EQ** marks the start of equation text; a line consisting of **.EN** marks the end of equation text. Neither of these lines is altered by the commands, so they can be defined in macro packages to give you centering and numbering. The program **checkeq** reports missing or unbalanced delimiter pairs and **.EQ/.EN** pairs. For information on how to format **eqn** text, see in the *AIX Text Formatting Guide*.

The **eqn** command recognizes the following mathematical words and prints the associated symbol:

above	dotdot	italic	rcol	to
back	down	lcol	right	under
bar	dyad	left	roman	up
bold	fat	lineup	rpile	vec
ccol	font	lpile	rpile	~
col	from	mark	size	^
cpile	fwd	matrix	sub	{}
define	gfont	ndefine	sup	"..."
delim	gsize	over	tdefine	
dot	hat	pile	tilde	

Flags

-dxy Sets **x** and **y** as one character delimiters of the text to be processed by the **eqn** command, in addition to the **.EQ** and **.EN** macros. The text between these delimiters will be treated as input to the **eqn** command.

Note: Within a file, you can also set delimiters for **eqn** command text by using the command **delim xy**. The delimiters are turned off by the command **delim off**. All text that is not between delimiters or **.EQ** and **.EN** is passed through unprocessed.

-ffont Acts the same as **-s** for fonts. See the discussion of **gfont** and **font** in *Text Formatting Guide* for information on changing font within the text.

-pnum Reduces subscripts and superscripts **num** points in size (the default is 3).

-ssize Changes point size in all text processed by the **eqn** command to **size**. See the discussion of **gsize** and **size** in the *AIX Text Formatting Guide* for information on changing the point size within the text.

Related Information

See the following commands: "cw, checkcw" in topic 1.1.108, "mm, checkmm" in topic 1.1.274, "mmt, mant, mvt" in topic 1.1.275, and "nroff, troff" in topic 1.1.301.

See the **eqnchar** and **mv** miscellaneous facilities in *AIX Operating System Technical Reference*.

See the discussion of **eqn** in the *AIX Text Formatting Guide*.

1.1.153 errdead

Purpose
Extracts error records from dump.

Syntax

```
           +--- dumpfile ---+
errdead ---¦                +---¦
           +- kernel-image -+
```

Description
The **errdead** command provides a means of extracting error records from the internal buffer maintained by the **/dev/error** special file when the error logging daemon, **errdemon**, is not active or from a system dump. The **errdead** command extracts the error records from memory or from the dump file and passes them to **errdemon** to be added to the system error log.

The *dumpfile* parameter specifies the file (or memory) to be examined. The *kernel-image* parameter specifies the system name list, by default **/unix**.

Only a user operating with the proper authority can run **errdead**.

Files

/unix System kernel image.

Related Information
The following commands: "errdemon" in topic 1.1.154, "errpt, errpd" in topic 1.1.155, "errstop" in topic 1.1.156, and "errupdate" in topic 1.1.157.

1.1.154 errdemon

Purpose
Starts the error-logging demon.

Syntax

```
                        ¦
/usr/lib/errdemon ---¦


----------------
¦ This command is not usually
  run from the command line.
```

Description
The error-logging demon **errdemon** collects error records from the operating system by reading the special file **/dev/error** and places them in one of two error log files. **errdemon** creates the names of the two log files by adding a **.0** and **.1** to the end of the file name found in **/etc/rasconf**. If an error log file does not already exist, **errdemon** creates one.

The **errdemon** command adds error records to the first error log file until it reaches the maximum allowable length specified in **/etc/rasconf**. At that point, **errdemon** closes the first error log file, changes the file name from **file name.0** to **file name.1**, and opens a new **file name.0**. Thus, the newest error records are always in **file name.0**. When it is full, **errdemon** overwrites the first file.

You can stop the error-logging demon by sending it a **SIGKILL** signal (see "errstop" in topic 1.1.156). Normally, the **/etc/rc** command file runs **errdemon** at system startup. Only a user operating with superuser authority can start **errdemon**, and only one demon may be active at any time.

Files

/dev/error	Source of error records.
/etc/rasconf	Configuration file.
/etc/rc	System startup file.
/usr/adm/ras/errfile*	Repository for error records.

Related Information
See the following commands: "errpt, errpd" in topic 1.1.155, "errstop" in topic 1.1.156, and "kill" in topic 1.1.221.

See the **error** and **nvram** files in *AIX Operating System Technical Reference*.

1.1.155 errpt, errpd

Purpose
Processes a report of logged errors.

Syntax

```
         +-------------+   +------------+
errpt ---¦ +---------+ +---¦            +---¦
         +-¦ -s date +-+   +--- file ---+
           ¦ -e date ¦              ¦
          ¦¦ -a      ¦¦      +--------+
          ¦¦ -d list ¦¦
          ¦+---------+¦
          +-----------+

/usr/lib/errpd ---¦
```

Description
The **errpt** command reads a specified error **file** or **file**s, processes the data, and writes a report of that data to standard output. These error files are named **file.0** or **file.1**, but do not include the **.0** or **.1** extension when you specify the file name argument. The **errpt** command adds the extension. If you do not specify a file name, the **errpt** command uses the file listed in the file **/etc/rasconf**, adding the **.0** and **.1** extensions (these are usually **/usr/adm/ras/errfile.0** and **/usr/adm/ras/errfile.1**). The default report is a summary of all errors posted in the named file, as well as system information events, such as time changes, system starts, and so on.

The **errpt** command pipes error entries through the program **/usr/lib/errpd**, which adds probable cause information to certain entries. If no probable cause information is added, the **errpt** command logs records exactly as it receives them.

Flags

-a	Produces a detailed report. This report contains specific error information for every event that the **errpt** command formats.
-d list	Limits the report to certain types of error records as defined by **list**. The list items can either be separated by commas or enclosed in double quotation marks and separated by commas or blanks. See "Error Identifiers" for the valid list values.
-e date	Includes all records posted earlier than **date**, where **date** has the form **MMddhhmmyy** (month, day, hour, minute and year).
-s date	Includes all records posted later than **date**, where **date** has the form **MMddhhmmyy**.

Error Identifiers
In the following error identifiers, **0** acts as a wildcard character, such that, for example, **H00** gives you all hardware errors (**H11** to **HFF**), and **H10** gives you all errors from **H11** to **H1F**, and so on.

The following identifiers can be used in conjunction with the **-d** option to report only those errors for a given class. For example:

```
  errpt -d s00
```

generates a report containing only software errors.

1. Class

```
        H00 = Hardware (01)
        S00 = Software (02)
        I00 = IPL/Shutdown (03)
        G00 = General System Condition (04)
        U00 = User Defined, Non-Hardware
```

2. Class/Subclass

```
        H20 = Hardware/Fixed Disk Drive and Adapter
        H30 = Hardware/Diskette Drive and Adapter
        H40 = Hardware/Tape and Adapter
        H50 = Hardware/Display Station
        H70 = Hardware/Keyboard/Mouse
        H80 = Hardware/Communication Adapters
        H81 = Hardware/RS232 Multi-port
        H84 = Hardware/Serial or Serial/Parallel
        H85 = Hardware/IBM PC Network Adapter
        H86 = Hardware/RS422 Multi-port
        H90 = Hardware/Parallel Printer and Adapter
        H91 = Hardware/Parallel or Serial/Parallel
        HA0 = Hardware/Printers
        HF0
        .
        .
        .
        HFF = User Defined Hardware

        S30 = Software/Program Error AIX
        S33 = Software/Program Error AIX Kernel
        S40 = Software/Program Error AIX Device Driver
        S80 = Software/Program Error Application
        S80 = Software/Program Error Application - Error Log Analysis
        S80 = Software/Program Error Application - Interactive Workstation
        S90 = Software/Program Error Application
        SA0 = Software/Program Error Application
        SB0 = Software/Program Error Application
        SC0 = Software/Program Error Application
        SD0 = Software/Program Error Application
        SE0 = Software/Program Error Application
        SF0 = Software/Program Error Application

        I10 = IPL/Shutdown/Manual IPL
        I20 = IPL/Shutdown/Soft IPL
        I30 = IPL/Shutdown/Auto IPL
        I40 = IPL/Shutdown/Shutdown
        I50 = IPL/Shutdown/Maintenance Shutdown
        G10 = General System Condition/Degraded Config
        G20 = General System Condition/Set Date/ Time
        G40 = General System Condition/Error Reporting
        G50 = General System Condition/POST
        G41 = General System Condition/Cause Codes
```

```
          G42 = General System Condition/Device Information
          G43 = General System Condition/Counters
          U10
          .
          .
          .
          UFF = User Defined, Non-Hardware
```

Subtopics

1.1.155.1 errpd

The error log analysis program, **/usr/lib/errpd**, analyzes the error log data. This program processes error data to determine if the error is a hardware error and if the error is a temporary or permanent error.

Files

/usr/adm/ras/errfile?	Error file.

Related Information

See the following command: "errdemon" in topic 1.1.154.

See the **errfile** file in *AIX Operating System Technical Reference*.

1.1.156 errstop

Purpose
Terminates the error-logging daemon.

Syntax

```
           +---- /unix -----+
errstop ---¦                +---¦
           +- kernel-image -+
```

Description
The **errstop** command stops the error-logging daemon **errdemon** by running the **ps** command to determine the daemon process ID and then sending it a Software Terminate signal (see the **sigaction** system call in *AIX Operating System Technical Reference*). If you do not specify **kernel-image**, **errstop** uses **/unix**. Only a user operating with superuser authority can run **errstop**.

Files

/unix	System kernel image.

Related Information

See the following commands: "errdemon" in topic 1.1.154 and "ps" in topic 1.1.337.

See the **kill** system call in *AIX Operating System Technical Reference*.

1.1.157 errupdate

Purpose

Updates an error report template.

Syntax

```
              +-- file --+
errupdate ---¦           +---¦
              +- file-o -+
```

Description

The **errupdate** command adds, replaces, or deletes error report format templates in the file **/etc/errfmt**. The **errupdate** command creates an undo file in the current directory that it names **file.undo.err**. You can use this undo file as input to **errupdate** with the **-o** (override) flag to undo the changes **errupdate** has just made.

The **errupdate** command adds the extension **.err** to the **file** name you specify and reads update commands from the file with that name and extension. The first field of each template contains an operator:

+ To add or replace a template

- To delete a template.

If the operation is +, then the following fields contain the template to be replaced. If the operation is a -, the second field contains the class/subclass/mask identifier of the template to delete. The **errupdate** command checks for valid combinations of identifiers and writes error messages if it encounters invalid combinations. When adding or replacing, it compares the version numbers of each input template with the version number of the existing template of the same class/subclass/mask and, if the version number of the input template is later, replaces the old template with the input template. If the template does not already exist, it is added to the file. The input template **must** contain an identifier line on the first line:

```
* /etc/errfmt
```

or the **errupdate** command rejects the input file. All delete operations are performed before the add/replace operations.

Flag

-o Does no version number checking.

Example

The following is an example input file:

```
* /etc/errfmt
+ H87 2.0 Native Serial: IODN D2: IOCN D2: Base_Addr D4:\
        Dev_Name A4: \n: Dev_Type X4: DDI_Length D4: Error_Type X1:\
        Last_I/O X1: Line_Status X1: Printer_Status X1:
- H92
```

Files

/etc/errfmt	Contains error report format files.
file.err	**errupdate** file.
file.undo.err	**errupdate** undo file.

Related Information

See the following commands: "errpt, errpd" in topic 1.1.155.

1.1.158 ex

Purpose
Edits lines interactively with screen display.

Syntax

```
     +------------+   +--------+   +-----------------+
ex ---¦ +--------+ +---¦ one of +---¦     +----------+ +---
     +-¦ -l     +-+   ¦ +----+ ¦   +- + -¦¦          +-+
       ¦ -R     ¦     +-¦ -v +-+         +- subcmd -+
       ¦ -t tag ¦       ¦ -  ¦
       ¦ -w num ¦       +----+
       +--------+

   +----------------+   +------------+
 ---¦     +--------+ +---¦           +---¦
   +- -r -¦        +-+   +--- file ---+
         +- file -+               ¦
                      +--------+
```

```
-----------------
¦ Do not put a blank between these items.
```

Description
The **ex** command is a line-oriented text editor that is a subset of the **vi** screen editor. The **ex** editor is similar to **ed**, but is more powerful, providing multi-line displays and access to a screen editing mode. You may prefer to call **vi** directly to have environmental variables set for screen editing. Also **edit**, a limited subset of **ex**, is available for novice or casual use. For more information on **vi**, see "vi, vedit, view" in topic 1.1.522. For more information on **edit**, see "edit" in topic 1.1.149.

Notes:

1. Some **vi** subcommands have meanings that differ from **ed** subcommands.

2. To determine how to drive your work station more efficiently, **ex** uses the work station capability data base **terminfo** and the type of the work station you are using from the shell environment variable **TERM**.

The **ex** editor has the following features:

You can view text in files. The **z** subcommand lets you access windows of text, and you can scroll through text by pressing **Ctrl-D**. The **vi** subcommand provides further viewing options and active screen-editing by invoking the **vi** editor.

You can you revoke the last previous subcommand entered (except for **q** and **w**). The **undo** subcommand allows you to "undo" the last subcommand, even if it's an **undo** subcommand. Thus you can switch back and forth between the latest change in the edit file and the last prior file status and view the effect of a subcommand without that effect being permanent. The **ex** command displays changed lines and indicates when more than a few lines are affected by a subcommand. The **undo** subcommand causes all marks to be lost on lines changed and then restored if the marked lines were changed. It does not clear the

buffer modified condition.

You can retrieve your work (except changes that were in the buffer) i the system or the editor crashes by re-entering the editor with the **-r** flag and the file name. When the file name is not specified, all open files in your partition are listed.

You can queue a sequence or group of files to edit. You can list th files in the **ex** command and then use the **next** subcommand to access each file sequentially. Or after you enter the editor, you can enter the **next** subcommand with a list of file names or a pattern (as used by the shell) to specify a set of files. In general, you can designate file names to the editor using the pattern-matching symbols that the shell accepts. You can use the wild card character **%** to form file names and represent the name of the current edit file.

You can use a group of buffers (buffers named **a** through **z**) to move text between files and within a file. You can temporarily place text in these buffers and copy or reinsert it in a file, or you can carry it over to another file. The buffers are cleared when you quit the editor. The editor does not notify you if text is placed in a buffer and not used before exiting the editor. The buffer names can contain only ASCII digits.

You can use patterns that match words. For example, you can searc only for the word "ink" when your document also contains the word "inkblot" or "blink". The patterns can contain Japanese characters.

You can display a window of logical lines. The **z** subcommand allows you to select the number of lines displayed and locate the current line within the display simultaneously. More than a screen of output can result when the file lines are longer than the output display lines because the set number of logical lines are displayed rather than a number of physical lines.

You can read a file of editor subcommands. The **so** command allows you to read and execute a file of subcommands. Nesting of source files is permitted, allowing one file to call another; however, no return mechanism is provided.

The **ex** editor has the following maximum limits. If have selected a language (through the **LANG** environment variable) that supports multibyte characters, the character limits can be reduced by as much as 50%, depending on the character code set being used.

1024 characters per lin
256 characters per global command lis
128 characters in the previous inserted and deleted tex
100 characters in a shell escape comman
63 characters in a string-valued optio
30 characters in a tag nam
250,000 lines of 1024 characters per line silently enforce
128 map macros with 2048 characters tota
100 characters per each **map** macro subcommand (or rhs).

Subtopics

1.1.158.1 Editing States

command	Normal and initial state. Input is prompted for by : (colon). Pressing END OF FILE (**Ctrl-D**) clears an uncompleted subcommand from the command line.
visual	Entered by **vi**, **vi.**, **vi-**, or **o**. Each of the first three commands gives you a full screen **vi** editor, but puts the current line in a different place on entry. Enter **vi** to put the current line at the top of the screen; enter **vi.** to put the current line in the middle of the screen; and enter **vi-** to put the current line at the bottom of the screen. The **o** command opens a one-line window. All three commands share the input state of the visual editor. Press the **Esc** key to exit the input state. To return to the **ex** command state at the current line, enter **Q** or **^\** while not in the input state.
entry	Entered by **a, i** and **c**. In this state you can enter text. Entry state ends normally with a line that has only a . (period) on it or ends abnormally if you press INTERRUPT (**Ctrl-C**). **Note:** This editing state applies only to the RT PC.

Subcommands

The following is a list of the **ex** subcommands. Most of these subcommands are discussed under "edit" in topic 1.1.149 or "vi, vedit, view" in topic 1.1.522.

ab	abbrev	**n**	next	**una**	unabbrev
a	append	**nu**	number	**u**	undo
ar	args			**unm**	unmap
c	change	**pre**	preserve		
co	copy	**p**	print	**vi**	visual
d	delete	**pu**	put	**w**	write
e	edit	**q**	quit	**x**	exit
f	file	**re**	read	**ya**	yank
g	global	**rec**	recover	**z**	window
i	insert	**rew**	rewind	**!**	escape
j	join	**se**	set	**<**	lshift
l	list	**sh**	shell	**CR**	print next
map	map	**so**	source	**&**	resubst
ma	mark	**>**	rshift		
m	move	**s**	substitute	**^D**	scroll

1.1.158.2 Subcommand Addresses

$	The last line	**x-num**	The **num**th line before **x**
+	The next line	**x,y**	Lines **x** through **y**
-	The previous line	**'m**	The line marked with **m**
+ **num**	The **num**th line forward	''	The previous context
-**num**	The **num**th previous line	/**pat$**	The next line with **pat** at end of line
%	The first through last lines	/^**pat**	The next line with **pat** at start of line
num	line **num**	**/pat**	The next line with **pat**
.	The current line	**?pat**	The previous line with **pat**

1.1.158.3 Scanning Pattern Formation

^	The beginning of the line
$	The end of the line
.	Any character
\<	The beginning of the word
\>	The end of the word
[string]	Any character in **string**
[^string]	Any character not in **string**
[x-y]	Any character between **x** and **y**, inclusive
*****	Any number of the preceding character.

Flags

-l	Indents appropriately for Lisp code, and modifies the functions of the () {} [and] characters when used as **vi** subcommands. The **Lisp** modifier is active in **open** or **visual** modes.
-r [file]	Recovers **file** after an editor or system crash. If you do not specify **file**, a list of all saved files is displayed.
-R	The **readonly** option is set, preventing you from altering the file if you are an ordinary user.
-t tag	Loads the file that contains **tag** and positions the editor at **tag**. The **tag** can contain only ASCII characters.
-v	Invokes the **visual** editor. **Note:** When the **v** flag is selected, an enlarged set of subcommands are available, including screen editing and cursor movement features. See "vi, vedit, view" in topic 1.1.522.
-	Suppresses all interactive-user feedback. If you use this flag, file input/output errors do not generate a helpful error message.
+subcmd	Begins the edit at the specified editor search or subcommand. When **subcom** is not entered, +places the current line to the bottom of the file. Normally **ex** sets current line to the start of the file, or to some specified tag or pattern.
-w num	Defines the number of lines desired in the window. The **num** variable can contain only ASCII characters.

Files

/usr/lib/exrecover	Recover subcommand.
/usr/lib/expreserve	Preserve subcommand.
/usr/lib/*/*	Describes capabilities of work stations.
$HOME/.exrc	Editor startup file.
./.exrc	Editor startup file.
/tmp/Exnnnnn	Editor temporary.
/tmp/Rxnnnnn	Names buffer temporary.
/usr/preserve	Preservation directory.

Related Information

See the following commands: "vi, vedit, view" in topic 1.1.522, "edit" in topic 1.1.149, "ctags" "awk, nawk, oawk" in topic 1.1.29, "ed, red" in topic 1.1.147, "grep, egrep, fgrep" in topic 1.1.193, and "sed" in topic 1.1.415.

See **curses** subroutine and the **TERM** and **terminfo** files in *AIX Operating System Technical Reference*.

1.1.159 expr

Purpose
Evaluates arguments as an expression.

Syntax

expr -- **expression** --¦

Description
The **expr** command reads an **expression**, evaluates it, and writes the result to standard output. Within **expression**, you must separate each term with blanks, precede characters special to the shell with a \(backslash), and quote strings containing blanks or other special characters. Note that **expr** returns 0 rather than the null string, to indicate a zero value. Integers may be preceded by a unary minus sign. Internally, integers are treated as 32-bit, two's-complement numbers.

The operators and keywords are described in the following listing. Characters that need to be escaped are preceded by a \ (backslash). The list is in order of increasing precedence, with equal precedence operators grouped within {} (braces).

\c \)
Used to override the normal precedence of the operators.

expression1 **\|** *expression2*
Returns **expression1** if it is neither null nor 0; otherwise, it returns **expression2**.

expression1 **\&** *expression2*
Returns **expression1** if neither **expression1** nor **expression2** is null or 0; otherwise, it returns 0.

expression1 **{ =, \>, \>=, \<, \<=, != }** *expression2*
Returns the result of an integer comparison if both expressions are integers; otherwise, returns the result of a string comparison.

expression1 **{+, - }** *expression2*
Adds or subtracts integer-valued arguments.

expression1 **{ *, /, % }** *expression2*
Multiplies, divides, or provides the remainder from the division of integer-valued arguments.

expression1 : expression2
Compares **expression1** with **expression2**, which must be a pattern; pattern syntax is the same as that of the **ed** command (see page 1.1.147), except that all patterns are anchored, so **^** (which anchors a pattern to the beginning of a line), is not a special character in this context.

Normally, the matching operator returns the number of characters matched. Alternatively, you can use the **\(...\)** symbols in **expression2** to return a portion of **expression1**. In an expression such as **[a-z]**, the minus means "through" according to the current collating sequence. A collating sequence may define equivalence classes for use in character ranges. See the

"Overview of International Character Support" in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

The **expr** command returns the following exit values:

0 The expression is neither null nor 0.

1 The expression is null or 0.

2 The expression is invalid.

Note: After parameter processing by the shell, the **expr** command cannot distinguish between an operator and an operand except by the value. Thus, if **$a** is **=**, the command:

```
expr $a = '='
```

looks like:

```
expr = = =
```

after the shell passes the arguments to the **expr** command, and they all are taken as the **=** operator. The following works:

```
expr X$a = X=
```

Examples

1. To modify a shell variable:

   ```
   COUNT=`expr $COUNT + 1`
   ```

   This command adds **1** to the shell variable **COUNT**. The **expr** command is enclosed in grave accents, which causes the shell to substitute the standard output from **expr** into the **COUNT=** command. For more details, see "Command Substitution" in topic 1.1.420.5.

2. To find the length of a shell variable:

   ```
   LENGTH=`expr $STR : ".*"`
   ```

   This command sets **LENGTH** to the value given by the **:** (colon) operator. The pattern **.*** matches any string from beginning to end, so the colon operator gives the length of **STR** as the number of characters matched. Note that **.*** must be in quotes to prevent the shell from treating the ***** as a pattern-matching character. The quotes themselves are not part of the pattern.

   If **STR** is set to the null string, the error message **expr: syntax error** is displayed because the shell does not normally pass null strings to commands. In other words, the **expr** command sees only

   ```
   : .*
   ```

   (The shell also removes the quotation marks.) This does not work because the colon operator requires two values. This problem can be fixed by enclosing the shell variable in double quotation marks:

   ```
   LENGTH=`expr "$STR" : ".*"`
   ```

Now if **STR** is null, **LENGTH** is set to zero. Enclosing shell variables in double quotes is recommended in general. However, do **not** enclose shell variables in single quotes. See page 1.1.420.8 for details about quoting.

3. To use part of a string:

   ```
   FLAG=`expr "$FLAG" : "-*\(.*\)"`
   ```

   This command removes leading minus signs, if any, from the shell variable **FLAG**. The colon operator gives the part of **FLAG** matched by the part of the pattern enclosed in \(\). If you omit the \(\), the colon operator gives the number of characters matched.

   If **FLAG** is set to - (minus), a syntax error message is displayed because the shell substitutes the value of **FLAG** before running the **expr** command. The **expr** command does not know that the minus is the value of a variable. It can only see:

   ```
   - : -*\(.*\)
   ```

   and it interprets the first minus sign as the subtraction operator. We can fix this problem by using:

   ```
   FLAG=`expr "x$FLAG" : "x-*\(.*\)"`
   ```

4. To use the **expr** command in an **if** statement:

   ```
   if expr "$ANSWER" : "[yY]" >/dev/null
   then
      # ANSWER begins with "y" or "Y"
   fi
   ```

   If **ANSWER** begins with **y** or **Y**, the **then** part of the **if** statement is performed. If the match succeeds, the result of the expression is **1** and **expr** returns an exit value of **0**, which is recognized as the logical value TRUE by the **if** statement. If the match fails, the result is **0** and the exit value is **1** (FALSE).

   Redirecting the standard output of the **expr** command to the **/dev/null** special file discards the result of the expression. If you do not redirect the result, it is written to the standard output, which is usually your work station display.

5. Consider the following expression:

   ```
   expr "$STR" = "="
   ```

   If **STR** has the value **=** (equal sign), after the shell processes this command **expr** sees the expression:

   ```
   = = =
   ```

   The **expr** command interprets this as three **=** operators in a row and displays a syntax error message. This error happens whenever the value of a shell variable is the same as one of the **expr** operators. You can avoid this problem by doing the following:

   ```
   expr "x$STR" = "x="
   ```

Related Information

See the following commands: "ed, red" in topic 1.1.147 and "sh, Rsh" in topic 1.1.420.

See the "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.160 factor

Purpose
Factors a number.

Syntax

```
          +----------+
factor ---¦          +---¦
          +- number -+
```

Description
When called without an argument, the **factor** command waits for you to enter a positive number less than 10¦4. It then writes the prime factors of that number to standard output. It displays each factor the proper number of times. To exit, enter a 0 or any non-numeric character.

When called with an argument, the **factor** command determines the prime factors of **number**, writes the results to standard output, and exits.

Example

To calculate the prime factors of **123**:

```
  factor  123
```

This displays:

```
  123
       3
       41
```

1.1.161 fast, fastsite

Purpose

Finds the least loaded site in the network.

Syntax

```
fast ------------------- cmd -----------+
        ¦ -v   - -  ¦ ¦        ¦ arg ¦ ¦
      ¦ ¦ -a        ¦ ¦      ¦ +-----+ ¦
      ¦ ¦ -pcputype ¦ ¦      +---------+
      ¦ +-----------+ ¦
      +---------------+

fastsite -----------------
          ¦ -v  -n    ¦
          ¦ -a        ¦
          ¦ -pcputype ¦
          +-----------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

Note: The **fast** and **fastsite** commands consider only those sites for which the user has executable permissions.

The **fastsite** command selects and then writes to standard output the site number of the least loaded site in the network. The load average (see "loads" in topic 1.1.236) and the CPU performance factor (see **site** in the *AIX Operating System Technical Reference*) are used to select the least loaded site. This can be used to make programs which are CPU-intensive run on a machine with the lightest load. **fast** is similar to **fastsite**, except that it also runs the indicated command on the selected site.

Without the **-a** option, site selection is restricted to sites with the same CPU-type as the site on which **fast** or **fastsite** is run.

If the **-v** option is specified, the name of the site that is chosen is written to the standard error file descriptor.

If the **-p** option is specified, it must be followed immediately by the machine type required (for example, -pi370). This will select the least loaded machine of the specified type rather than the least loaded machine of the same type as the machine on which the command was run. The **-a** cannot be used with this option.

If the **-n** option is specified, the name of the fastest site is written to the standard output instead of the number.

Use the -- option to terminate the list of flag options to **fast** when the command name begins with a -.

Using **fast**, or **fastsite** together with **onsite**, will not make I/O-intensive programs which need to access files or devices on a particular site run faster, and may even make them slower by making them access their resources remotely.

The possible **cputype** values are:

i386	An AIX PS/2 system.
i370	An AIX/370 system.
xa370	An XA AIX/370 system.
s370	A non-XA AIX/370 system.

Examples

```
  fast cc foo.c -o foo
```

csh users can establish a **fast** alias which uses the built-in **onsite** command and this allows other aliases to be specified as the command to run:

```
  alias fast "onsite `fastsite` "!*"
```

Files

/etc/local_loads	Loads information for local site.
/etc/loads	Loads information for all sites.
/etc/loadserver	Daemon which updates **/etc/local_loads** and **/etc/loads**.
/etc/sitegroups	Execute permissions for all sites

Diagnostics
fast and **fastsite** silently ignore any site which is down.

Related Information

See the following commands: "anet" in topic 1.1.18 and "onsite, on" in topic 1.1.306.

1.1.162 fastboot, fasthalt

Purpose
Reboots/halts the system without checking the disks.

Syntax

```
                +------------------+
/etc/fastboot ---¦                  +---¦
                +- reboot-options -+

                +----------------+
/etc/fasthalt ---¦                +---¦
                +- halt-options -+
```

Description

The **fastboot** and **fasthalt** commands are shell scripts which reboot and halt the system without checking the file systems. This is done by creating a file **<LOCAL>/fastboot**, then invoking the **reboot** program. The system startup script, **/etc/rc**, looks for this file and, if present, skips the normal invocation of **fsck**.

Related Information
See the following commands: "halt" in topic 1.1.195, "rc" in topic 1.1.354 and "reboot" in topic 1.1.363.

1.1.163 ff

Purpose
Lists the file names and statistics for a file system.

Syntax

```
      +---------------------+   +------------+
ff ---¦ +-----------------+ +---¦            +-- device --¦
      +-¦ -a num -n file   +-+   +- -i inode -+
        ¦ -c num -p prefix ¦¦              ¦
       ¦¦ -l     -s        ¦¦       +--,---+
       ¦¦ -m num -u        ¦¦
       ¦¦ -I               ¦¦
       ¦+-----------------+¦
       +-------------------+
```

Description
The **ff** command reads the i-list and directories specified by **device** and writes information about them to standard output. It assumes that **device** is a file system, and saves inode data for files specified by flags. The output from the **ff** command consists of the path name for each saved inode, in addition to other file information that you request with the flags. The output is listed in order by inode number, with tabs between all fields. The default line produced by the **ff** command includes the path name and i-number fields. With all flags enabled, the output fields include path name, i-number, size, and UID.

The **num** parameter in the flags descriptions is a decimal number, where **+num** means more than **num**, **-num** means less than **num**, and **num** means exactly **num**. A day is defined as a 24-hour period.

The **ff** command lists only a single path name out of many possible ones for an inode with more than one link, unless you specify the **-l** flag. With this flag, the **ff** command applies no selection criteria to the names listed. All possible names for every linked file on the file system are included in the output. On very large file systems, memory may run out before the **ff** command finishes.

If device names a copy of a replicated file system, only files actually stored in this copy of the file system are listed. In particular, if a directory entry for a file is present but the file is not present, the file is ignored by **ff**.

Flags

-a num	Gets selected if the inode has been accessed in **num** days.
-c num	Gets selected if the inode has been changed in **num** days.
-i inode	Generates names for only those inodes specified in the **inode** list. The maximum number of inodes for the PS/2 or 370 is 128. For other machines, it is 64.
-I	Does not display the inode number after each path name.
-l	Generates a list of all path names for files, including those with more than one link.

-m num	Selects if the file associated with the inode has been modified in **num** days.
-n file	Selects if the file associated with the inode has been modified more recently than the specified **file**.
-p prefix	Adds the specified **prefix** to each path name. The default prefix is **.** (dot).
-s	Writes the file size, in bytes, after each path name.
-u	Writes the owner's login name after each path name.

Examples

1. To list the path names of all files in a given file system:

   ```
   ff  -I  /dev/hd1
   ```

   This command displays the path names of the files on the **/dev/hd1** disk. If you do not specify the **-I** flag, the **ff** command also displays the i-number of each file.

2. To list files that have been modified recently:

   ```
   ff  -m  -2  -u  /dev/hd1
   ```

   This command displays the path name, i-number, and owner's user name (**-u**) of each file on the **/dev/hd1** disk that has been modified within the last two days (**-m -2**).

3. To list files that have **not** been used recently:

   ```
   ff  -a  +30  /dev/hd1
   ```

   This command displays the path name and i-number of each file that was last accessed more than 30 days ago (**-a +30**).

4. To find out the path names of certain inodes:

   ```
   ff  -l  -i  451,76  /dev/hd1
   ```

   This command displays all the path names (**-l**) associated with inodes **451** and **76**.

Related Information
See the following commands: "find" in topic 1.1.165 and "ncheck" in topic 1.1.286.

1.1.164 file

Purpose
Determines file type.

Syntax

```
        +- -m /etc/magic -+   +- -f file --+
file ---¦                 +---¦            +---¦
        +--- -m mfile ----+   +--- file ---+
                                         ¦
                                +--------+


             +- -m /etc/magic -+
file -- -c --¦                 +---¦
             +--- -m mfile ----+
```

Description
The **file** command reads its input **file**s, performs a series of tests on each one, and attempts to classify them by their types. The **file** command then writes the file types to standard output. If a file appears to be ASCII, **file** examines the first 512 bytes and tries to determine its language. If a file does not appear to be ASCII, **file** further attempts to distinguish a binary data file from a text file that contains extended characters. If **file** is an **a.out** file, the **file** command displays the version stamp.

The **file** command uses the file **/etc/magic** to identify files that have some sort of ***magic number***; that is, any file containing a numeric or string constant that indicates its type. Comments at the beginning of **/etc/magic** explain its format.

If the file given is a hidden directory, the **file** command selects the component that corresponds to the machine type on which the **file** command is being executed. If the file is an **a.out** file, **file** displays the machine type for which the program was compiles.

Flags

-c	by default (**/etc/magic**) for format errors. This validation is not normally done. File typing is not done under this flag.
-f file	Reads **file** for a list of files to examine.
-m mfile	Specifies **mfile** as the magic file (**/etc/magic** by default).

Examples

1. To display the type of information a file contains:

```
file  myfile
```

   This command displays the file type of **myfile** (directory, data, ASCII text, C-program source, archive, and so forth).

2. To display the type of each file named in a list of file names:

```
file  -f  file_names
```

This command displays the type of each file with a name that appears in **file names**. Each file name must appear alone on a line.

To create **file names**:

```
  ls  -C1 > file_names
```

then edit **file names** as desired.

Files

/etc/magic	File type database.

Related Information

See the "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.165 find

Purpose
Finds files matching expression.

Syntax

```
find --- path --- expression ---¦
                ¦
      +--------+

find --- pattern ---¦
```

Note: The **find** command should be used in conjunction with the **- print** option.

Description
The **find** command recursively searches the directory tree for each specified **path**, seeking files that match a Boolean **expression** written using the terms given below. The output from the **find** command depends on the terms used in **expression**.

When searching a directory tree, symbolic links are treated as terminal nodes even when they point to directories. The exception is when **path** is a symbolic link that points to a directory, in which case that directory tree will be searched.

The second form rapidly searches a data base for all pathnames which match **pattern**. Usually the data base is recomputed weekly by the **updatedb** script which is run from the **/usr/adm/weekly** shell script. The data base contains the pathnames of all files which are publicly accessible. If escaped, normal shell "globbing" characters (* , ?, [,] may be used in **pattern**, but the matching differs in that no characters (for example, /) have to be matched explicitly. As a special case, a simple **pattern** containing no globbing characters is matched as though it were ***pattern***; if any globbing character appears, there are no implicit globbing characters.

Expression Terms
In the following descriptions, the parameter **num** is a decimal integer that can be specified as **+num** (more than **num**), **-num** (less than **num**), or **num** (exactly **num**).

-fstype type	True if the file system to which the file belongs is of the type **type**, where **type** is typically **nfs** or **aix**.
-inum n	True, if file has inode **n**.
-name file	True, if **file** matches the file name. You can use pattern-matching characters, provided they are quoted. In an expression such as **[a-z]**, the minus means "through" according to the current collating sequence. A collating sequence may define equivalence classes for use in character ranges. See "Introduction to International Character Support" in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.
-perm onum	True, if the file permission code of the file exactly

matches the octal number **onum** (see "chmod" in topic 1.1.67 for an explanation of file permissions). The **onum** parameter may be up to three octal digits. If you want to test the higher-order permission bits (the set-user-ID bit or set-group-ID bit, for example), prefix the **onum** parameter with a minus (-) sign. This makes more flag bits significant (see the **stat** system call for an explanation of the additional bits), and also changes the comparison to:

(**flags&onum**)==**onum**

-prune	Always yields true. Has the side effect of pruning the search tree at the file. That is, if the current path name is a directory, find will not descend into that directory.
-type type	True, if the file **type** is of the specified type as follows: **b** Block special file **c** Character special file **d** Directory **f** Plain file **h** Hidden directory **l** Symbolic link **p** FIFO (a named pipe). **s** Socket
-links num	True, if the file has **num** links. See "ln" in topic 1.1.234.
-user uname	True, if the file belongs to the user **uname**. If **uname** is numeric and does not appear as a login name in the **/etc/passwd** file, it is interpreted as a user ID.
-nouser	True, if the file belongs to a user *not* in the **/etc/passwd** data base.
-group gname	True, if the file belongs to the group **gname**. If **gname** is numeric and does not appear in the **/etc/group** file, it is interpreted as a group ID.
-nogroup	True, if the file belongs to a group *not* in the **/etc/group** data base.
-size num	True, if the file is **num** blocks long (512 bytes per block). For this comparison the file size is rounded up to the nearest block.
-atime num	True, if the file has been accessed in **num** days.
-mtime num	True, if the file has been modified in **num** days.
-ctime num	True, if the file inode has been changed in **num** days.
-exec cmd	True, if **cmd** runs and returns a zero value as exit status. The end of **cmd** must be punctuated by a quoted or escaped semicolon. A command parameter **{}** is replaced by the current path name.

-ok cmd	The **find** command asks you whether it should start **cmd**. If your response begins with **y**, **cmd** is started. The end of **cmd** must be punctuated by a quoted or escaped semicolon.
-print	Always true; causes the current path name to be displayed. The **find** command does not display path names unless you specify this expression term.
-hidden	Always true; causes hidden directories encountered in the descent to be treated as directories and their contents searched. Otherwise, hidden directories are treated as ordinary files and are not searched.
-cpio device	Write the current file to **device** in **cpio** format. See "cpio" in topic 1.1.93.
-newer file	True if the current file has been modified more recently than the file indicated by **file**.
-depth	Always true. This causes the descent of the directory hierarchy to be done so that all entries in a directory are affected before the directory itself. This can be useful when **find** is used with **cpio** to transfer files that are contained in directories without write permission.
-ls	Always true; causes current pathname to be printed together with its associated statistics. These include (respectively) inode number, size in kilobytes (1024 bytes), protection mode, number of hard links, user, group, size in bytes, and modification time. If the file is a special file, the size field will instead contain the major and minor device numbers. If the file is a symbolic link, the pathname of the linked-to file is printed preceded by "->". The format is identical to that of "**ls -gilds**N" (note however that formatting is done internally, without executing the **ls**.)
-xdev	Always true; causes find **not** to traverse down into a file system different from the one on which current **argument** pathname resides.
\(expression \)	True, if the expression in parentheses is true.

You may perform the following logical operations on these terms (listed in order of decreasing precedence):

Negate a term (! is the NOT operator)
Concatenate terms (juxtaposing two terms implies the AND operation)
Alternate terms **-o** is the OR operator).

Examples

1. To list all files in the file system with a given base file name:

```
find / -name .profile -print
```

This command searches the entire file system and writes the complete path names of all files named **.profile**. The **/** tells the **find** command to search the root directory and all of its subdirectories. This may take a while, so it is best to limit the search by specifying the directories where you think the files might be.

2. To list the files with a specific permission code in the current directory tree:

```
find  .  -perm  0600  -print
```

   This command lists the names of the files that have **only** owner-read and owner-write permission. The **.** (dot) tells **find** to search the current directory and its subdirectories. See the command: "chmod" in topic 1.1.67 for details about permission codes.

3. To search several directories for files with certain permission codes:

```
find  manual  clients  proposals  -perm  -0600  -print
```

   This command lists the names of the files that have owner-read and owner-write permission and possibly other permissions. The directories **manual**, **clients**, and **proposals**, and their subdirectories, are searched. Note that **-perm 0600** in the previous example selects only files with permission codes that match **0600 exactly**. In this example, **-perm -0600** selects files with permission codes that allow **at least** the accesses indicated by **0600**. This also matches the permission codes **0622** and **2744**.

4. To search for regular files with multiple links:

```
find  .  -type  f  -links  +1  -print
```

   This command lists the names of the ordinary files (**-type f**) that have more than one link (**-links +1**). Every directory has at least two links: the entry in its parent directory and its own **.** (dot) entry. See "ln" in topic 1.1.234 for details about multiple file links.

5. To back up selected files in **cpio** format:

```
find  .  -name  "*.c"  -cpio  /dev/rfd0
```

   This command saves all the **.c** files onto the diskette in **cpio** command format. See "cpio" in topic 1.1.93 for details. The pattern ***.c** must be quoted to prevent the shell from treating the * as a pattern-matching character. This is a special case in which the **find** command itself decodes the pattern-matching characters.

6. To perform an action on all files that meet complex requirements:

```
find . \( -name a.out -o -name "*.o" \) -atime +7 -exec rm {} \;
```

   This command deletes (**-exec rm {} \;**) all files named **a.out** or that end with **.o**, and that were last accessed over seven days ago (**-atime +7**). The **-o** flag is the logical OR operator.

Files

/etc/group	File that contains all known groups.
/etc/passwd	File that contains all known users.

Related Information

See the following commands: "cpio" in topic 1.1.93, "sh, Rsh" in topic 1.1.420, and "test" in topic 1.1.469.

See the **stat** system call and the **cpio** command and **fs** files in *AIX Operating System Technical Reference*.

See the "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.166 fish

Purpose

Plays the card game Go Fish.

Syntax

/usr/games/fish ---¦

Note: This command does not have MBCS support.

Description

The object of the **fish** game is to accumulate ***books*** of four cards with the same face value. You and the program take turns asking each other for a card in your hand. If your opponent has any of that card, he must hand them over. If not, he says "GO FISH", and you draw a card from the pool of undealt cards. If you draw the card you asked for, you draw again. As books are made, they are laid down on the table. Play continues until there are no cards left. The player with the largest number of books wins the game. **fish** tells you the winner and exits.

The **fish** game asks if you want instructions before play begins. To see the instructions, enter "y" or "yes".

Entering **p** as your first move gives you the professional level game.

The **fish** game tells you the cards in your hand each time it prompts for a move. It tells you when either side makes a book, says "GO FISH" for you, and draws for you. All you must enter as play progresses is the value of the card you want to ask for. If you press only the **Enter** key, you are given information about the number of cards in your opponent's hand and in the pool.

To exit the game before play is completed, press INTERRUPT (refer to keyboard definition).

1.1.167 flcopy

Purpose
Copies diskettes.

Syntax

```
          +- -f/dev/rfd0 -+   +---------+   +-------+
flcopy ---¦               +---¦ one of  +---¦       +---¦
          +- -f filename -+   ¦ +-----+ ¦   +- -tn -+
                              +-¦ -h  +-+
                                ¦ -r  ¦
                                +-----+
```

Description
The **flcopy** command copies a diskette (opened as **/dev/rfd0**) to a file created in the current directory, named "floppy", then prints the message "Change floppy, hit return when done." **flcopy** then copies the local file back out to the diskette.

Flags

-f filename	Allows you to specify a file other than **/dev/rfd0**.
-h	Causes **flcopy** to open a file named "floppy" in the current directory and copy it to **/dev/rfd0**.
-r	Tells **flcopy** to exit after copying the diskette into the file named "floppy" in the current directory.
-tn	Causes only the first **n** tracks to participate in a copy.

Files
/dev/rfd0
floppy (in current directory)

Related Information

See the following commands: "format, fdformat" in topic 1.1.172.

See the **fd** file in *AIX Operating System Technical Reference*.

1.1.168 fmt

Purpose
Formats mail messages prior to sending.

Syntax

/usr/bin/fmt -- **file** --¦

Description

The **fmt** command invokes a simple text formatter that reads the concatenation of input **files** (or standard input if no **files** are specified). It then produces, on standard output, a version of the input with lines as close to 72 characters long as possible. The spacing at the beginning of the input lines is preserved in the output, as are blank lines and interword spacing.

The **fmt** command is used generally to format mail messages prior to sending them through the mail facility. It may also be useful, however, for other simple formatting tasks. For example, within visual mode of a text editing program such as **vi**, the command **!}fmt** reformats a paragraph so that all lines are approximately 72 characters long.

Note: The **fmt** command is a fast, simple formatting program. Standard text editing programs are more appropriate than the **fmt** command for complex formatting operations.

Related Information

See the following commands: "mail, Mail" in topic 1.1.253 and "nroff, troff" in topic 1.1.301.

1.1.169 fold

Purpose

fold long lines for finite width output device.

Syntax

```
        +----------+
fold ---¦          +--- file ---¦
        +- -width -+
```

Description

The **fold** command is a filter which will fold the contents of the specified files, or the standard input if no files are specified, breaking the lines to have maximum width *width*. The default for *width* is 80. *Width* should be a multiple of 8 if tabs are present, or the tabs should be expanded using *expand*(1) before using **fold**.

If overstrikes such as underlining are present, **fold** may truncate line inappropriately.

Related Information

See the following command: "tab, untab, expand, unexpand" in topic 1.1.458.

1.1.170 folder

Purpose

Selects and lists folders and messages.

Syntax

```
          +-----------+  +-------------------+  +--------+  +-- -nopack --+
folder ---¦           +--¦                   +--¦        +--¦             +---
          ¦           ¦  ¦      one of       ¦  ¦        ¦  ¦   one of    ¦
          +- +folder -+  ¦ +----------------+ ¦  +- -all -+  ¦ +---------+ ¦
                         +-¦ num      cur  +-+              +-¦ -pack   +-+
                           ¦ sequence .     ¦                 ¦ -nopack ¦
                           ¦ first    next  ¦                 +---------+
                           ¦ prev     last  ¦
                           +----------------+

   +-- -nofast --+  +-- -norecurse --+  +--------------+  +---------------+
   ¦             ¦  ¦                ¦  ¦              ¦  ¦               ¦
 --¦   one of    +--¦     one of     +--¦    one of    +--¦    one of     +---
   ¦ +---------+ ¦  ¦ +------------+ ¦  ¦ +-----------+ ¦  ¦ +-----------+ ¦
   +-¦ -fast   +-+  +-¦ -recurse   +-+  +-¦ -print    +-+  +-¦ -header   +-+
     ¦ -nofast ¦      ¦ -norecurse ¦      ¦ -noprint  ¦      ¦ -noheader ¦
     +---------+      +------------+      +-----------+      +-----------+

   +-- -nototal --+  +-----------+  +--------------+
   ¦              ¦  ¦           ¦  ¦              ¦
 --¦    one of    +--¦  one of   +--¦   one of     +---¦
   ¦ +----------+ ¦  ¦ +-------+ ¦  ¦ +---------+  ¦
   +-¦ -total   +-+  +-¦ -push +-+  +-¦ -list   +--+
     ¦ -nototal ¦      ¦ -pop  ¦      ¦ -nolist ¦
     +----------+      +-------+      +---------+

folder --- -help ---¦
```

Description

The **folder** command sets the current folder and the current message for that folder, and lists information about your folders. The **folder** command is part of the Message Handling (MH) package and can be used with other MH and AIX commands. The **folder** command entered with only the **-all** flag gives a line of information for each folder in your mail directory, telling how many messages are in the folder, what the current message is, the range of message numbers in the folder, and whether it is the current folder. The **folder** command specified without arguments provides information for the current folder. The **-recurse** flag displays information for all folders and sub-folders in your entire mail directory structure.

Use the **+folder** argument to specify a current folder, and specify a message (**+folder msg**) to designate the current message for that folder. You also can create a folder stack to manipulate a group of folders by using the **-push**, **-pop**, and **-list** flags.

Flags

-all Displays a line of information about each folder in your mail directory. Specify **+folder** to display the information about that folder and the sub-folders within that folder's

	directory. Add the **-recurse** flag to display the information about the specified folder and for all sub-folders in all directories under the specified folder's directory. Other options available with the **-all** flag are **-fast**, **-pack**, and **-total**.
+folder msg	Sets the specified **folder** as the current folder. You also can specify a current message using the **msg** argument; use one of the following message references: **num**, **sequence**, **first**, **prev**, **cur**, **.** (period), **next**, or **last**. If you specify **sequence**, that sequence must contain only one message.
-fast	Displays only the names of the folders.
-header	Displays column headings for the folder information.
-help	Displays help information for the command.
-list	Displays the current folder followed by the contents of the folder stack.
-nofast	Displays information about each folder. This flag is the default.
-noheader	Suppresses column headings for the folder information. This flag is the default.
-nolist	Suppresses the display of the folder stack contents. This flag is the default.
-nopack	Does not renumber the messages in the folder option. This flag is the default.
-noprint	Does not display folder information. The **-noprint** option only works when the **-push**, **-pop**, or **-list** flags are specified.
-norecurse	Displays information about the top-level folders in your current folder only. It does not display information about sub-folders. This flag is the default.
-nototal	Does not display a total of all messages and folders in your mail directory structure. The **-total** option is the default when the **-all** flag is specified; otherwise, **-nototal** is the default.
-pack	Renumbers the messages in the specified **folder**. This renumbering eliminates the gaps in the message numbering after messages have been deleted.
-pop	Removes the folder from the top of the folder stack and makes it the current folder. The **+folder** argument cannot be specified with the **-pop** flag.
-print	Displays information about the folders, including the number of messages in each folder, the current message for each folder, and the current folder. If the **-push**, **-pop**, or **-list** flag is specified, the **-noprint** option is the default; otherwise, the **-print** option is the default.

-push	Moves the current folder to the top of the folder stack and sets the specified **folder** as the current folder. If no folder is specified, the **-push** options swaps the current folder with the folder on the top of the folder stack.
-recurse	Displays information about all folders and sub-folders in your current folder. You can specify a folder to display the information about that folder and its sub-folders only.
-total	Displays a total of all messages and folders in your mail directory structure. The **-total** option does not display information for sub-folders unless you also specify the **-recurse** flag. The **-total** flag is the default if the **-all** flag is specified.

Profile Entries

Current-Folder: Sets your default current folder.

Folder-Protect: Sets the protection level for your new folder directories.

Folder-Stack: Specifies your folder stack.

Isproc: Specifies the program used to list the contents of a folder.

Path: Specifies your **user_mh_directory**.

Files

$HOME/.mh_profile MH user profile.

Related Information

See other MH commands: "folders" in topic 1.1.171, "mhpath" in topic 1.1.265, "packf" in topic 1.1.310, and "refile" in topic 1.1.366.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See the "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.171 folders

Purpose

Lists **folders** and messages.

Syntax

```
           +--------------+
folders ---¦ +----------+ +---¦
           +-¦ -fast    +-+
             ¦ -pac     ¦¦
            ¦¦ -recurse ¦¦
            ¦¦ -total   ¦¦
            ¦+----------+¦
            +------------+
```

Description

The **folders** command is used to list information about your **folders**. The **folders** command is part of the MH (Message Handling) package and can be used with other **folders** MH and AIX commands.

The **folders** command is equivalent to the **folder** command specified with the **-all** flag.

Flags

-fast Displays only the names of the **folders** in your mail directory.

-pack Renumbers the messages in the **folders**. This eliminates the gaps in the message numbering after messages have been deleted.

-recurse Displays information about all **folders** and sub-folders in your entire mail directory structure.

-total Displays a total of all messages and **folders** in your mail directory structure. **-total** does not display information for sub-folders unless you also specify the **-recurse** flag. The **-total** flag is the default.

Profile Entries

Current-Folder:	Sets your default current folder.
Folder-Protect:	Sets the protection level for your new folder directories.
Folder-Stack:	Specifies your folder stack.
Isproc:	Specifies the program used to list the contents of a folder.
Path:	Specifies your *user_mh_directory.*

Files

$HOME/.mh_profile	MH user profile.

Related Information

See other MH commands: "folder" in topic 1.1.170, "mhpath" in topic 1.1.265, "packf" in topic 1.1.310, and "refile" in topic 1.1.366.

See the mh-profile file in *AIX Operating System Technical Reference*.

See the "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.172 format, fdformat

Purpose
Formats diskettes.

Syntax

```
         +------------+   +- -d/dev/fd0 -+
format ---¦   +----+   +---¦              +---¦
         +---¦ -f +---+   +-- -ddevice --+
            ¦ -l ¦
           ¦ +----+ ¦
           +--------+

           +------+
fdformat ---¦      +--- device ---¦
           +- -h -+
```

Note: The command is for the PS/2 only.

Description

The **format** command formats diskettes in the specified **device** (**/dev/fd0** by default). The **format** command determines the device type, which is one of the following:

a "5-1/2" low density diskette (360K) containing 40x2 tracks, eac with nine sectors

a "5-1/2" high capacity diskette (1.2M) containing 80x2 tracks, eac with 15 sectors

a "3-1/2" low density diskette (720K) containing 80x2 tracks, eac with nine sectors

a "3-1/2" high capacity diskette (1.4M) containing 80x2 tracks, eac with 18 sectors

The sector size is 512 bytes for all diskette types.

The **fdformat** command formats a diskette for low density unless the **-h** flag is specified. The **format** command formats a diskette for high density unless **device** specifies a different density, or the **-l** flag is specified.

Before formatting a diskette, **format** and **fdformat** prompt for verification. This allows a user to abort the operation cleanly. Formatting a diskette destroys existing data.

Note: The **fdformat** command is placed in **/usr/ucb** as a link to the AIX **format** command.

Flags

-ddevice	Specifies the device containing the diskette to be formatted.
-f	Formats the diskette without checking for bad tracks, thus formatting the diskette faster.

-l	Formats a 360K diskette in a "5-1/2" 1.2M diskette drive. Formats a 720K diskette in a "3-1/2" 1.4M diskette drive. Warning: A 360K diskette drive may not be able to read a 360K diskette that has been formatted in a 1.2M drive.
-h	Force high-density formatting.
device	Device containing the diskette to be formatted (/dev/rfd0 for drive 0).

Files
/dev/rfd[01]

Related Information

See the following command: "flcopy" in topic 1.1.167

See the **fd** file in *AIX Operating System Technical Reference.*

1.1.173 fortune

Purpose

Tells a fortune.

Syntax

```
/usr/games/fortune ---¦
```

Note: This command does not have MBCS support.

Description

The **fortune** game tells a fortune, selected at random from the file **/usr/games/lib/fortunes,** and exits.

You can edit the file **/usr/games/lib/fortunes** to add your own fortunes. Each saying in the file should be a single line. **fortune** folds long sayings into multiple lines as necessary.

1.1.174 forw

Purpose
Forwards messages.

Syntax

```
        +------------------------------------------------------+   +----------
forw ---¦                 +---------------+   +---------------+ +---¦
        +- -digest name -¦               +---¦               +-+   +- -form fi
                          +- -issue num -+   +- -volume num -+

    +-----------------+   +--- -noformat ----+   +-- -noinplace --+   +-------
 ---¦     one of      +---¦      one of      +---¦     one of     +---¦
    ¦ +-------------+ ¦   ¦ +--------------+ ¦   ¦ +------------+ ¦   ¦ +-----
    +-¦ -editor cmd +-+   +-¦ -format      +-+   +-¦ -inplace   +-+   +-¦ -wha
      ¦ -noedit     ¦       ¦ -noformat    ¦       ¦ -noinplace ¦       ¦ -nov
      +-------------+       ¦ -filter file ¦       +------------+       +-----
                            +--------------+

forw --- -help ---¦
```

Description

The **forw** command is used to create a message containing other messages. This command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

By default, the **forw** command copies a message form to a new draft message and invokes an editor. You can then fill in the message header fields **To:** and **Subject:** and fill in or delete the other header fields (such as **cc:** and **Bcc:**). When you exit the editor, the **forw** command invokes the MH command **whatnow**. You can press **Enter** to see a list of the available **whatnow** subcommands. These subcommands enable you to continue editing the message, list the message, direct the disposition of the message, or end the processing of the **forw** command. See "whatnow" in topic 1.1.533 for a description of the subcommands.

You can specify the messages that you want to distribute by using the **+folder msgs** flag. If you do not specify a message, the **forw** command forwards the current message.

You can specify the format of the message by using the **-form** flag. If you do not specify this flag, the **forw** command uses your default message format located in the file **user_mh_directory/forwcomps**. If this file does not exist, the **forw** command uses the system default message format, which is located in the file **/usr/lib/mh/forwcomps**.

Note: The line of dashes or a blank line must be left between the header and the body of the message for the message to be identified when it is sent.

Flags

-annotate Annotates the messages being forwarded with the lines:

```
Forwarded: date
Forwarded: addrs
```

The annotation appears in the original draft message so that you can maintain a complete list of recipients with the original message. If you do not actually redistribute the message using the immediate **forw** command, the **-annotate** flag may fail to provide annotation. The **-inplace** flag forces annotation to be done in place.

-digest *name*

Uses the digest facility to create a new issue for the digest called *name*. The **forw** command expands the format strings in the **components** file (using the same format string mechanism used by the **repl** command) and composes the draft using the standard digest encapsulation algorithm. After the draft has been composed, the **forw** command writes out the volume and issue entries for the digest and invokes the editor.

-draftfolder +*folder*

Places the draft message in the specified folder. If you do not specify this flag, the **forw** command selects a default draft folder according to the information supplied in the MH profiles. You can define a default draft folder in the file **$HOME/.mh_profile**. If the **-draftfolder +***folder* flag is followed by *msg*, *msg* represents the **-draftmessage** attribute.

-draftmessage *msg*

Specifies the draft message. You can specify one of the following message references as *msg*:

num	*sequence*	**firs**
prev	**cur**	**.**
next	**last**	**new**

The default draft message is **new**.

-editor *cmd*

Specifies that *cmd* is the initial editor for preparing the message. If you do not specify this flag, the **forw** command selects a default editor or suppresses the initial edit, according to the information supplied in the MH profiles. You can define a default initial editor in the file **$HOME/.mh_profile**.

-filter *file*

Reformats each message being forwarded and places the reformatted message in the draft message. The **-filter** flag uses the **mhl** command and the specified format file. When you also specify the **-digest** flag, you may want to use the filter file **/usr/lib/mh/mhl.digest.**

+*folder msgs*

Specifies the messages that you want to forward. You can specify *msgs*, which can be several messages, a range of messages, or a single message. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**

last	**all**	**sequen**

The default message is the current message in the current folder. If you specify several messages, the first message forwarded becomes the current message. If you specify a folder, that folder becomes the current folder.

-form *file*	Uses the form contained in the specified file to construct the beginning of the message. The **forw** command treats each line in *file* as a format string. If the **-digest** flag is also specified, the **forw** command uses the form specified in *file* for the format of the digest. If you do not specify a form for a digest, the **forw** command uses the format in the file **user_mh_directory/digestcomps**. If this file does not exist, the **forw** command uses the system default digest form specified in the file **/usr/lib/mh/digestcomps**.
-format	Reformats each message being forwarded and places the reformatted message in the draft message. The **-format** flag uses the **mhl** command and a default format file. If the file **user_mh_directory/mhl.forward** exists, it contains the default format. Otherwise, the file **/usr/lib/mh/mhl.forw** contains the default format.
-help	Displays help information for the command.
-inplace	Forces annotation to be done in place in order to preserve links to the annotated message.
-issue *num*	Specifies the issue number of the digest. The default issue number is one greater than current value of the **digest_name-issue-list** entry in the file **user_mh_directory/context**.
-noannotate	Does not annotate the message. This flag is the default.
-nodraftfolder	Places the draft in the file **user_mh_directory/draft**.
-noedit	Suppresses the initial edit.
-noformat	Does not reformat the messages being forwarded. This flag is the default.
-noinplace	Does not perform annotation in place. This flag is the default.
-nowhatnowproc	Does not invoke a program that guides you through the forwarding tasks. The **-nowhatnowproc** flag also prevents any edit from occurring.
-volume *num*	Specifies the volume number of the digest. The default volume number is the current value of the **digest_name-volume-list** entry in the file

	user_mh_directory/context.
-whatnowproc *cmdstring*	Invokes *cmdstring* as the program to guide you through the forwarding tasks. See "whatnow" in topic 1.1.533 for information about the default **whatnow** program and its subcommands. **Note:** If you specify **whatnow** for *cmdstring*, the **forw** command invokes an internal **whatnow** procedure rather than a program with the file name **whatnow**.

Profile Entries

Current-Folder:	Sets your default current folder.
Draft-Folder:	Sets your default folder for drafts.
Editor:	Sets your default initial editor.
fileproc:	Specifies the program used to refile messages.
mhlproc:	Specifies the program used to filter messages being forwarded.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your **user_mh_directory**.
whatnowproc:	Specifies the program used to prompt "What now?" questions.

Files

/usr/lib/mh/forwcomps	The MH default message skeleton.
user_mh_directory/forwcomps	The user's default message skeleton. (If it exists, it overrides the MH default message skeleton.)
/usr/lib/mh/digestcomps	The MH default message skeleton when the **-digest** flag is specified.
user_mh_directory/digestcomps	The user's default message skeleton when the **-digest** flag is specified. (If it exists, it overrides the MH default message skeleton.)
/usr/lib/mh/mhl.forward	The default MH message filter.
user_mh_directory/mhl.forward	The user's default message filter. (If it exists, it overrides the MH default message filter.)
$HOME/.mh_profile	The MH user profile.
user_mh_directory/draft	The draft file.
user_mh_directory/context	The context file.

Related Information

See other MH commands: "ali" in topic 1.1.17, "anno" in topic 1.1.19, "comp" in topic 1.1.85, "dist" in topic 1.1.131, "dp" in topic 1.1.138, "inc" in topic 1.1.206, "mhl" in topic 1.1.263, "msh" in topic 1.1.281, "repl" in topic 1.1.369, "send" in topic 1.1.416, "whatnow" in topic 1.1.533, and "whom" in topic 1.1.539.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.175 fpr

Purpose
Prints a FORTRAN file.

Syntax

```
fpr ----¦
```

Description
The **fpr** command is a filter that transforms files formatted according to FORTRAN's carriage control conventions into files formatted according to UNIX line printer conventions. The **fpr** command copies its input onto its output, replacing the carriage control characters with characters that will produce the intended effects when printed using **print**. The first character of each line determines the vertical spacing as follows:

Character	**Vertical Space Before Printing**
Blank	One line
0	Two lines
1	To first line of next page
+	No advance

A blank line is treated as if its first character is a blank. A blank that appears as a carriage control character is deleted. A zero (0) is changed to a newline. A one (1) is changed to a form feed. The effects of a "+" are simulated using backspaces.

Examples

```
a.out | fpr | print
fpr < f77.output | print
```

Note
Results are undefined for input lines longer than 170 characters.

1.1.176 from

Purpose

Who is my mail from?

Syntax

```
        +-------------+   +--------+
from ---¦             +---¦        +---¦
        +- -s sender -+   +- user -+
```

Description

The **from** command prints out the mail header lines in your mailbox file to show you who your mail is from. If **user** is specified, then the **user**'s mailbox is examined instead of your own. Normally you cannot read other users' mailboxes. A user must have used the **chmod** command to allow others permission to read his mailbox.

Flags

-s sender Prints headers for mail sent by **sender**.

Files

/usr/mail/*

Related Information

See the "biff" in topic 1.1.41 and "mail, Mail" in topic 1.1.253.

1.1.177 fsck, dfsck

Purpose
Interactively checks consistency and repairs a file system.

Syntax

```
        +--------+   +---------------------+   +-------------+   +-----------
fsck ---¦ one of +---¦ +- -s cyl : skip -+ +---¦ +---------+ +---¦
        ¦ +----+ ¦   +-¦                 +-+   +-¦ -f      +-+   +- -b blocknu
        +-¦ -y +-+     +- -S cyl : skip -+       ¦ -t file ¦¦
          ¦ -n ¦                                ¦+---------+¦     +-----------
          ¦ -p ¦                                +-----------+
          +----+

    +------+   +---------------+   +-----------+   +--------------+
 ---¦      ¦+---¦               +---¦           +---¦            ²+---¦
    +- -c -+   +- -d blocknum -+   +- -i inum -+   +- filesystem -+
                              ¦                ¦                  ¦
                +-------------+     +---------+     +------------+

         +-------------+                   ²  +-------------+
dfsck ---¦             +--- filesystem --- - ---¦             +--- filesystem2
         +- flaglist1 -+                      +- flaglist2 -+
```

¦ The default action is to check every file system with the attribute check=true in the file **/etc/filesystems**.

² Use a minus (**-**) to separate the groups when you specify flags as part of the argument.

Description

Warning: Always run the **fsck** command on file systems **after** a system crash. Corrective actions may result in some loss of data. If you have write permission, the system responds with **yes** or **no**. If you do not have write permission for an affected file, the **fsck** command defaults to a **no** response in spite of your actual response.

Since the **fsck** command rearranges data on the file system, do not run this command on a mounted file system unless you specify the **-n** flag. Otherwise, **fsck** could damage your file system beyond repair.

The **fsck** command checks and interactively repairs inconsistent **filesystem**s. It should be run on every file system as part of system initialization (see "rc" in topic 1.1.354). You must have superuser authority to run the **fsck** command. Normally, the file system is consistent, and the **fsck** command merely reports on the number of files, used blocks and free blocks in the file system. If the **filesystem** is inconsistent, the **fsck** command displays information about the inconsistencies found and prompts you for permission to repair them. The **fsck** command is conservative in its repair efforts and tries to avoid actions that might result in the loss of valid data. In certain cases, however, the command recommends the destruction of a damaged file.

If you do not specify **filesystem**, the **fsck** command looks at the file **/etc/filesystems** to find a list of file systems to check by default. **fsck** can perform checks (on separate arms) in parallel (running in parallel processes). This type of checking can reduce the time required to check a large number of file systems.

In the file **/etc/filesystems**, automatic checking may be enabled by adding a line in the stanza, as follows:

```
  check=true
```

If you specify the **-p** flag, the **fsck** command can perform multiple checks at the same time. To tell this command which file systems are on the same drives, change the **check** specification in the file **/etc/filesystems** to the following:

check=number

The **number** tells the **fsck** command which group contains a particular file system. File systems on a single drive are placed in the same group. Each group is checked in a separate parallel process. File systems are checked, one at a time, in the order that they appear in the file **/etc/filesystems**. All **check=true** file systems are in group 1. The **fsck** command attempts to check the root file system before any other file system regardless of the order specified on the command line or in the file **/etc/filesystems**.

On input and output **fsck** uses units of 4096 bytes for block numbers.

The **fsck** command checks for the following inconsistencies:

Blocks allocated to multiple files or to a file and the free list

Blocks allocated to a file or on the free list outside the range o allowable block numbers.

Discrepancies between the number of directory references to a file an the link count in the file.

Size checks

- Incorrect number of blocks.
- Directory size not 16-byte aligned.

Bad inode format

Blocks not accounted for anywhere

Directory checks

- File pointing to an inode that is not allocated.
- Inode number out of range.
- Dot (.) link missing or not pointing to itself.
- Dot dot (..) link missing or not pointing to the parent directory.
- Files that are not referenced or directories that are not reachable.

Superblock checks

- More than 65536 inodes.
- More blocks for inodes than there are in the file system.

Bad free-block list format

Total free block and/or free inode count incorrect

If Transparent Computing Facility (TCF) is installed

- Low high-water mark.
- Missing commit counts in the superblock list.
- Gaps in the commit count array of the superblock.
- Duplicate commit counts in the superblock array.

Orphaned files and directories (those that cannot be reached) are, if you allow, reconnected by placing them in the **lost+found** subdirectory in the root directory. The name assigned is the inode number. The only restriction is that the directory **lost+found** must already exist in the root directory of the file system being checked and must have empty slots in which entries can be made (accomplished by copying a number of files to the directory and then removing them before you run the **fsck** command). If you do not allow the **fsck** command to reattach an orphaned file, it requests permission to destroy the file.

The **fsck** command records the outcome of its checks and repairs through its exit values. The return codes signal the following conditions:

0	All checked file systems are now okay.
1	Invalid arguments given.
2	**fsck** was interrupted before it could complete the checks.
4	**fsck** changed the mounted file system; the user must restart the system immediately.
8	The file system contains unrepaired damage.
16	System resource error.

When the system is being started up normally, the **fsck** command should be run with the **-p** flag from **/etc/rc**. (See "rc" in topic 1.1.354.) If the **fsck -p** command detects and repairs errors on a file system to be mounted, it displays a message on the console. If errors are detected that it cannot repair, it displays appropriate messages on the console and exits, causing **init** to halt operation after displaying its own message on the screen.

dfsck

The **dfsck** command lets you check two file systems simultaneously on two different drives. Use the **flaglist1** and **flaglist2** arguments to pass flags and parameters for the two sets of file systems. Use a - (minus) to separate the file system groups if you specify flags as part of the arguments.

The **dfsck** command permits you to interact with two **fsck** commands at once. To aid in this, the **dfsck** command displays the file system name with each message. When responding to a question from the **dfsck** command, you must prefix your response with a **1** or **2** to indicate whether the answer refers to the first or second file system group.

Note: Do not use the **dfsck** command to check the root file system **/dev/root**.

Flags

-bblocknum The **blocknums** parameter is a list of block numbers separated by blanks. **fsck** will print out the inode number of all files for which a block specified is allocated. Blocks are in the file system blocksize, which is 4096 bytes on both AIX/370 and AIX/PS2.

-c Instructs the **fsck** command to use the devices specified on the command line and not the raw devices.

-dblocknum Searches for references to a specified disk block. Whenever the **fsck** command encounters a file that contains a specified disk block, it displays the inode number and all path names that refer to it. Specify block numbers in 4096 byte units. Only inodes referring to the individual block number are flagged by error messages. This option does not add the blocks to inode 1.

-f Performs a fast check. Under normal circumstances, the only file systems likely to be affected by halting the system without shutting down properly are those that were mounted when the system stopped. The **-f** flag tells the **fsck** command not to check file systems that were cleanly unmounted. The **fsck** command determines this by inspecting the **s_fmod** flag in the file system superblock. This flag is set whenever a file system is mounted and cleared when it is cleanly unmounted. If a file system was cleanly unmounted, it is unlikely to have any problems. Since most file systems are cleanly unmounted, not checking those file systems can reduce the checking time.

-iinum Searches for references to a specified inode. Whenever the **fsck** command encounters a directory reference to a specified inode number, it displays the full path name of the reference.

-n Assumes a **no** response to all questions asked by the **fsck**command; does not open **filesystem** for writing.

-p Does not display messages about minor problems, but fixes them automatically. This flag does not grant the wholesale license that the **-y** flag does and is useful for performing automatic checks when the system is to be started normally. You should use this flag whenever the **fsck** command is being run automatically as part of the system startup procedures.

When the **fsck -p** command is run, the following inconsistencies are corrected automatically:

Unreferenced inodes
Link counts that are too large
Missing blocks or bad blocks in the free list
Blocks in the free list that are also in files
Counts in the superblock that are wrong (total free inode and free block counts), specified in 4096K byte limits.
Low high-water mark in the superblock (if TCF is installed)
Missing commit counts in the superblock list (if TCF is installed)
Gaps in the superblock commit-count list (if TCF is installed)

If **fsck -p** encounters any other inconsistencies, it exits with an abnormal return status. You can then run **fsck** manually to correct the problems that exist on the file system.

If TCF is installed, the **fsck** command is extended to automatically handle conditions specific to TCF file systems as a result of commit and primary-copy replication. When a non-primary copy of a primary-copy-replicated file system is checked, the **fsck** command skips several of its internal passes, and it avoids checking path names, reference counts, and connectivity, since a non-primary copy need not store the parent directory of every file that it stores.

-s[cyl:skip] Ignores the actual free list and unconditionally reconstructs a new one. You can specify an optional interleave specification with this flag, **cyl** to specify the number of blocks per cylinder and **skip** to specify the number of blocks to skip. If you do not specify **cyl** or **skip**, the **fsck** command uses the interleave parameters in the superblock. The file system should be unmounted while this is done; if this is not possible, be sure that you are running no programs and that you perform a system restart immediately afterwards so that the old copy of the superblock in memory is not written to disk.

-S[cyl:skip] Conditionally reconstructs the free list. This flag is like the **-s** flag except that the free list is rebuilt only if there are no discrepancies discovered in the filesystem. Using **-S** forces a **no** response to all questions asked by the **fsck** command. Use this flag to force free list reorganization on uncontaminated file systems.

-tfile Uses **file** as a scratch file if the **fsck** command cannot obtain enough memory to keep its tables. If you do not specify the **-t** flag and the **fsck** command needs a scratch file, it prompts you for the name of the scratch file. However, if you have specified the **-p** flag, the **fsck** command fails. The file chosen must not be on the file system being checked. If it is not a special file, it is removed when the **fsck** command ends.

-y Assumes a **yes** response to all questions asked by the **fsck** command. This lets the **fsck** command take any action that it considers necessary. Use this flag only on severely damaged file systems.

Examples

1. To check all the default file systems:

```
fsck
```

   This command checks all the file systems marked **check=true** in the file **/etc/filesystems**. This form of the **fsck** command asks you for permission before making any changes to a file system.

2. To fix minor problems with the default file systems automatically:

```
fsck -p
```

3. To check a specific file system:

```
fsck /dev/hd1
```

   This command checks the unmounted file system located on the device **/dev/hd1**.

4. To simultaneously check two file systems on two different drives:

```
dfsck  -p /dev/hd1  -  -p /dev/hd7
```

   This command checks both file systems simultaneously, if the file systems on the devices **/dev/hd1** and **/dev/hd7** are located on two different drives. You can also specify the file system names that are found in the **/etc/filesystems** file.

Files

/etc/filesystems	Contains the default list of file systems to check.

Related Information

See the following commands: "rc" in topic 1.1.354, "fsdb" in topic 1.1.178, "istat" in topic 1.1.218, "mkfs" in topic 1.1.269, "ncheck" in topic 1.1.286, and "shutdown" in topic 1.1.425.

See the **filesystems** and **fs** files in *AIX Operating System Technical Reference*.

See the discussion of the **fsck** and **dfsck** commands in *Managing the AIX Operating System*

1.1.178 fsdb

Purpose
Debugs file systems.

Syntax

```
                       +-----+   +------+
fsdb --- filesystem ---¦     +---¦      +---¦
                       +- - -+   +- -a -+
```

Description

Warning: You should have a thorough working knowledge of the file system structure if you want to use the **fsdb** command. Uninformed use of this command can lead to serious, unrecoverable damage to the file system resulting in extensive data loss. (You must have read and write permission on the device being examined.)

You can use the **fsdb** command to examine and patch a damaged file system after a system crash. It allows you to access blocks and i-numbers and to examine various parts of an inode. You can reference components of the inode symbolically. These features simplify procedures for correcting control-block entries or for descending the file system tree.

The file system to be examined can be specified by a block device name or a raw device name.

Any numbers you enter are considered decimal by default, unless you prefix them with a 0 (zero) to indicate an octal number.

Because the **fsdb** command reads and writes one block at a time, it works with raw as well as with block I/O. It uses a buffer management routine to retain commonly used blocks of data in order to reduce the number of **read** system calls. All assignment operations write the corresponding block immediately.

Flags

- Disables the error checking routines used to verify inode and block addresses. The **O** subcommand toggles these routines on and off. When these routines are running, the **fsdb** command reads the i-size and f-size entries from the superblock of the file system.

-a Always execute for the file system specified, even if the magic number in the superblock is invalid. Normally, the **fsdb** command does not work if the superblock magic number is invalid.

Subcommands
The **fsdb** subcommands allow you to display or modify information. A display subcommand is a block address optionally followed by a display format specification. A field modification subcommand is similar to the display subcommand but may include a subfield specification, an operator, and a value. An address specification is a number optionally followed by a type specifier and subfield specification.

The display subcommands are:

CL Displays the commit list if the file system is

	replicated.
num	Displays data at absolute address **num**.
i-numberi	Displays data at **i-number**.
block-addressb	Displays data at **block-address**, specified in 4096-byte units.
ddirectory-slot-offset	Displays data at **directory-slot-offset**.
S	Displays the superblock.
q	Quits.
A	Aborts and causes a core dump.
?	Prints a synopsis of the **fsdb** subcommands.
!	Escapes to the shell.

The display formats are:

p	General display facilities.
f	File display facility.

You can step through the inode information examining each byte, word, or double word. Select the desired display mode by entering one of the following subcommands:

B	Begins displaying in byte mode.
D	Begins displaying in double word mode.
W	Begins displaying in word mode.
O	Toggles error checking on or off.

Moving forward or backward through the inode data is done with the following symbols:

+num	Moves forward the specified number of units currently in effect.
-num	Moves backward the specified number of units currently in effect.

The following symbols allow you to store the current address and return to it conveniently:

>	Stores current address for later reference.
<	Returns to the previously stored address.

The display format applied to the information at the selected address is the one currently in effect. You may receive an error message indicating improper alignment if the address you specify does not fall on an even boundary.

The display facilities display a formatted output in various styles. The current address is normalized to an appropriate boundary before display begins. It advances with the displaying and is left at the address of the last item displayed. The output can be ended at any time by pressing INTERRUPT (**Ctrl-C**).

If you enter a number after the **p** symbol, the **fsdb** command displays that number of entries. A check is made to detect block boundary overflows because logically sequential blocks are generally not physically sequential. If you enter a count of zero, the **fsdb** command displays all entries to the end of the current block.

The display formats available are:

i	Displays as inodes.

d	Displays as directories.
o	Displays as octal words.
e	Displays as decimal words.
c	Displays as characters.
b	Displays as octal bytes.
r	Displays as small-block directory.
x	Displays as hex words.
y	Displays as hex bytes.

Use the **f** symbol to display data blocks associated with the current inode. If you enter a number after **f**, the **fsdb** command displays that block of the file. Block numbering begins at zero. The desired display subcommand follows the block number, if present, or the **f** symbol. The display facility works for large as well as small files. It checks for special devices and also checks the data are not zero.

You can use . (dots), tabs, and spaces as subcommand delimiters, but they are not necessary. Pressing just the **Enter** key (entering a blank line) increments the current address by the size of the data type last displayed. That is, the address is set to the next byte, word, double word, directory entry or inode, allowing you to step through a region of a file system. The **fsdb** command displays information in a format appropriate to the data type. Bytes, words and double words are displayed as an octal address followed by the octal representation of the data at that address and the decimal equivalent enclosed in parentheses. The **fsdb** adds a **.B** or **.D** to the end of the address to indicate a display of byte or double word values. It displays directories as a directory slot offset followed by the decimal i-number and the character representation of the entry name. The command displays inodes with labeled fields describing each element. The environment variables **LANG** and **LC_TIME** control the formats of the date and time.

The following mnemonics are used for the names of the fields of an inode and refer to the current working inode:

md	Permission mode
ln	Link count
uid	User number
gid	Group number
sz	File size
nm	Directory name
cc	Commit count
fs	fstore value
qid	Unique creation ID
v	Version
sb	Small-block flag
an	Data block numbers (0 - 12)
at	Access time
mt	Modification time
maj	Major device number
min	Minor device number
df	Inode dflags
site	Device site number

The following mnemonics are used to modify super-block information:

M	Magic number (valid only if **-a** option is used)
fsz	File system size
gfs	Global file system number
gp	gfs pack number

isz	Inode table size
t	Modification time
V	File system version
Fs	fstore value
I	Free inode list in superblock
SF	Free block list in superblock
H	High-water mark
L	Low-water mark
P	Read-only flag
C	Free-block list offset
CL	Commit list for replicated file system

The following mnemonics are used for small-block examination:

zf	Small-block flag
zd	Small-block data area
zr	Small-block directory slot

The general form for assigning new values is:

mnemonic operator new-value

The **fsdb** command modifies the value of the field specified by **mnemonic** according to the **operator** and **new-value**.

Valid operators include:

=	Assign **new-value** to the specified **mnemonic**.
=+	Increment the **mnemonic** by the specified **new-value**. The default **new-value** is 1.
=-	Decrease the **mnemonic** by the specified **new-value**. The default **new-value** is 1.
="	Assign character string **new-value** to the specified **mnemonic**.

Examples

The following examples show subcommands that you can use after starting the **fsdb** command.

1. To display an inode:

   ```
   386i
   ```

   This command displays i-number **386** in inode format. It now becomes the current inode.

2. To change the link count for the current inode to **4**:

   ```
   ln=4
   ```

3. To increase the link count of the current inode by **1**:

   ```
   ln=+1
   ```

4. To display part of the file associated with the current inode:

   ```
   fc
   ```

   This displays as ASCII text block zero of the file associated with the current inode.

5. To display entries of a directory:

   ```
   2ifd
   ```

   This changes the current inode to the root inode (inode **2**), then displays the directory entries in the first block associated with that inode.

6. To go down a level of the directory tree:

   ```
   d5ifc
   ```

   This changes the current inode to the one associated with directory entry **5**. Then it displays the first block of the file as ASCII text (**fc**). Directory entries are numbered starting from 0 (zero).

7. To display a block when you know its block number:

   ```
   1bp0o
   ```

   This displays the superblock (block **1**) of file system in octal.

8. To change the i-number of a directory entry:

   ```
   2.a0b.d7=3
   ```

   This changes the i-number of directory entry **7** in the root directory (**2i**) to **3**. This example also shows how several operations can be combined on one line.

9. To change the file name of a directory entry:

   ```
   d7nm="chap1.rec"
   ```

   This changes the name field of directory entry **7** to **chap1.rec**.

10. To display a given block of the file associated with the current inode:

    ```
    a2bp0d
    ```

    This displays block **2** of the current inode as directory entries.

11. To display the superblock:

    ```
    S
    ```

    This displays the fields of the superblock for this file system.

12. To display the commit list:

    ```
        CL
    ```

    This displays the commit list if the file system is replicated.

13. To display directory entries in a small-block file:

    ```
    zdp0r
    ```

This displays the small-block area of the current inode as directory entries.

14. To display the data in a small-block file:

    ```
    zdp200c
    ```

    This displays the first 200 characters of the small-block file for the current inode.

15. To change a directory entry in a small-block file:

    ```
    zr7=100
    ```

    This changes the i-number of the seventh directory slot in the current small-block file to 100.

16. To change the name field in a small-block directory entry:

    ```
    zr5nm="new.name"
    ```

    This changes the name field of small-block directory entry five to **new.name**.

17. To go down a level in a small-block directory tree:

    ```
    zr8i
    ```

    This changes the current inode to be the inode corresponding to the directory entry number **8** in the small-block file.

Related Information

See the following commands: "fsck, dfsck" in topic 1.1.177.

See the **fs** and **dir** files and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See the "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.179 fsplit

Purpose

Splits a multi-routine Fortran file into individual files.

Syntax

```
          +- file -+
fsplit ---¦        +---¦
          +-- -e --+
```

Description

The **fsplit** command takes as input either a file or standard input containing Fortran source code. It attempts to split the input into separate routine files of the form **name.f**, where **name** is the name of the program unit (for example, function, subroutine, block data or program). The name for unnamed block data subprograms has the form **blkdtaNNN.f** where NNN is three digits and a file of this name does not already exist. For unnamed main programs the name has the form **mainNNN.f.** If there is an error in classifying a program unit, or if **name,f** already exists, the program unit will be put in a file of the form **zzzNNN.f**, where **zzzNNN.f** does not already exist.

Flags

-e Causes only the specified subprogram units to be split into separate files. (Normally each subprogram unit is split into a separate file.) For example,

```
fsplit -e readit -e doit prog.f
```

will split "readit" and doit into separate files.

Diagnostics

If names specified via the **-e** option are not found, a diagnostic is written to **standard error**.

1.1.180 fstorchkdmon

Purpose

Syntax

```
fstorchkdmon  -----¦
```

Note: This command does not have MBCS support.

Description

The **fstorchkdmon** command is a shell script, often set up to run by **cron** on the cluster site which has the primary copy of the root file system. This command runs **chkfstore**, uses **ncheck** to determine the pathnames of the 0-fstore files, and sends mail to the mail alias "fstore", which the administrator should set up using the **newaliases** command.

Related Information

See the following commands: "ncheck" in topic 1.1.286, "sendmail, mailq, newaliases" in topic 1.1.417, "chkfstore" in topic 1.1.66 and "chfstore" in topic 1.1.64.

1.1.181 fs2fd

Purpose
Copies a file system to a boot diskette set.

Syntax

```
        +---------------+
fs2fd ---¦ +-----------+ +--- source --- diskette ---¦
        +-¦ -b program +-+
          ¦ -h         ¦¦
         ¦¦ -n program ¦¦
         ¦¦ -o         ¦¦
         ¦¦ -q         ¦¦
         ¦+-----------+¦
         +-------------+
```

Note: This command is for the PS/2 only.

Description
The **fs2fd** command is intended for use by system administrators to use within the **custboot** command. The **fs2fd** command copies a file system to a boot diskette set. In the process, the identification field of each diskette that is written to is checked, ensuring that diskettes are changed between writes.

The following information is helpful when using this command:

The set of diskettes can be from one to ten diskettes

The number of blocks that fit on each diskette is determined from th **devinfo** structure for the diskette device (see the **ioctl** system call and the **devinfo** file in the *AIX Operating System Technical Reference*).

The first block on each diskette in the set is reserved for boo programs and diskette identification.

Diskette information is stored in bytes 3 through 10, in the form o DISKID**n**, where "**n**" represents the number of the diskette set ("0" through "9").

The identification for the first diskette must be part of the progra stored in the boot block.

fs2fd checks the identification field of each diskette that is being written to. It requests confirmation before overwriting the book block. This ensures that diskettes were changed between writes. If previously used diskettes are written to, this field may contain the value of the diskette written before.

The identification is stored on subsequent diskettes by **fs2fd**; therefore, any program stored in the boot blocks of those diskettes must provide free space for the identification.

The following information is automatically displayed when using **fs2fd**:

The number of blocks that are available on each diskette (in 4 bytes).

The sizes of the programs loaded into the first diskette blocks (i
bytes).

The size of the file system to be copied (in 4k byte units)

A period (".") for each block written to diskette

A message that the diskette identification was written on eac
diskette after the first.

Note: The **-q** option suppresses all of these messages.

Flags

-b Specifies the name of the program to be loaded into the boot block of the first diskette in the set.

-h Print a verbose usage message, and exit. This flag causes **fs2fd** to ignore all other legitimate flags.

-n Specifies the name of the program to be loaded into the boot block of diskettes 2 through 9.

-o Overwrite (ignore) presence of disk identification information on diskettes to be written. Disables checking of the diskette identification field.

-q Suppress information messages.

Examples

To copy a file system to diskettes, without information messages

```
fs2fd -q /tmp/bootimage /dev/fd0
```

To copy a file system to diskettes, and load a boot program on th
first diskette, ignoring any previous diskette identification. This
permits re-use of diskettes without reformatting:

```
fs2fd -o /etc/bootprogram /tmp/bootimage /dev/fd0
```

Related Information
See the "**mkfs**" command in the *AIX Operating System Commands Reference*.

1.1.182 fuser

Purpose
Identifies processes using a file or file structure.

Syntax

```
          +--------+   +- -n namelist -+   +-----+
fuser ---¦ +----+ +---+- -c corefile -+---¦     +---¦
          +-¦ -k +-+   +---- file -----+   +- - -+ ¦
        ¦   ¦ -u ¦¦                ¦                ¦
        ¦  ¦+----+¦      +----------+               ¦
        ¦  +------+                                 ¦
        +-------------------------------------------+
```

Description
The **fuser** command lists the process numbers of the processes on the local cluster site which are using the specified **file**. For block special devices, all processes using any file on that device are listed. The process number is followed by a letter indicating how the process is using the file:

c Using **file** as the current directory
p Using **file** as the parent of the current directory (only when in use by the system)
r Using **file** as the root directory.

The process numbers are written as a single line to standard output, separated by spaces and ended with a single new-line character. All other output is written to standard error.

Flags

-c corefile	Reports on the specified core file.
-k	Sends the **SIGKILL** signal to each process. Only the person operating with superuser authority can kill another user's process (see "kill" in topic 1.1.221).
-n namelist	Reports on the kernel files specified by the **namelist** argument.
-u	Indicates the login name in (parentheses) after the process number.
-	Cancels any flags selected for the previous set of file or files.

Flags may be respecified between groups of files on the command line. The new set of flags replaces the old set.

Examples

1. To list the ID numbers of the processes using the **/etc/passwd** file:

   ```
   fuser  /etc/passwd
   ```

2. To list the process IDs and user names of the processes using the **/etc/filesystems** file:

```
fuser  -u  /etc/filesystems
```

3. To kill all of the processes using a given disk drive:

   ```
   fuser  -k  -u  /dev/hd1
   ```

   This command lists the process ID and user name, and then stops each process that is using the **/dev/hd1** disk drive. You must have superuser authority to stop processes that belong to someone else. You might want to do this if you are trying to unmount the disk drive **/dev/hd1**, and a process accessing it is preventing you from doing so.

4. To perform the actions of the previous examples in reverse order:

   ```
   fuser  -k  -u  /dev/hd1  -  -u  /etc/filesystems  -  /etc/passwd
   ```

   Lone hyphens before the **-u** and before **/etc/passwd** turn off both the **-k** and **-u** flags.

Files

/unix	System kernel image.
/dev/kmem	Default system image file.
/dev/mem	Default user memory.

Related Information

See the following commands: "killall" in topic 1.1.222, "mount" in topic 1.1.278, and "ps" in topic 1.1.337.

See the **kill** and **signal** system calls in *AIX Operating System Technical Reference*.

1.1.183 fwtmp, wtmpfix, acctwtmp

Purpose
Manipulates connect accounting records.

Syntax

```
                       +--------+
/usr/lib/acct/fwtmp ---¦ +----+ +---¦
                       +-¦ -i +-+
                         ¦ -c ¦
                        ¦+----+¦
                        +------+

                         +------------+
/usr/lib/acct/wtmpfix ---¦            +---¦
                         +--- file ---+
                                    ¦
                           +--------+

/usr/lib/acct/acctwtmp --- "reason" ---¦
```

Description

1.1.183.1 fwtmp

The **fwtmp** command reads **wtmp** records from standard input and converts them to formatted ASCII records, which it writes to standard output.

Flags

-i	Reads ASCII.
-c	Writes binary output.

1.1.183.2 acctwtmp

The **acctwtmp** command writes to standard output a **utmp** record containing the string **reason** and the current date and time. The string **reason** can contain 11 characters or less.

1.1.183.3 wtmpfix

The **wtmpfix** command examines standard input or the named **file**s containing records in **wtmp** format, corrects the date and time stamps to make the entries consistent, and writes the corrected input to standard output. (It is necessary that date and time stamps be consistent because **acctcon1** generates an error and stops when it encounters inconsistent date change records.)

Each time the date is set (on system startup or with the **date** command) a pair of date change records is written to the file **/usr/adm/wtmp**. The first record is the old date, denoted by the string **old time** placed in the line field and the flag **OLD_TIME** placed in the type field. The second record is the new date, denoted by the string **new time** placed in the line field and the flag **NEW_TIME** placed in the type field. The **wtmpfix** command uses these records to synchronize all date and time stamps in the file.

In addition to correcting date and time stamps, the **wtmpfix** command checks the validity of the name field to ensure that it consists solely of alphanumeric characters, a dollar sign (**$**), or spaces. If it encounters an invalid name, it changes the login name to **INVALID** and writes a diagnostic to standard error. In this way, the **wtmpfix** command reduces the chance that the **accton2** command will fail when it processes connect accounting records.

Files

/usr/adm/wtmp	Contains records of date changes that include an old date and a new date.
/usr/include/utmp.h	Data structure for **/etc/utmp** and **/usr/adm/utmp** entries.

Related Information

See the following commands: "acct/*" in topic 1.1.6, "acctcms" in topic 1.1.7, "acctcom" in topic 1.1.8, "acctcon1, acctcon2" in topic 1.1.9, "acctdisk, acctdusg" in topic 1.1.10, "acctmerg" in topic 1.1.11 and "runacct" in topic 1.1.398.

See the **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

1.1.184 gencat

Purpose
Generates a message catalog.

Syntax

```
                       +------------+
gencat --- catfile ---¦            +---¦
                       +- descfile -+ ¦
                     +----------------+
```

Description

The **gencat** command generates a message catalog. It can be used to add, change, or delete messages within an existing message catalog. The **gencat** command takes as arguments a message catalog name (**catfile**) and one or more optional message catalog descriptor files (**descfile**).

The descriptor files contain commands to **gencat** that specify message set numbers, message numbers, and text that make up messages in the catalog. The **gencat** command reads from standard input if no descriptor files are specified on the command line. If the message catalog does not exist, a new message catalog with the name **catfile** is created. If the catalog does exist, the messages in the catalog are read into memory and are acted upon by the commands specified in the descriptor files. The descriptor files are completely processed in the order specified to assure consistent message catalog generation.

The **gencat** command first searches the directory **/usr/lib/mbcs/msg/language.codeset/** for **catfile**. The **language.codeset** depends on the current locale. (The locale may be changed using the **LANG** environment variable.) For example, if the current locale is English, the default search directory would be **/usr/lib/mbcs/msg/En.pc850/**. If **catfile** is not found in the default search directory, **gencat** searches the current directory for **catfile**.

Message Text Source File Format

The format of a message text source file is defined as follows. Predefined values and limits for message text source files are defined in the files **/usr/include/limits.h** and **/usr/include/nl_types.h**. The fields of a message text source line are separated by a single ASCII space or tab character. Any other ASCII spaces or tabs are considered to be part of the subsequent field.

$set n comment
Specifies the set identifier of the following messages until the next **$set**, **$delset**, or end-of-file appears. The **n** denotes the set identifier, defined in the range 1...**NL_SETMAX**. Set identifiers must be presented in ascending order within a single source file but need not be contiguous. Any string following the set identifier is treated as a comment. If no **$set** directive is specified in a message text source file, all messages will be located in an implementation-defined default message set **NL_SETD**.

delset n comment
Deletes message set **n** from an existing message catalog. The **n**

denotes the set number 1...**NL_SETMAX**. Any string following the set number is treated as a comment.

$ comment
A line beginning with $ followed by an ASCII space or tab character is treated as a comment.

m message text
The **m** denotes the message identifier, which is defined as a number in the range 1...**NL_MSGMAX**. The **message text** is stored in the message catalog with the set identifier specified by the last **$set** directive and with message identifier **m**. If the **message text** is empty and an ASCII space or tab field separator is present, an empty string is stored in the message catalog. If the message source line has a message number but neither a field separator nor **message text**, the existing message with that number (if any) is deleted from the catalog. Message identifiers must be in ascending order with a single set but need not be contiguous. The length of **message text** must be in the range 0...**NL_TEXTMAX**.

$quote c
Specifies an optional quote character **c**, which can be used to surround **message text** so that trailing spaces or empty messages are visible in a message source line. By default or if an empty **$quote** directive is supplied, no quoting of **message** is recognized.

Empty lines in a message text source file are ignored. The effects of lines starting with any character other than those defined above are considered empty lines and are ignored.

Text strings can contain the special characters and escape sequences defined as follows:

\n	New line
\t	Horizontal tab
\v	Vertical tab
\b	Backspace
\r	Carriage return
\f	Form feed
\	Backslash
\ddd	Octal digit; consists of a backslash followed by one, two or three octal digits, which specify the value of the desired character. If the character following a backslash is not one of those specified, the backslash is ignored.
\xdddd	Hexadecimal digit; represents one, two, or four hexadecimal digits, which specify the value of the desired character. If a character that can be represented as a hexadecimal digit follows a character that represents a hexadecimal digit, a space must separate the two characters.

Messages larger than **NL_TEXTMAX** bytes are truncated to the last character in the message that fits into **NL_TEXTMAX** bytes. This method guarantees truncation does not occur in the middle of a multibyte character.

Related Information

See the following commands: "mkcatdefs" in topic 1.1.267, "dspcat" in topic 1.1.139, "dspmsg" in topic 1.1.140, and "runcat" in topic 1.1.399.

See the **catopen** subroutine in the *AIX Technical Reference*.

See the "Message Catalog Generation" section of the "International Character Support" chapter of the *AIX Operating System Programming Tools and Interfaces*.

See the "Messages" chapter in the *AIX MBCS Guide*.

1.1.185 genxlt

Purpose

Creates a translation table for ASCII-to-EBCDIC or EBCDIC-to-ASCII conversion.

Syntax

```
        +--------------+
genxlt---¦              +--¦
        +-- filename --+
```

Note: This command is for the System/370 only.

Note: This command does not have MBCS support.

Description

The **genxlt** utility reads a source translation file from standard input and writes the compiled version to standard output, or optionally, to **filename**.

The source translation file contains directives that the **genxlt** program uses to produce the compiled version of the translation table. Each line in a translation source file uses one of the following two formats:

# comment A line whose initial **non-white-space** character is the # (**comment**) symbol. The text line that follows this symbol is a **comment** and therefore ignored by the program.

offset tab value [**white-space comment**]
Offset is a hexadecimal value of the form **0Xnn** which designates the source byte value to be translated to **value**. The value **n** in **0Xnn** may be any hexadecimal digit from the set: 0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F. There are four field descriptors:

tab	The TAB character.
value	The **value** is a hexadecimal value of the form **0Xnn** to which the byte corresponding to **offset** will be translated. The rules for **offset** for values of **n 0Xnn** also apply to value.
white-space	Either a TAB or a SPACE character.
comment	The **comment** is optional.

If the same **value** for **offset** is found in the source translation file multiple times, the last entry will be used in the compilation of the translation table.

Entries that do not conform to the above syntax are considered to be in error. If there is no specification of an **offset** which corresponds to a specific byte offset, this is also an error. An erroneous source file causes an error message to be written to standard error. The state of the output is undefined when an error is found.

Related Information

"axeb, ebxa" in topic 1.1.30, "uucp" in topic 1.1.506 and "uvcp" in topic 1.1.517.

1.1.186 get

Purpose
Extracts a specified version of a Source Code Control System (SCCS) file.

Syntax

```
       +-------------------------------+   +---------+   +------------+
get ---¦ +-------------------------+ +---¦ one of  +---¦            +---¦
       +-¦ -b        -k    -s        +-+   ¦ +-----+ ¦   +--- file ---+
         ¦ -c cutoff -     -t        ¦¦    +-¦ -l  +-+            ¦
        ¦¦ -e        -n    -x list   ¦¦      ¦ -lp ¦        +--------+
        ¦¦ -g        -p    -w string ¦¦      +-----+
        ¦¦ -i list   -rSID           ¦¦
        ¦+---------------------------+¦
        +-----------------------------+
```

Description
The **get** command reads the specified versions of the named Source Code Control System (SCCS) **file**s, creates an ASCII text file for each **file** according to the specified flags, and writes each text file to a file with the same name as the original SCCS file without the **s.** (s period) prefix (the **g-file**). The flags and **file**s can be specified in any order, and all flags apply to all named files.

If you specify a directory in place of **file**, the **get** command performs the requested actions on all the files in the directory that begin with the **s.** prefix. If you specify a **-** (minus) in place of a **file**, the **get** command reads standard input and interprets each line as the same of an SCCS file. The **get** command continues to read input until it reads END OF FILE (**Ctrl-D**).

If the effective user has write permission in the directory containing the SCCS files but the real user does not, only one file can be named when the **-e** flag is used.

If you are not familiar with the terms **SID** and **delta** or you do not know the numbering system of the deltas, see *AIX Operating System Programming Tools and Interfaces* for more information.

Subtopics

1.1.186.1 SCCS Files

In addition to the file with the **s.** prefix (the **s-file**), the **get** command can create several auxiliary files: the **g-file**, **l-file**, **p-file**, and **z-file**. These files are identified by their **tag**, the letter before the hyphen. The **get** command names auxiliary files by replacing the leading **s.** in the SCCS file name with the proper tag, except for the **g-file**, which is named by removing the **s.** prefix. So, for a file named **s.sample**, the auxiliary file names would be **sample**, **l.sample**, **p.sample**, and **z.sample**.

These files serve the following purposes:

s-file This file contains the original file text and all the changes (***deltas***) made to the file. It also includes information about who can change the file contents, who has made changes, when those changes were made, and what the changes were. You cannot edit this file directly since the file is read-only. It contains the information needed by the SCCS commands to build the **g-file**, the file you can edit.

g-file The **g-file** is an ASCII text file that contains the text of the SCCS file version that you specify with the **-r** flag (or the latest trunk version by default). You can edit this file directly. When you have made all your changes and you want to make a new delta to the file, you can then apply the **delta** command to the file. The **get** command creates the **g-file** in the current directory.

The **get** command creates a **g-file** whenever it runs, unless the **-g** flag or the **-p** flag is specified. The real user owns it (not the effective user). If you do not specify the **-k** or the **-e** flag, the file is read-only. If the **-k** or the **-e** flag is specified, the owner has write permission for the **g-file** You must have write permission in the current directory to create a **g-file**.

l-file The **get** command creates the **l-file** when the **-l** flag is specified. The **l-file** is a read only file. It contains a table showing which deltas were applied in generating the **g-file**. You must have write permission in the current directory to create an **l-file**. Lines in the **l-file** have the following format:

1. A blank character if the delta was applied; a ***** appears otherwise.

2. A blank character if the delta was applied or was not applied and ignored; a ***** appears if the delta was not applied and was not ignored.

3. A code indicating a special reason why the delta was or was not applied:

Blank	Included or excluded normally.
I	Included using the **-i** flag.
X	Excluded using the **-x** flag.
C	Cut off using the **-c** flag.

4. The SID.

5. The date and time the file was created.

6. The login name of person who created the delta.

Comments and MR data follow on subsequent lines, indented one horizontal tab character. A blank line ends each entry.

For example, for a delta cutoff with the **-c** flag, the entry in the **l-file** might be:

```
  **C 1.3 85/03/13 12:44:16 pat
```

and the entry for the initial delta might be:

```
      1.1 85/02/27 15:42:20 pat
  date and time created 85/02/27 15:42:20 by pat
```

p-file The **get** command creates the **p-file** when the **-e** or the **-k** flag is specified. The **p-file** passes information resulting from a **get -e** to a **delta** command. The **p-file** also prevents a subsequent execution of **get** with a **-e** flag for the same SID until **delta** is run or the joint edit keyletter (**j**) is set in the SCCS file. The **j** keyletter allows several **get**s on the same SID. The **p-file** is created in the directory containing the SCCS **file**. To create a **p-file** in the SCCS directory, you must have write permission in that directory. The permission code of the **p-file** is read-only to all but its owner, and it is owned by the effective user. The **p-file** contains:

The current SID
The SID of new delta to be created
The user name
The date and time of the **get**
The **-i** flag, if it was present
The **-x** flag, if it was present

The **p-file** contains an entry with the above information for each pending delta for the file. No two lines have the same new delta SID.

z-file The **z-file** is a lock mechanism against simultaneous updates. The **z-file** contains the binary process number of the **get** command that created it. It is created in the directory containing the SCCS file and exists only while the **get** command is running.

When you use the **get** command, it displays the SID being accessed and the number of lines created from the SCCS file. If you specify the **-e** flag, the SID of the delta to be made appears after the SID accessed and before the number of lines created. If you specify more than one file, or a directory, or standard input, the **get** command displays the file name before each file is processed. If you specify the **-i** flag, the **get** command lists included deltas below the word **Included**. If you specify the **-x** flag, the **get** command lists excluded deltas below the word **Excluded**.

1.1.186.2 Identification Keywords

You can use identification keywords in your files to insert identifying information. These keywords are replaced by their values in the **g-file** when the **get** command is invoked without the **-e** or **-k** flag.

Note: For unformatted files, insert a null effect troff sequence into the keywords so that sccs will not expand them; for example, **%M%** becomes **%M\&%**.

The following identification keywords can be used in SCCS files:

%M%	Module name: the value of the **m** flag in the SCCS file
%I%	The SID (%R%.%L%.%B%.%S%) of the **g-file**
%R%	Release
%L%	Level
%B%	Branch
%S%	Sequence
%D%	Date of the current **get** (YY/MM/DD)
%H%	Date of the current **get** (MM/DD/YY)
%T%	Time of the current **get** (HH:MM:SS)
%E%	Date newest applied delta was created (YY/MM/DD)
%G%	Date newest applied delta was created (MM/DD/YY)
%U%	Time newest applied delta was created (HH:MM:SS)
%Y%	Module type: the value of the **t** flag in the SCCS file
%F%	SCCS file name
%P%	Full path name of the SCCS file
%Q%	The value of the **q** flag in the file
%C%	The current line number. This keyword is intended for identifying messages output by the program. It is not intended to be used on every line to provide sequence numbers.
%Z%	The 4-character string **@(#)** recognized by the **what** command
%W%	A shorthand notation for constructing **what** command strings for AIX program files. Its value is the characters and keyletters: %W% = %Z%%M%<horizontal-tab>%I%
%A%	Another shorthand notation for constructing **what** command strings for non-AIX program files. Its value is the keyletters: %A% = %Z%%Y% %M% %I%%Z%

Table 1-4 illustrates how the **get** command determines the SID of the file it retrieves, and what the pending SID is. The column **SID Specified** shows the various ways the SID can be specified with the **-r** flag. The two columns illustrate the various conditions that can exist, including whether the **-b** flag is used with the **get -e** command. The **SID Retrieved** indicates the SID of the file that makes up the **g-file**. The **SID of Delta to be Created** column indicates the SID of the version that will is created when the **delta** command is applied.

Table 1-4. SID Determination for get Command

SID Specified	**-b Used**	**Other Conditions**	**SID Retrieve**	**SID of Delta to be Created**

none¦	no	R defaults to mR[2]	mR.mL	mR.(mL+1)
none¦	yes	R defaults to mR	mR.mL	mR.mL.(mB+1).
(R)elease	no	R > mR	mR.mL	R.1¦
R	no	R = mR	mR.mL	mR.(mL+1)
R	yes	R > mR	mR.mL	mR.mL.(mB+1).
R	yes	R = mR	mR.mL	mR.mL.(mB+1).
R	N/A	R < mR and R does not exist	hR.mL4	hR.mL.(mB+1).
R	N/A	R < mR and R exists	R.mL	R.mL.(mB+1).1
R.(L)evel	no	No trunk successor	R.L	R.(L+1)
R.L	yes	No trunk successor	R.L	R.L(mB+1).1
R.L	N/A	Trunk successor in release = R	R.L	R.L.(mB+1).1
R.L.(B)ranch	no	No branch successor	R.L.B.mS	R.L.B.(mS+1)
R.L.B	yes	No branch successor	R.L.B.mS	R.L.(mB+1).1
R.L.B.(S)eque	cno	No branch successor	R.L.B.S	R.L.B.(S+1)
R.L.B.S	yes	No branch successor	R.L.B.S	R.L.(mB+1).1
R.L.B.S	N/A	Branch successor	R.L.B.S	R.L.(mB+1).1

¦ Applies only if the **d** (default SID) flag is not present in the file (see "admin" in topic 1.1.16)
[2] The mR indicates the maximum existing release.
¦ Forces creation of the first delta in a new release.
4 The hR is the highest existing release that is lower than the specified, nonexistent, release R.

Flags

-b Specifies that the delta to be created should have an SID in a new branch. The new SID is numbered according to the rules stated in Table 1-4. You can use the **-b** flag only with the **-e** flag. It is necessary only when you want to branch from a **leaf delta** (a delta without a successor). Attempting to create a delta at a nonleaf delta automatically results in a branch, even if the **b** header flag is not set. If you do not specify the **b** header flag in the SCCS file, the **get** command ignores the **-b** flag because the file does not allow branching (see the discussion of header flags on page 1.1.16.3).

-c cutoff Specifies a **cutoff** date and time, in the form: **YY[MM[DD[HH[MM[SS]]]]]**. The **get** command includes no deltas to the SCCS file created after the specified **cutoff** in the **g-file**. The values of any unspecified items in the **cutoff** default to their maximum allowable values. Thus, a cutoff date and time specified with only the year (**YY**) specifies the last month, day, hour, minute, and second of that year. Any number of non-numeric characters can separate the two-digit items of the **cutoff** date and time. This allows you to specify a date and time in a number of ways, as follows:

```
-c85/9/2,9:00:00
-c"85/9/2 9:00:00"
"-c85/9/2 9:00:00"
```

-e	Indicates that the **g-file** being created is to be edited by the user applying the **get** command. The changes are recorded later with the **delta** command. **get -e** creates a **p-file** that prevents other users from issuing another **get -e** command and editing a second **g-file** on the same SID before **delta** is run. The owner of the file can override this restriction by allowing joint editing on the same SID through the use of the **admin** command with the **-fj** flag. Other users, with permission, can obtain read-only copies by using the **get** command without the **-e** flag. The **get -e** command enforces SCCS file protection specified via the ceiling, floor, and authorized user list in the SCCS file (see "admin" in topic 1.1.16).
-g	Suppresses the actual retrieval of text from the SCCS file. Use the **-g** flag primarily to create an l-file or to verify the existence of a particular SID. Do not use it with the **-e** flag.

-i list	Specifies a **list** of deltas to be included in the creation of a **g-file**. The **SID list format** consists of a combination of individual SIDs separated by commas and SID ranges indicated by two SIDs separated by a hyphen. You specify the same SIDs with both the following command lines:
	
	 `get -e -i1.4,1.5,1.6 s.file`
	 `get -e -i1.4-1.6 s.file`
	
	 You can specify the SCCS Identification of a delta in any form shown in the **SID Specified** column of Table 1-4. The **get** command interprets partial SIDs as shown in the **SID Retrieved** column of the table.

-k	Suppresses replacement of identification keywords in the **g-file** by their value (see "Identification Keywords"). The **-k** flag is implied by the **-e** flag. If you accidentally ruin the g-file created by **get** with an **-e** flag, you can recreate it by reissuing the **get** command with the **-k** flag in place of the **-e** flag.
-l[p]	Writes a delta summary to an **l-file**. If you specify **-lp**, the delta summary is written to standard output, and the **get** command does not create the **l-file**. Use this flag to determine which deltas were used to create the **g-file** currently in use. See "SCCS Files" in topic 1.1.186.1 for the format of the **l-file**.
-m	Writes before each line of text in the **g-file** the SID of the delta that inserted the line into the SCCS file. The format is:

```
SID  tab  line of text
```

-n Writes the value of the %M% keyword before each line of text in the **g-file** (see "Identification Keywords" for information on keywords). The format is the value of %M%, followed by a horizontal tab, followed by the text line. When both the **-m** and **-n** flags are used, the format is:

```
%M% value    tab    SID    tab    line of text
```

-p Writes the text created from the SCCS file to standard output and does not create a **g-file**. The **get** command sends output normally sent to standard output to file descriptor 2 instead. If you specify the **-s** flag with the **-p** flag, output normally sent to standard output does not appear anywhere. Do not use the **-p** flag with the **-e** flag.

-rSID Specifies the SCCS identification string (SID) of the SCCS file version to be created. Table 1-4 shows what version of a file is created and the SID of the pending delta as functions of the SID specified.

-s Suppresses all output normally written to standard output. Error messages (written to standard error output) remain unaffected.

-t Accesses the most recently created delta in a given release or release and level. Without the **-r** flag, the **get** command accesses the most recent delta regardless of its SID.

-w string Substitutes **string** for the %W% keyword in **g-files** not intended for editing (see "SCCS Files" in topic 1.1.186.1 for information on **g-files**).

-x list Excludes a **list** of deltas in the creation of a file. See the **-i** flag for the SID list format in topic 1.1.186.2.

Examples

1. To get an SCCS file for editing:

   ```
   get  -e  s.prog.c
   ```

   This command creates a file named **prog.c** that only you have permission to modify. It also creates a **p.prog.c** file that prevents other users from using a **get -e** and editing a second **prog.c** file on the same SID before the delta is run. No one else can use **prog.c** or **s.prog.c** until you use the **delta** command to indicate that you are finished.

2. To get an SCCS file for reading:

   ```
   get  s.prog.c
   ```

   This command creates a file named **prog.c** that anyone can read, but that no one can modify. You can do this before searching files with the **grep** command or before compiling programs that are controlled with SCCS.

Related Information

See the following commands: "admin" in topic 1.1.16, "delta" in topic 1.1.117, "sccshelp" in topic 1.1.411, "prs" in topic 1.1.336, and "what" in topic 1.1.531.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.187 getopt

Purpose
Delimits command line flags and flag parameters.

Syntax

getopt --- **flag-string --- command-string** ---¦

Description

The **getopt** command prints the **command-string** to the standard output with all legal flags and flag parameters delimited by white-space. This allows a shell program to find flag settings in a command line string easily.

Legal flags are specified in the **flag-string**. A flag may be any single character except **?,-,** or an extended character. Using shell meta-characters such as **<,*, or |,** while allowable, is discouraged, because such characters must be escaped or quoted to prevent interpretation by the shell. Characters followed by **:** specify flags which are followed by a flag parameter.

The **getopt** command evaluates the **command-string** up to the first occurrence of **--**. If no **--** is present, this command processes the entire string. In either case, **getopt** appends a **--** to the output string.

When all flags have been processed (that is, up to the first nonflag argument), the **getopt** subroutine returns EOF. The special flag **--** (dash dash) can be used to delimit the end of the flags; EOF is returned, and **--** is skipped.

The **getopt** command writes an error message to standard error and exits with a non-zero status if it encounters a flag in the **command-string** not specified in the **flag-string**.

Example

The following shell procedure is a front end to the **ar** command. It uses **getopt** to separate the flags and parameters, then translates them into the **ar** command syntax and runs **ar** with these flags. Note how the output of **getopt** is used as the command line for **set** in order to reset the shell positional parameters ($1, $2, . . .), and how the return value from **getopt** is tested in order to detect illegal command lines.

```
  # @(#) lib: Front end to the ar command.
  #
  # Accepts the following flags:
  #    ar flags with "-" prefixes.  See the ar command.
  #    -L library   The default library is "libsubs.a".
  # Note: "lib -r -b sub1.o -v -l newsub.o" performs
  #       "ar rbvl sub1.o libsubs.a newsub.o"
  #       The ar command DOES interpret this correctly.

  set -- `getopt clsvmrua:b:i:dpqtxwL: $*`
  if [ $? != 0 ]          # Test for syntax error
  then
     exit 2
  fi
  FLAGS=  POSNAME=  LIBRARY=libsubs.a     # Default library name
  while [ $1 != -- ]
  do
     case $1 in
        -L)
           LIBRARY=$2
           shift; shift   # Shift past the -L and library name
           ;;
        -a|-b|-i)
           FLAGS=$FLAGS`expr "$1" : "-\(.\)"`
           POSNAME=$2
           shift; shift   # Shift past the flag and parameter
           ;;
        -*)               # Strip the "-" from the flag
           FLAGS=$FLAGS`expr "$1" : "-\(.\)"`
           shift
           ;;
     esac
  done

  shift                   # Shift past the "--" from getopt

  FLAGS=${FLAGS:-vt}      # Default if action not specified

  ar $FLAGS $POSNAME $LIBRARY $*
```

If this shell procedure is stored in a file named **lib**, all of the following commands are equivalent:

```
  lib -L mylib.a -v -r -b putfld.o getnam.o getfld.o getaddr.o
  lib -Lmylib.a -v -r -bputfld.o getnam.o getfld.o getaddr.o
  lib -Lmylib.a -vrbputfld.o getnam.o getfld.o getaddr.o
  lib -Lmylib.a -v -rbputfld.o -- getnam.o getfld.o getaddr.o
```

In each of these cases, the **getopt** command breaks down the command into:

```
  -L mylib.a -v -r -b putfld.o -- getnam.o getfld.o getaddr.o
```

The **getopt** command writes to its standard output. Because this command is enclosed in ` ` (grave accents), the shell takes its standard output and uses it to construct the command:

```
  set -- -L mylib.a -v -r -b putfld.o -- getnam.o
  getfld.o getaddr.o
```

This is called ***command substitution***. For more details on command substitution, see "Command Substitution" in topic 1.1.420.5.

The **set** command (page 1.1.420.23) sets the positional parameters **$1**, **$2**, **$3** and so forth, to each of the values **-L**, **mylib.a**, **-v** ..., respectively.

The shell procedure then uses the positional parameters to construct and run the command:

```
  ar vrb putfld.o mylib.a getnam.o getfld.o getaddr.o
```

The **ar** command (page 1.1.23) accepts the flags in any order. Therefore, you can specify flags to **lib** in any order, as long as a parameter immediately follows the **-a**, **-b**, **-i**, or **-L** flag, and all the flags come before any file names. This means that:

```
  lib -bputfld.o -rv -Lmylib.a getnam.o getfld.o getaddr.o
```

produces the command:

```
  ar brv putfld.o mylib.a getnam.o getfld.o getaddr.o
```

which performs the same action as each of the previous commands. See "test" in topic 1.1.469 and "expr" in topic 1.1.159 for more information about these commands.

Related Information

See the following commands: "sh, Rsh" in topic 1.1.420.

See the **getopt** subroutine in *AIX Operating System Technical Reference*.

1.1.188 gettext

Purpose
Extracts message, insert, and help descriptions.

Syntax

```
           +---------- -p ------------------------+
gettext ---¦           maximum three each         +-- outfile --¦
           ¦ +------+   +--------------+          ¦
           +-¦      +---¦ -h helpnum   +--- infile -+
             +- -p -+   ¦ -m mesgnum   ¦ ¦
                      ¦ ¦ -t insertnum ¦ ¦
                      ¦ +--------------+ ¦
                      +------------------+
```

Note: This command does not have MBCS support.

Description
The **gettext** command gets message, insert, or help descriptions from **infile** and places the descriptions in **outfile**. If you specify the **-p** flag or **gettext outfile**, the **gettext** command places a message/insert/help template in **outfile**. When you have your message, insert, or help descriptions or your message/insert/help template in **outfile**, you can edit **outfile**.

The **outfile** is an AIX ASCII file that consists of a header to identify the component and a group of message/insert/help descriptions. The contents of the message/insert/help descriptions includes a delimiter, control information and message/insert/help text. See *AIX Operating System Programming Tools and Interfaces* for a description of the **outfile** format and contents.

Flags

-h helpnum	Extracts help information from **infile**. You specify the index value used for the desired help number with **helpnum**.
-m mesgnum	Extracts message information from **infile**. You specify the index value used for the desired message number with **mesgnum**.
-p	Makes a message/insert/help template for **outfile**.
-t insertnum	Extracts text insert information from **infile**. You specify the index value used for the desired insert number with **insertnum**.

The syntax for the **mesgnum**, **insertnum**, and **helpnum** parameters is as follows:

num-num	Retrieves index numbers **num** to **num**.
num,num...	Retrieves a list of index numbers specified with **num**, **num**, **num**, and so on (maximum of 50 numbers).
num-	Retrieves index numbers equal to and larger than **num**.
-num	Retrieves index numbers from one to **num**.

Related Information

See the following command: "puttext" in topic 1.1.342.

1.1.189 getty

Purpose
Sets the characteristics of ports.

Syntax

```
       +--------------+
       ¦ +--- -d ---+ ¦
getty ---¦  one of  +---- portname --¦
       ¦  +----+  ¦
       +--¦ -r +--+
          ¦ -u ¦
          +----+
```

Description
The **init** process runs the **getty** command for each **portname** enabled for login. This command's primary function is to set the characteristics of the **port** specified by **portname**. Port characteristics include:

Bidirectional use **tty** line can be used in both directions)
Line speed (baud rate
Parit
Carriage return, tab, new-line, and form feed delay
Character set mapping, such as lowercase to uppercase, carriage retur to new-line translation, and tab expansion
Extended character suppor
Character erase and line erase editing character
Local or remote ech
Screen length for paging

The **getty** command obtains these settings by reading the port attributes specified in the **/etc/ports** configuration file and by observing the behavior of the port itself. (For details regarding the format of the **/etc/ports** file, see *AIX Operating System Technical Reference*. For the **logmodes** and **runmodes** parameter settings, see "stty, STTY" in topic 1.1.447.) When the **getty** command is invoked, it first opens the specified port. However, if carrier detection (modem control) is available on the port, the **getty** command cannot open the port until the carrier is present. Once the port is opened, the command sets the work station attributes according to the first **speed**, **logmodes**, **parity**, **erase**, **kill** and other parameters in the **ports** file and writes the herald message **herald** to the port. Then the **getty** command reads a login name from the port. If the login name contains extended characters, they are translated to the single ASCII characters most resembling them.

If a framing error occurs while reading, either because a user generates a **BREAK** signal from the work station or because the line speed is not the same as that of the transmitting work station, the port parameters are reset to the next combination specified in the **ports** file. Once the **getty** command reads a login name, it resets the work station modes according to the **runmodes** parameter, turns on carriage-return-to-new-line mapping if the login name was ended by a carriage return, turns on lowercase-to-uppercase mapping if the alphabetic characters in the login name were all uppercase, and executes the program specified by the **logger** parameter. That program, defaulting to the file **/bin/login**, runs in the same process as the **getty** command not as its child process.

Any additional arguments entered after the login name are passed to the

logger program. The **login** command interprets these as shell variable settings and places them in the environment.

On dial-in ports, it is often desirable to set no parity generation or checking as a default, but to permit the user to select parity as an option. For example, the following line in the **/etc/ports** file:

```
  parity = none,odd+inpck,even+inpck
```

accepts logins with any parity, but if a user generates **BREAK** before typing a login name, the **getty** command sets the port to generate odd parity and to check incoming characters for odd parity, while two **BREAK**s generate and check for even parity. Similarly, the line:

```
  speed=1200,300
```

works with 1200 baud, reverting to 300 baud when a **BREAK** is received before the login name. The default **runmodes** parameter, which must appear on one line in the **ports** file, is generally satisfactory. However, for work stations that have built-in tabs to every eight character positions and do not require tab delays, eliminating the **tab3** from the default in the **/etc/ports** file provides faster output with less system load.

Special Purpose Options

If there is a **timeout** keyword in the **ports** file, the **getty** command waits only the specified number of seconds for a response to the herald before advancing to the next port settings or, after all the settings are exhausted, exiting. If there is a **program** keyword for the port, instead of displaying the herald and gathering a login name, it executes the specified program immediately. This feature is a general mechanism for supporting special service ports such as network mail demons that need to be created when a connection is made from the external world. As a special case, if you specify:

```
  program = HOLD
```

the **runmodes**, **owner**, and **protection** parameters of the port are set and the **getty** command holds the port open indefinitely, thereby preventing the port modes from reverting to their open-default settings. This is useful, for example, in setting the modes on serial printer ports when it is inconvenient or impossible to have the programs that use them do so.

Flags

-d Uses standard input as the work station for which parameters are to be set according to those governing **portname**. Instead of executing a logger or a program, the **getty** command displays the name of the program that would have been run.

-u Makes the port available for shared (bidirectional) use. With this flag **getty** attempts to create a lock file in **/etc/locks** with the name of the device. This file can then be used by the **uucp** command to determine the status of the line. If the lock fails (because some other process is using the line), the **getty** command waits until the lock file is removed and exits. The **init** command creates a new **getty** to attempt the locking process again.

-r Makes the port available for shared (bidirectional) use. This is the same as the **-u** flag; however, the login herald is not displayed until at least one character is read from the port. This flag is used for direct lines or lines that have intelligent modems. This prevents **getty** from talking to a **getty** on the remote system or modem.

Example

To test a new **/etc/ports** entry:

```
getty -d /dev/tty5
```

This command tests a new port definition for **/dev/tty5** by simulating the login sequence of this device at your work station.

Files

/etc/locks	Directory containing lock files.
/etc/ports	System terminal port characteristics.
/bin/login	Program to log you in to system.
/bin/setmaps	Sets terminal maps.

Related Information

See the following commands: "login" in topic 1.1.241, "init, telinit" in topic 1.1.208, "pstart, penable, pshare, pdelay" in topic 1.1.338, and "stty, STTY" in topic 1.1.447.

See the **tty** and **ports** files in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.190 gprof

Purpose
Produces an execution profile of C, Pascal, or FORTRAN77 programs.

Syntax

```
        +------------+   +-- a.out ---+
gprof ---¦ +---------+ +---¦            +---¦
        +-¦ -a      +-+   +- gmon.out -+
          ¦ -       ¦¦
         ¦¦ -c      ¦¦
         ¦¦ -e name ¦¦
         ¦¦ -E name ¦¦
         ¦¦ -f name ¦¦
         ¦¦ -F name ¦¦
         ¦¦ -s      ¦¦
         ¦¦ -S      ¦¦
         ¦¦ -z      ¦¦
         ¦+---------+¦
         +-----------+
```

Description

The **gprof** command produces an execution profile of C, Pascal, or FORTRAN77 programs. The effect of called routines is incorporated in the profile of each caller. The profile data is taken from the call graph profile file (**gmon.out** default) which is created by programs which are compiled with the **-pg** option of **cc**, **pc**, and **f77**. That option also links in versions of the library routines which are compiled for profiling. The symbol table in the named object file (**a.out** default) is read and correlated with the call graph profile file. If more than one profile file is specified, the **gprof** output shows the sum of the profile information in the given profile files.

First, a flat profile is given, similar to that provided by **prof**. This listing gives the total execution times and call counts for each of the functions in the program, sorted by decreasing time.

Next, these times are propagated along the edges of the call graph. Cycles are discovered, and calls into a cycle are made to share the time of the cycle. A second listing shows the functions sorted according to the time they represent including the time of their call graph descendents. Below each function entry is shown its (direct) call graph children, and how their times are propagated to this function. A similar display above the function shows how this function's time and the time of its descendents is propagated to its (direct) call graph parents.

Cycles are also shown, with an entry for the cycle as a whole and a listing of the members of the cycle and their contributions to the time and call counts of the cycle.

Flags

-a Suppresses the printing of statically declared functions. If this option is given, all relevant information about the static function (for example, time samples, calls to other functions, calls from other functions) belongs to the function loaded just before the static function in the **a.out** file.

-b	Supresses the printing of a description of each field in the profile.
-c	The static call graph of the program is discovered by a heuristic which examines the text space of the object file. Static-only parents or children are indicated with call counts of 0.
-e name	Suppresses the printing of the graph profile entry for routine **name** and all its descendants (unless they have other ancestors that aren't suppressed). More than one **-e** option may be given. Only one **name** may be given with each **-e** option.
-E name	Suppresses the printing of the graph profile entry for routine **name** (and its descendants) as **-e**, above, and also excludes the time spent in **name** (and its descendants) from the total and percentage time computations. (For example, **-E mcount -E mcleanup** is the default.)
-f name	Prints the graph profile entry of only the specified routine **name** and its descendants. More than one **-f** option may be given. Only one **name** may be given with each **-f** option.
-F name	Prints the graph profile entry of only the routine **name** and its descendants (as **-f**, above) and also uses only the times of the printed routines in total time and percentage computations. More than one **-F** option may be given. Only one **name** may be given with each **-F** option. The **-F** option overrides the **-E** option.
-s	A profile file **gmon.sum** is produced which represents the sum of the profile information in all the specified profile files. This summary profile file may be given to subsequent executions of gprof (probably also with a **-s**) to accumulate profile data across several runs of an **a.out** file.
-S	Displays a summary of monitoring parameters and statistics on standard error.
-z	Displays routines which have zero usage (as indicated by call counts and accumulated time). This is useful in conjunction with the **-c** option for discovering which routines were never called.

Files

a.out	The namelist and text space.
gmon.out	Dynamic call graph and profile.
gmon.sum	Summarized dynamic call graph and profile.

Related Information
See the following commands: "cc" in topic 1.1.52 and "prof" in topic 1.1.332.

1.1.191 graph

Purpose
Draws a graph.

Syntax

```
       +-----------------------+   +-----------------------------+   +------
graph ---¦ +-------------------+ +---¦      +---- 1 -------------+ +---¦
       +-¦ -g grid  -s       +-+   +- -a -¦       +--- 0 ----+ +-+   +- -ccl
         ¦ -h space -t       ¦           +- num -¦          +-+
         ¦ -l label -u space ¦                  +- lowlim -+
         ¦ -m style -w space ¦
         ¦ -r space          ¦
         +-------------------+

   +------------------------------------------------------+
 ---¦      +------+   +----------------------------------+ +---
   +- -x -¦      +---¦        +---------------------+ +-+
         +- -l -+   +- lolim -¦         +---------+ +-+
                            +- uplim -¦         +-+
                                     +- space -+

   +------------------------------------------------------+
 ---¦      +------+   +----------------------------------+ +---¦
   +- -y -¦      +---¦        +---------------------+ +-+
         +- -l -+   +- lolim -¦         +---------+ +-+
                            +- uplim -¦         +-+
                                     +- space -+
```

Description
The **graph** command reads pairs of numbers from standard input, where each pair is the x and y coordinates of a point on a graph. The command processes the data that allows the successive points to be connected by straight lines when printed and then writes the graph to standard output. See "tplot" in topic 1.1.477 for information on how to code the output for printing.

In the input, non-numeric strings following the coordinates of a point are labels. Labels begin on the point. Labels can be surrounded with **"** (double quotation marks), in which case they can be empty or contain blanks and numbers. Labels cannot contain new-line characters.

The **graph** command stores all points internally and drops those for which there is not room. It also drops segments that run out of bounds. The **graph** command produces a legend indicating grid range with a grid unless you specify the **-s** flag. If a specified lower limit exceeds the upper limit, the **graph** command reverses the axis. The logarithmic axes cannot be reversed.

Flags

-a [num [lolim]]
Supplies abscissas missing from the input automatically. The **num** determines the spacing on the axis (the default is 1). The **lolim** determines the starting point for automatic abscissas (the default is 0 or the lower limit given by **-x[lolim]**.

-b	Breaks the graph after each label in the input.
-c char	Uses the character string **char** as the default label for each point.
-g grid	Uses **grid** as the grid style, where **grid** = 0 indicates no grid, **grid** = 1 indicates a frame with tick marks, and **grid** = 2 indicates a full grid (default).
-h space	Uses **space** as a fraction of space for height.
-l "label"	Uses **label** as a label for the graph.
-m style	Uses **style** as the style of connecting lines, where **style**=0 indicates disconnected lines, and **style**=1 indicates connected lines (default).
-r space	Uses **space** as the fraction of space to move to the right before plotting.
-s	Saves the current graphic screen image; does not erase before starting the plot.
-t	Transposes horizontal and vertical axes (**-x** now applies to the vertical axis).
-u space	Uses **space** as the fraction of space to move up before plotting.
-w space	Uses **space** as a fraction of space for width.

-x [l] [lolim [uplim [space]]]
Makes the x axis logarithmic if **l** is used. Use **lolim** as the lower x axis limit and **uplim** as the upper x axis limit. Use **space** for the grid spacing on the **x** axis. Normally these are determined automatically.

-y [l] [lolim [uplim [space]]]
Acts the same as **-x** for the y axis.

Related Information
See the following commands: "spline" in topic 1.1.437 and "tplot" in topic 1.1.477.

1.1.192 greek

Purpose
Converts output for a TELETYPE Model 37 work station to output for other work stations.

Syntax

```
        +---- -T$TERM -----+
greek ---¦                 +---¦
        +- -T workstation -+
```

Note: This command does not have MBCS support.

Description
The **greek** command reinterprets the TELETYPE Model 37 character set, including reverse and half-line motions, for display on other work stations. It simulates special characters, when possible, by overstriking. The **greek** command reads standard input and writes to standard output.

Flag

-Twork station Uses the specified **workstation**. If you omit the **-T** flag, the **greek** command attempts to use the work station specified in the environment variable **$TERM** (see the **environ** special facility in *AIX Operating System Technical Reference*). The **workstation** can be any one of the following:

300	DASI 300.
300-12	DASI 300 in 12-pitch.
300s	DASI 300s.
300s-12	DASI 300s in 12-pitch.
450	DASI 450.
450-12	DASI 450 in 12-pitch.
1620	Diablo 1620 (alias DASI 450).
1620-12	Diablo 1620 (alias DASI 450) in 12-pitch.
2621	Hewlett-Packard 2621, 2640, and 2645.
2640	Hewlett-Packard 2621, 2640, and 2645.
2645	Hewlett-Packard 2621, 2640, and 2645.
4014	Tektronix 4014.
hp	Hewlett-Packard 2621, 2640, and 2645.
tek	Tektronix 4014.

Files

/usr/bin/300
/usr/bin/300s
/usr/bin/4014
/usr/bin/450
/usr/bin/hp

Related Information
See the following commands: "300, 300s" in topic 1.1.559, "4014" in topic 1.1.560, "450" in topic 1.1.561, "eqn, neqn, checkeq" in topic 1.1.152, "hp" in topic 1.1.200, "mm, checkmm" in topic 1.1.274, "tplot" in topic 1.1.477, and "nroff, troff" in topic 1.1.301.

See the **greek** miscellaneous facility in *AIX Operating System Technical Reference*.

1.1.193 grep, egrep, fgrep

Purpose
Searches a file for a pattern.

Syntax

```
        +-------------------+   +------------+
grep ---¦    ¦ +----------+ +---¦   +----+   +---
        +- -p -¦          +-+   +---¦ -s +---+
               +- parsep -+         ¦ -v ¦ ¦
                                  ¦ ¦ -i ¦ ¦
                                  ¦ ¦ -w ¦ ¦
                                  ¦ ¦ -q ¦ ¦
                                  ¦ +----+ ¦
                                  +--------+

    +----------------+               +------------+
 ---¦     one of     +--- pattern ---¦            +---¦
    ¦     +----+     ¦               +--- file ---+
    ¦     ¦ -c ¦     ¦                          ¦
    ¦ +---¦ -l +---+ ¦                 +--------+
    +-¦   +----+   +-+
      ¦   +----+   ¦
      +---¦ -b +---+
          ¦ -n ¦ ¦
        ¦ +----+ ¦
        +--------+

         +--------+   +----------------+
egrep ---¦ +----+ +---¦     one of     +---
         +-¦ -v +-+   ¦     +----+     ¦
           ¦ -s ¦¦    ¦     ¦ -c ¦     ¦
          ¦¦ -h ¦¦    ¦ +---¦ -l +---+ ¦
          ¦+----+¦    +-¦   +----+   +-+
          +------+      ¦   +----+   ¦
                        +---¦ -b +---+
                            ¦ -n ¦ ¦
                          ¦ +----+ ¦
                          +--------+

    +---- pattern ----+   +------------+
 ---+-  -e pattern   -+---¦            +---¦
    +- -f stringfile -+   +--- file ---+
                                     ¦
                            +--------+

-----------------
¦ No space allowed between these arguments.


         +------------+   +----------------+
fgrep ---¦   +----+   +---¦     one of     +---
         +---¦ -v +---+   ¦     +----+     ¦
             ¦ -x ¦ ¦     ¦     ¦ -c ¦     ¦
           ¦ ¦ -i ¦ ¦     ¦ +---¦ -l +---+ ¦
           ¦ ¦ -s ¦ ¦     +-¦   +----+   ¦-+
           ¦ ¦ -h ¦ ¦       ¦   +----+   ¦
           ¦ +----+ ¦       +---¦ -b +---+
           +--------+           ¦ -n ¦ ¦
```

```
                                       ¦ +----+ ¦
                                       +--------+

        +---- pattern ----+   +------------+
     ---+-  -e pattern   -+---¦            +---¦
        +- -f stringfile -+   +--- file ---+
                                        ¦
                                  +--------+
```

Description

Commands of the **grep** family search input **file**s (standard input, by default) for lines matching a pattern. Normally, these commands copy each line found to standard output. Three versions of the **grep** command permit you to express the matching pattern in varying levels of complexity. The versions are:

grep Searches for **pattern**s, which are limited regular expressions in the style of the **ed** command. The **grep** command uses a compact non-deterministic algorithm.

egrep Searches for **pattern**s which are full regular expressions as in the **ed** command, except for **\(** and **\)** and with the addition of the following rules:

A regular expression followed by a + (plus sign) matches one or more occurrences of the regular expression.

A regular expression followed by a **?** (question mark) matches 0 or 1 occurrences of the regular expression.

Two regular expressions separated by a | (vertical bar) or by a new-line character match strings that are matched by either expression.

A regular expression may be enclosed in () (parentheses) for grouping.

The order of precedence of operators is [], then *** ? +**, then concatenation, then | and the new-line character.

The **egrep** command uses a deterministic algorithm that needs exponential space.

fgrep Searches for **pattern**s which are fixed strings. It searches for lines that contain one of the strings (lines are separated by new-line characters).

All versions of the **grep** command display the name of the file containing the matched line if you specify more than one **file** name. Characters with special meaning to the shell (**$ * [¦ ^ ( )** \), must be quoted when they appear in **pattern**s. When **pattern** is not simple string, you usually must enclose the entire **pattern** in single quotation marks. In an expression such as **[a-z]**, the minus means "through" according to the current collating sequence. A collating sequence may define equivalence classes for use in character ranges. See the "Introduction to International Character Support" in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

The exit value of these commands is:

0	A match was found.
1	No match was found.
2	A syntax error was found or a file was inaccessible (even if matches were found).

Notes:

1. The maximum line length is 512 characters; longer lines are broken into multiple lines of 512 or fewer characters (the **grep** command only).

2. Paragraphs (under the **-p** flag) are limited to a maximum length of 5000 characters. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the character limits can be reduced by as much as 50%, depending on the character code set being used.

3. Running the **grep** command on a special file produces unpredictable results and is discouraged.

Flags

-b	Precedes each line by the block number on which it was found. Use this flag to help find disk block numbers by context.
-c	Displays only a count of matching lines.
-e pattern	Specifies a **pattern**. This works the same as a simple **pattern** but is useful when the **pattern** begins with a - (minus).
-f stringfile	Specifies a file that contains patterns (**egrep**) or strings (the **fgrep** command).
-h	When multiple files are being processed, suppresses file names (**fgrep** and **egrep** only).
-i	Makes no distinction between uppercase and lowercase characters when making comparisons (the **grep** and **fgrep** commands only).
-l	Lists just the names of files (once) with matching lines. Each file name is separated by a new-line character.
-n	Precedes each line with its relative line number in the file.
-pparsep	Displays the entire paragraph containing matched lines. Paragraphs are delimited by paragraph separators, **parsep**, which are patterns in the same form as the search pattern. Lines containing the paragraph separators are used only as separators; they are never included in the output. If used, the **parsep** pattern must follow the **-p** option without a space. If **parsep** is not used, the default is a blank line.

-q	Suppresses error messages about inaccessible files (**grep** only).
-s	Silent mode. Nothing is printed except error messages. This is useful for checking status. Displays all lines except those that match the specified pattern.
-v	Displays all lines except those that match the specified pattern. Returns exit code if lines were found that did not match the pattern.
-w	The pattern is searched for as a word (as if surrounded by '\<'pattern'\>'). This flag can be used only with the **grep** command.
-x	Displays lines that match the pattern exactly with no additional characters (the **fgrep** command only).

Examples

1. To search several files for a simple string of characters:

   ```
   fgrep  "strcpy"  *.c
   ```

   This command searches for the string **strcpy** in all files in the current directory with names ending in **.c**.

2. To count the number of lines that match a pattern:

   ```
   fgrep  -c  "{"  pgm.c
   fgrep  -c  "}"  pgm.c
   ```

   This command displays the number of lines in **pgm.c** that contain open and close braces.

   If you do not put more than one **{** or **}** on a line in your C programs, and if the braces are properly balanced, the two numbers displayed are the same. If the numbers are not the same, you can display the lines that contain braces in the order that they occur in the file with:

   ```
   egrep  "{|}"  pgm.c
   ```

3. To use a pattern that contains some of the pattern-matching characters *****, ^, **?**, [,], \(, \), \\**{**, and \**}**:

   ```
   grep  "^[a-zA-Z]"  pgm.s
   ```

   This command displays all lines in **pgm.s** that begin with a letter.

   Because the **fgrep** command does not interpret pattern-matching characters, the following command makes **fgrep** search only for the string **^[a-zA-Z]** in **pgm.s**.

   ```
   fgrep  "^[a-zA-Z]"  pgm.s
   ```

4. To use an extended pattern that contains some of the pattern-matching characters +, **?**, |, (, and):

   ```
   egrep  "\(  *([a-zA-Z]*¦[0-9]*)  *\)"  my.txt
   ```

This command displays lines that contain letters in parentheses or digits in parentheses, but not parenthesized letter-digit combinations. It matches **(y)** and **(783902)**, but not **(alpha19c)**.

Note: When using the **egrep** command, \(and \) match parentheses in the text, but (and **)** are special characters that group parts of the pattern. The reverse is true for **grep**.

5. To display all lines that do **not** match a pattern:

```
grep  -v  "^#" pgm.c
```

   This command displays all lines in pgm.c that do not begin with a # character.

6. To display the names of files that contain a pattern:

```
fgrep  -l  "strcpy"  *.c
```

   This command searches the files in the current directory that end with **.c** and displays the names of those files that contain the string **strcpy**.

7. To display all lines containing **$** using **grep**:

```
grep  \\$  file.foo
```

   This command demonstrates how to quote a special character to find the literal character, the **$** in this example.

Related Information
See the following commands: "ed, red" in topic 1.1.147, "sed" in topic 1.1.415, and "sh, Rsh" in topic 1.1.420.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.194 groups

Purpose
Displays your group membership.

Syntax

```
          +--------+
groups ---¦        ¦+---¦
          +- user -+
```

```
-----------------
¦ The default user is the person running the command.
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **groups** command writes to standard output the groups to which you or the specified **user** belong. The AIX Operating System allows you to belong to many different groups at the same time.

Your primary group is specified in the file **/etc/passwd**. Once you are logged in, you can change your active group with the **newgrp** command (see page 1.1.290). When you create a file, its group ID is that of your active group.

Other groups that you belong to are specified in the file **/etc/group**. If you belong to more than one group, you can access files belonging to any of those groups without changing your primary group ID. These are called your **concurrent groups**.

Files

/etc/group	Group file; contains group IDs.
/etc/passwd	Password file; contains user IDs.

Related Information
See the following command: "newgrp" in topic 1.1.290.

See the **setgroups** system call and the **initgroups** subroutine in *AIX Operating System Technical Reference*.

1.1.195 halt

Purpose
Stops the processor.

Syntax

```
        +--------+
halt ---¦ +----+ +---¦
        +-¦ -l +-+
          ¦ -n ¦
         ¦¦ -q ¦¦
         ¦¦ -Y ¦¦
         ¦+----+¦
         +-------+
```

Description
The **halt** command syncs data to the disks and then stops the processor. The machine does not reboot.

When the **halt** command is finished, you will see one of the following messages. When you do, it is safe to turn the machine off.

PS/2:	System halted, you may turn the power off now Type **Enter** to reboot :
AIX/370:	916-790 Loading a wait PSW (Use CP to re-ipl)

The **halt** command normally logs the shutdown using **syslog** and places a shutdown record in the login accounting file **/usr/adm/wtmp**. These actions are inhibited if the **-n** or **-q** options are used.

Flags

-l	Do not log halt. This flag is implied by **-n** and **-q**.
-n	Prevents the sync before stopping.
-q	Causes a quick halt, no graceful shutdown is attempted.
-y	Halts the system for a dial-up.

Related Information

See the following commands: "sync" in topic 1.1.453, "syslogd" in topic 1.1.457 and "fastboot, fasthalt" in topic 1.1.162.

1.1.196 hangman

Purpose

Plays hangman, the word-guessing game.

Syntax

```
                      +--------+
/usr/games/hangman ---¦        +---¦
                      +- file -+
```

Note: This command does not have MBCS support.

Description

The **hangman** game chooses a word of at least seven letters from a standard dictionary. You try to guess the word by guessing the letters in it, one at a time. You are allowed seven mistakes. The **file** parameter specifies an alternate dictionary.

To quit the game, press the INTERRUPT key or END-OF-FILE key (**Ctrl-D**).

1.1.197 head

Purpose
Prints the first few lines of a file or files.

Syntax

```
        +----------+
head ---¦          +--- file ---¦
        +- -count -+
```

Description
The **head** command prints the first **count** lines of each of the specified files, or of the standard output. If **-count** is omitted, it defaults to 10.

Related Information

See the following command: "tail" in topic 1.1.460.

1.1.198 help

Purpose

Provides information for the new user.

Syntax

```
help ----¦
```

Description

The **help** command presents a one-page display of information for the new user. A sample of the display appears below.

```
  Look in a printed manual for general help if you can.  To get started,
  refer to Using the AIX Operating System Manual.

  The commands:
      man -k keyword        lists commands relevant to a keyword
      man command           prints out the manual pages for a command
  are helpful; other basic commands are:
      cat        -concatenates files (and just prints them out)
      ex         -text editor
      finger     -user information lookup program
      ls         -lists contents of a directory
      mail       -send and receive mail
      passwd     -change login password
      sccshelp   -view information on the Source Code Control System
      tset       -sets terminal modes
      who        -who is on the system
      write      -write to another user
  You could find programs about mail by the command:   man -k mail
  And print out the man command documentation via:     man mail
  You can log out by typing a control-d (if your prompt is $)
  or by typing logout (if your prompt is %).
```

Related Information

See the following command: "sccshelp" in topic 1.1.411.

1.1.199 hftinit, hftsmproc

Purpose
Initializes the default keyboard map and display model for the **hft** device driver and enables the "switching" between virtual terminals. These commands should not be entered on the command line. They are invoked automatically during installation.

Syntax

```
hftinit ---¦

              +----------+
hftsmproc ---¦           +---¦
              +- device -+
```

Note: These commands are for the PS/2 only.

Description

The **hftsmproc** program performs the context switching and controls necessary to switch between virtual terminals when the **Alt-Action**, **Shift-Action** and **Ctrl-Action** keys are pressed. Action is normally the right-hand **Ctrl** key.

The **hftinit** program loads the default keyboard map into the **hft** driver when the system goes from boot to single user mode. It uses the **lang** choice in the file **/etc/ddi/hft.ddi** to determine which keyboard map to use. This choice is made either at AIX installation time or by the **devices** program. The **lang** choice is used to determine which of the files in the **/etc/kbdtbls** should be opened to read the keyboard map. If the **lang** choice is U.S. English, no file is read because U.S. English is the driver default and is built into the driver code.

The program also determines the model of the physical display attached to the system and passes its code to the **hft** driver. It uses the **dspVGA** and **dsp8514A** choices in the file **/etc/ddi/hft.ddi** to make this determination.

The **hftinit** program runs only once, when you boot the system up. If for any reason the **hftinit** program needs to be run again, the system must be rebooted.

In most cases the **hftinit** program runs successfully even though some of the required input files are missing or corrupted. In such cases, it uses a default setting and displays an explanatory message.

Files
/etc/ddi/hft.ddi
/etc/kbdtbls

Related Information
See: Appendix A, "AIX Device Table."

See the **hft** section in *AIX Operating System Technical Reference*.

1.1.200 hp

Purpose
Handles special functions for the HP2640- and HP2621-series terminals.

Syntax

```
      +--------+
hp ---¦ +----+ +---¦
      +-¦ -e +-+
        ¦ -  ¦¦
       ¦+----+¦
       +------+
```

Note: This command does not have MBCS support.

Description
The **hp** command reads standard input (usually output from the **nroff**), and writes to standard output, which is usually Hewlett-Packard 2640- and 2621-series terminal displays.

If your terminal has the display enhancement feature, you can display subscripts and superscripts. With the mathematical symbol feature, you can display Greek and other special characters, with two exceptions: the **hp** command approximates the logical operator NOT with a right arrow and it only shows the top half of the integral sign.

For overstrike characters (characters followed by a backspace and another character), if either character is an underscore character, the other appears underlined or in inverse video depending on terminal enhancements.

Note: Some sequences of control characters (reverse line-feeds and backspaces) can make text disappear from the display. Tables with vertical lines generated by the **tbl** command are often missing lines of text containing the bottom of a vertical line. You can avoid these problems by first piping the input through the **col**, and then piping through the **hp** command.

Flags

-e Displays overstruck characters underlined, superscripts in half-bright, and subscripts half-bright underlined. Otherwise, all overstruck characters, subscripts, and superscripts appear in inverse video (dark-on-light). Use this flag only if your display has the display enhancements feature.

-m Produces only one blank line for any number of successive blank lines in the text.

Related Information
See the following commands: "col" in topic 1.1.77, "eqn, neqn, checkeq" in topic 1.1.152, "nroff, troff" in topic 1.1.301, and "tbl" in topic 1.1.463.

1.1.201 hyphen

Purpose
Finds hyphenated words.

Syntax

```
          +--------+
hyphen ---¦        +---¦
          +- file -+ ¦
        +------------+
```

Note: This command does not have MBCS support.

Description
The **hyphen** command reads the input **file**s (standard input, by default), finds all the lines ending with hyphenated words, and writes those words to standard output. A word is considered hyphenated only if the hyphen occurs at the end of a line. The **hyphen** command reads standard input if you do not specify any **file** names on the command line.

Note: The **hyphen** command cannot handle hyphenated italicized words. It also sometimes gives unnecessary output.

Examples

1. To check the way words are hyphenated in a text file:

   ```
   hyphen  chap1
   ```

   This command lists the words in **chap1** that are hyphenated at the end of a line.

2. To check the hyphenation performed by a text formatting program:

   ```
   mm  chap1  |  hyphen
   ```

   This command lists the words that the **nroff** command decides to hyphenate across lines.

Related Information
See the following commands: "mm, checkmm" in topic 1.1.274 and "nroff, troff" in topic 1.1.301.

1.1.202 icaload

Purpose

Downloads tasks to an installed RIC adapter.

Syntax

```
           +------+
icaload ---¦      +--- ric-number --- file --- task-number ---
           +- -v -+

    +---------------+
 ---¦               +---¦
    +- load-option -+
```

Note: This command is for the PS/2 only.

Description

The **icaload** command downloads tasks into RIC card memory. The task image is loaded from the **file** argument and must be either a DOS .COM or .EXE format file. If **file** is a .COM file, its name must end with **.com**. If **file** is a .EXE file, the name can end in anything but **.com**. The **ric-number** specifies which physical RIC co-processor is to be downloaded. Physical card 0 is the card in the lowest numbered slot in the PS/2. The task is loaded as **task-number** on the RIC card.

By default, the task is started after being loaded into the RIC card. However, you can load a task without starting it by setting **load-option** to either **1** or **L**.

If the **-v** option is given, **icaload** prints a message indicating the starting address of the task in RIC memory.

Examples

1. Load RIC 0 with the RCM. The RCM must be loaded as a task 0 and is called **icaaim.com**.

   ```
   icaload 0 /usr/lib/cyclone/icaaim.com 0
   ```

2. Load RIC 0 with the RS232 support program as task 1. Do not start the task running.

   ```
   icaload 0 /usr/lib/cyclone/com232.exe 1 1
   ```

Files

/dev/ric*

Related Information

See "icareset" in topic 1.1.203.

1.1.203 icareset

Purpose
Asserts a hardware reset on an installed RIC adapter.

Syntax

icareset --- **ric-number** ---¦

Description

Resets hardware on an installed RIC adapter.

Example

To reset RIC 0:

```
  icareset 0
```

Files

/dev/ric*

Related Information

See "icaload" in topic 1.1.202.

1.1.204 iconv

Purpose
Converts data encoded in one file or network code into another code.

Syntax

```
                                     +-----------------------------------------+
iconv --- -f code --- -t code ---¦                                          +-¦
                                     +- -d string -- -o filename -- filename -+
```

Description
The **iconv** command reads input from a named file or from standard input if no input file is specified. The command converts the input from one code set into another code set, directing the output to a specified file or to standard output if no output file is specified.

Each of the supported code set names listed with the **-f** and **-t** flags below indicate specific code sets, except for the code set named **ebcdic**. This code set name really specifies one of several different ebcdic code sets, as determined by the setting of your current language and locale.

Flags

-d string	Specifies a default string. If **iconv** does not support a particular file or network code conversion, it puts the value of **string** in your output file or standard output in place of unrepresentable characters. If you do not specify **string**, **iconv** uses the default value.
-f code	Specifies the file or network code of the input file. The code can be **pc850**, **pc932**, **8859-1**, **ujis**, or **ebcdic**.
-t code	Specifies the file or network code of the output file. The code can be **pc850**, **pc932**, **8859-1**, **ujis** or **ebcdic**.
filename	Designates the input file. If no file is specified, input is read from standard input.
-o filename	Designates the output file. If no file is specified, output is directed to standard output.

Related Information

See the "Characters" chapter in the *AIX MBCS Guide*.

1.1.205 id

Purpose
Displays the system identity of the user issuing the command.

Syntax

```
id ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **id** command writes a message to standard output containing the user and group IDs and corresponding names of the invoking process. When effective and real names and IDs do not match, the **id** command writes only the effective name and ID.

Related Information
See the following command: "logname" in topic 1.1.242.

See the **getuid** subroutine in *AIX Operating System Technical Reference*.

1.1.206 inc

Purpose
Incorporates new mail.

Syntax

```
       +-- +inbox --+   +------ -noaudit ------+   +--- -changecur ---+
inc ---¦            +---¦        one of        +---¦      one of      +---
       +- + folder -+   ¦ +------------------+ ¦   ¦ +--------------+ ¦
                        +-¦ -audit file      +-+   +-¦ -changecur   +-+
                          ¦ -noaudit         ¦       ¦ -nochangecur ¦
                          +------------------+       +--------------+

    +----------------------+   +--------------+   +-----------------+
 ---¦        one of        +---¦              +---¦     one of      +---
    ¦ +------------------+ ¦   +- -file name -+   ¦ +-------------+ ¦
    +-¦ -form file       +-+                      +-¦ -truncate   +-+
      ¦ -format string   ¦                          ¦ -notruncate ¦
      +------------------+                          +-------------+

    +-- -nosilent --+   +--- -width 72 ----+
 ---¦    one of     +---¦                  +---¦
    ¦ +-----------+ ¦   +- -width columns -+
    +-¦ -silent   +-+
      ¦ -nosilent ¦
      +-----------+

inc -- -help --¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **inc** command is used to incorporate incoming mail. This command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **inc** command takes all of the new messages from your mail drop and places them in the specified folder. If the specified folder does not exist, the **inc** command asks you if it should be created. The command assigns the new messages consecutive message numbers starting with the next highest number in the folder. It also assigns the new messages the protection code specified in the **Msg-Protect:** entry in your **.mh_profile**. If there is no **Msg-Protect:** entry, the **inc** command assigns the messages the protection code of 644. The **inc** command calls the **scan** command to display information about the new messages. If the **Unseen-Sequence:** profile entry specifies any sequences, the **inc** command adds the new messages to each sequence.

Flags

-audit *file*	Copies the current date to the specified file and appends the output from the **scan** command to the file.
-changecur	Sets the first new message as the current message for the specified folder. This flag is the default.

-file *file*	Incorporates messages from the specified file instead of the user's maildrop.
+*folder*	Incorporates the new messages into the specified folder. The default folder is **+inbox**.
-form *file*	Displays the **scan** command output in the alternate format described in *file*.
-format *string*	Displays the **scan** command output in the alternate format described by *string*.
-help	Displays help information for the command.
-noaudit	Does not record information about incorporating the new messages (see the **-audit** flag). This flag is the default.
-nochangecur	Does not alter the setting for the current message in the specified folder.
-nosilent	Prompts the user for any necessary information. This flag is the default.
-notruncate	Does not clear the mailbox or file from which the **inc** command is taking new messages. If the **-file** flag is specified, the **-notruncate** flag is the default.
-silent	Does not prompt you for any information. This flag is useful for running the **inc** command in the background.
-truncate	Clears the mailbox or file from which the **inc** command is taking new messages. If the **-file** flag is not specified, the **-truncate** flag is the default.
-width *num*	Sets the number of columns in the **scan** command output. The default is the width of the display.

Profile Entries

Alternate-Mailboxes:	Specifies your mailboxes.
Folder-Protect:	Sets the protection level for your new folder directories.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your **user_mh_directory**.
Unseen-Sequence:	Specifies the sequences used to keep track of your unseen messages.

Files

$HOME/.mh_profile	The MH user profile.
/usr/lib/mh/mtstailor	The MH tailor file.
$HOME/.newmail	The location of the mail drop.

Related Information
See other MH commands: "mhmail" in topic 1.1.264, "post" in topic 1.1.320, and "scan" in topic 1.1.409.

See the **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System*

Technical Reference. (The **mh-format** section describes formats for the **-form** and **-format** options.)

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.207 indent

Purpose
A C program formatter.

Syntax

```
         +-------------------------------+   +--------------+
indent ---¦             +---------------+ +---¦              +---¦
         +- input-file -¦               +-+   +--- option ---+
                        +- output-file -+                   ¦
                                                 +----------+
```

Description

The **indent** command is a **C** program formatter. It reformats the **C** program in the **input-file** according to the switches. The switches which can be specified are described below. They may appear before or after the file names.

Note: If you only specify an **input-file**, the formatting is done "in-place." That is, the formatted file is written back into **input-file** and a backup copy of **input-file** is written in the current directory, with a file name suffix of **.BAK**.

If **output-file** is specified, **indent** checks to make sure it is different from **input-file**.

You may set up your own profile of defaults for the **indent** command by creating a file called **.indent.pro** in your login directory or the current directory. Include in it whatever switches you like. The switches should be separated by spaces, tabs or newlines. Switches in **.indent.pro** in the current directory override those in your login directory (with the exception of **-T** type definitions, which just accumulate). If the **indent** command is run and a profile file exists, then it is read to set up the program's defaults. Switches on the command line, however, override profile switches.

Comments

"Box" comments. The **indent** command assumes that any comment with a dash or star immediately after the start of comment (that is, '/*-' or '/**') is a comment surrounded by a box of stars. Each line of such a comment is left unchanged, except that its indentation may be adjusted to account for the change in indentation of the first line of the comment.

Straight text. All other comments are treated as straight text. The **indent** command fits as many words (separated by blanks, tabs, or newlines) on a line as possible. Blank lines break paragraphs.

Comment indentation.

If a comment is on a line with code it is started in the comment column set by **-cn**. Otherwise, the comment is started at **n** indentation levels less than where code is currently being placed, where **n** is specified by **-dn**. If the code on a line extends past the comment column, the comment starts further to the right. The right margin may be extended automatically in extreme cases.

Preprocessor lines.

In general, the **indent** command leaves preprocessor lines alone. The only reformatting it does is to straighten up trailing comments. It leaves embedded comments alone. Conditional compilation (**#ifdef...#endif**) is recognized and **indent** attempts to compensate correctly for the syntactic peculiarities introduced.

C syntax.

The **indent** command understands a substantial amount about C syntax, but has a forgiving parser. It attempts to cope with the usual sorts of incomplete and misformed syntax. In particular, the use of macros like:

```
#define forever for(;;)
```

is handled properly.

Flags

The options listed below control the formatting style imposed by the **indent** command.

-bad, -nbad Forces or suppresses a blank line after every block of declarations. Default: **-nbad**

-bap, -nbap Forces or suppresses a blank line after every procedure body. Default: **-nbap.**

-bbb, -nbbb Forces or suppresses a blank line before every comment block. Default: **-nbbb**

Note: The third character in a block comment must be a blank. The third character in a box comment must be either a dash (-) or an asterisk (*).

-bc, -nbc Forces or suppresses a newline after each comma in a declaration. Default: **-nbc**

-br, -bl **-br** formats compound statements thus:

```
if (...) {
    code
}
```

-bl formats compound statements thus:

```
if (...)
{
    code
}
```

Default: **-br**

-cn Sets the tab position where comments on code start. Default: **33**

-cdn Sets tab position where comments on declarations start. Default: Uses **-c** value

-cdb, -ncdb	Enables or disables placing comment delimiters on blank lines. With **-cdb**, comments look like this:
	`/*`
	` * this is a comment`
	` */`
	With **-ncdb**, comments look like this:
	`/* this is a comment */`
	This only affects block comments, not comments to the right of code. Default: **-cdb**

-ce, -nce	Enables or disables forcing else's to follow the immediately preceding '}.' Default: **-ce**
-cin	Sets the continuation indent to **n**. Continuation lines will be indented **n** positions from the beginning of the first line of the statement. Expressions in parentheses have extra indentation added to indicate the nesting, unless **-lp** is in effect. Default: Uses **-i** value
-clin	Causes case labels to be indented **n** positions to the right of the containing **switch** statement. **-cli0.5** causes case labels to be indented half a tab stop. (This is the only option that takes a fractional argument.) Default: **-cli0**
-dn	Controls the placement of comments which are not to the right of code. Specifying **-d1** causes such comments to appear one indentation level to the left of code. Default: **-d0** (comments are aligned with code) lines up these comments with the code.
-din	Specifies the number of positions to indent an identifier from a preceding declaration keyword. Default: **-di16**
-dj, -ndj	Left-justifies or indents declarations. Default: **-ndj**
-ei, -nei	Enables or disables special **else-if** processing. **-ei** causes **if**s following **else**s to have the same indentation as the preceding **if** statement. Default: **-ei**
-fc1, nfc1	Enables or disables formatting comments that start in column 1. Often, comments whose leading '/' is in column 1 have been carefully hand-formatted by the programmer. In such cases, use **-nfc1**. Default: **-fc1**
-in	Sets the indentation level size. Default: **8 positions**
-ip, -nip	Enables or disables indenting parameter declarations. Default: **-ip**
-1cn	Sets the maximum length of the comment block to **n**. Default value is specified by **1c** flag or 78.
-ln	Sets the maximum length of an output line to **n**. Default: **78**

-lp, -nlp	Aligns or leaves unaligned code surrounded by parentheses in continuation lines. If a line has a left paren with no matching right paren on that line, continuation lines will start at the position following the left paren. With **-nlp** in effect, such lines appear thus:

```
  p1 = first_procedure(second_procedure(p2, p3),
      third_procedure(p4, p5));
```

With **-lp** in effect, such lines appear thus:

```
  p1 = first_procedure(second_procedure(p2, p3),
                       third_procedure(p4, p5));
```

Inserting two more newlines yields the following:

```
  p1 = first_procedure(second_procedure(p2,
                                        p3),
                       third_procedure(p4,
                                       p5));
```

Default: **-lp**

-npro	Causes the profile files **./.indent.pro** and **~/.indent.pro** to be ignored.
-pcs, -npcs	Inserts or suppresses a space between each procedure call name and the following '('. Default: **-npcs**
-ps, -nps	Inserts or suppresses spaces on both sides of the pointer following operator '->'. Default: **-nps**
-psl, -npsl	**-psl** causes the names of procedures being defined to appear left-justified. Their types, if any, remain on the previous lines. Default: **-psl**
-sc, -nsc	Enables or disables the placement of asterisks (*) to the left of all comments. Default: **-sc**
-sob, -nsob	Removes or retains optional blank lines. Use to remove blank lines after declarations. Default: **-nsob**
-st	Causes the **indent** command to take its input from stdin and put its output to stdout.
-T typename	Adds **typename** to the list of type keywords. Names accumulate; **-T** can be specified more than once. Specify all the typenames appearing in your program that are defined by **typedef**s. No harm is done if you miss a few, but the program won't be formatted as nicely. This may be a painful thing to do, but it's a symptom of a problem in C. **typedef** causes a syntactic change in the language and the **indent** command can't find all **typedef**s.
-troff	Causes the **indent** command to format the program for processing by troff. It will produce a fancy listing in much the same spirit as **vgrind**. If no output file is specified, the default is standard output, rather than

formatting in place.

-v, -nv	Turns on or off "verbose mode," which reports when **indent** splits one line of input into two or more lines of output, and gives some size statistics at completion. Default: **-nv**

Files

./.indent.pro	Profile file.
~/.indent.pro	Profile file.

1.1.208 init, telinit

Purpose
Initializes the system.

Syntax

```
/etc/init -------------------¦
            +- runlevel -¦
            +----  q ----¦
            +----  s ----¦
            +----  Q ----¦
            +----  S ----¦
            +---- -m ----¦
            +---- -M ----+

/etc/telinit -------------------¦
               +- runlevel -¦
               +---- a -----¦
               +---- b -----¦
               +---- c -----¦
               +---- q -----¦
               +---- s -----¦
               +---- Q -----¦
               +---- S -----¦
               +---- m -----¦
               +---- M -----+
```

Note: This command does not have MBCS support.

Description

The **init** command is a general process spawner. Its primary role is to create processes from stanzas stored in the file **/etc/inittab**. This file usually has the **init** command spawn **getty**s on each line so that a user can log in. The **init** command also controls autonomous processes required by a particular system.

The **telinit** command, which is linked to the **/etc/init** file, is used to direct the actions of the init command. For more information on **telinit**, see page 1.1.208.

The **init** command considers the system to be in a certain runlevel at any given time. A runlevel can be viewed as a software configuration of the system where each configuration allows only a selected group of processes to exist. The processes spawned by the **init** command for each of these runlevels is defined in the file **inittab**. The **init** command can be in a runlevel, from **0-6** and **M**, **m**, **S**, or **s**. The runlevel is changed by having the superuser run **/etc/telinit**. This user-spawned **init** command sends appropriate signals to the **init** daemon command spawned by the operating system when the system was rebooted, telling it to which runlevel to change.

The runlevels in the default **/etc/inittab** file are as follows:

0 Halt the system. This runlevel should be used to bring down the system when it is going to be powered off. This level unmounts the file systems, leaves the cluster, kills all processes, and then halts the system.

1 Allow access to the system without any other activity. When entered from the single-user level, runlevel 1 starts the processes that allow users to use their terminals (see "getty" in topic 1.1.189). If runlevel 1 is entered from any other state, no other processes are terminated.

2 Enter multi-user mode. This runlevel is used to bring up the system so that everyone can use it. During this transition, daemons are started, the cluster is joined, and file systems are checked and mounted. When successfully concluded, this state now enters runlevel 4 (via **init 4**).

3 Prepare the system for maintenance. This runlevel is used to quiesce the system. The file systems are unmounted. All processes are killed and the system enters single-user level (via **init s**).

4 Enable normal operations; allows terminal activity to begin.

Note: Runlevels 5 and 6 are not used in the default **/etc/inittab** file.

Normally you should use only runlevels 0, 2, and 3. Runlevels 0 and 3 are used to bring down the system for maintenance activity. Runlevel 2 is used to bring up the system for normal usage.

The **init** command is invoked inside the system as the last step in the boot procedure. The first thing the **init** command does is to look for the file **/etc/inittab** and see if it contains a stanza with an action of **initdefault**. If **initdefault** exists, the **init** command uses the **level** specified in that stanza as the initial runlevel. If **initdefault** or **inittab** do not exist, **init** requests that you enter a runlevel from the system virtual console, **/dev/syscon**. If an **M**, **m**, **S** or **s** is entered the **init** command enters **Single-User** level. This is the only runlevel that does not require the existence of a properly formatted **inittab** file. If the file **/etc/inittab** does not exist, the only runlevel that the **init** command can enter is the **Single-User** level. In **Single User** level, the system virtual console **/dev/syscon** is opened for reading and writing, and the command **/etc/sushell** is invoked. To exit **Single-User** level, one of two options can be used.

If the shell is terminated, the **init** command prompts for a new runlevel.

The **init** or **telinit** command can signal the **init** command and force a new runlevel.

When attempting to boot the system, failure of the **init** command to prompt for a runlevel may be because the device **/dev/syscon** is linked to a device other than the physical system unit (**/dev/systty**). If this occurs, you must boot the installation/maintenance system, mount the system's **/local** file system, and issue the command:

```
ln /local/dev/systty /local/dev/syscon
```

When the **init** command prompts for a new runlevel, you can enter **0-6**, **M**, **m**, **S** or **s**. If you enter **M**, **m**, **s**, or **S**, the **init** command operates as described in **Single-User** level with the additional result that the device **/dev/syscon** is linked to the user's terminal line, thus making it the virtual system console. A message is generated on the physical console,

/dev/systty, identifying where the virtual terminal is located.

When the **init** command is started initially and whenever it switches out of **Single-User** level to the normal run state, it sets the **ioctl** system-call states of the virtual console **/dev/syscon** to the modes saved in the file **/etc/ioctl.console**. This file is written by the **init** command whenever **Single-User** level is entered. If this file does not exist when the **init** command reads it, a message prints and the default settings are assumed.

If **0-6** is entered, the **init** command enters the corresponding runlevel. Any other input is rejected and the user is prompted for a runlevel. If this is the first time the **init** command has entered a runlevel other than the **Single-User** level, the **init** command scans the **inittab** file for stanzas with **action** values of **boot** or **bootwait**. These stanzas are processed, providing their level matches **runlevel**, before any normal processing of the **inittab** file takes place. In this way, any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. The **inittab** file is then scanned to find entries that are to be processed for that runlevel.

In the multi-user environment, the **inittab** file is usually set up so that the **init** command creates a process for each terminal on the system. The **init** command controls the number of login processes that are active at one time.

For terminal processes, ultimately the shell terminates because of the end-of-file, either typed explicitly or generated as the result of hanging up. When **init** receives a signal telling it that the process that it spawned has died, **init** records this along with the reason in **/etc/utmp** and **/usr/adm/wtmp**. A history of the processes spawned is kept in **/usr/adm/wtmp** if the file exists.

To spawn each process in the **inittab** file, the **init** command reads each entry stanza and for each stanza which should be respawned, it forks a child process. After it has spawned all processes specified in the file **inittab**, the **init** command waits for one of its descendant processes to die, a **powerfail** signal, or until it is signaled by another **init** command or **telinit** command to change the system runlevel. When one of these three conditions occurs, **init** re-examines the **inittab** file. New stanzas can be added to **inittab** at any time. However, **init** still waits for one of the above three conditions to occur. To provide for instantaneous response, the **init Q** or **init q** command causes **init** to re-examine the **inittab** file.

If the **init** command receives the **powerfail** signal, **SIGPWR**, and is not in the **Single-User** level, it scans the file **inittab** for special powerfail stanzas. These stanzas are invoked, if the runlevel permits, before any further processing takes place. In this way, the **init** command performs various cleanup and recording functions whenever the operating system experiences a power failure. The powerfail stanzas should not use devices that must first be initialized after a power failure.

When the **init** command is requested to change the runlevel, via the **telinit**, command the **init** command sends the warning signal, **SIGTERM**, to all processes that are undefined in the target runlevel. The **init** command waits 20 seconds before forcibly terminating these processes via the kill signal, **SIGKILL**.

If the **init** command finds that it is continuously respawning a stanza from the file **/etc/inittab** more than 10 times in two minutes, it assumes that there is an error in the command string, generates an error message to the system console, and refuses to respawn this stanza for five minutes or until it receives a signal from a user **telinit**. This prevents **init** from using excessive system resources when a typographical error occurs in **inittab** or a program is removed that is referenced by **inittab**.

The **telinit** command can only be used by someone with superuser authority or a member of the system group. The **runlevel** parameter corresponds to a configuration of processes in the system. For example, if the system is in runlevel 1, only those **/etc/inittab** entries with a **level=1** line are processed. When **telinit** is used to change the runlevel, all processes which do not have the specified **runlevel** are sent a kill signal after a 20-second grace period. You can change the actions of the **telinit** command with the following **runlevel** options:

0-6	Tells the **init** command to process only those **/etc/inittab** file entries that have the specified **runlevel** set.
a, **b**, **c**	Tells the **init** command to process only those **/etc/inittab** file entries that have the **a**, **b**, or **c** runlevel set.
Q, **q**	Tells the **init** command to re-examine **/etc/inittab**.
s, **S**, **m**, **M**	Tells **init** to enter the **Single User** level. **Warning:** When the **Single User** level change is effected, the virtual system console (**/dev/syscon**) changes to the terminal from which the command was executed.

Environments

Because the **init** command is the ultimate ancestor of every process on the system, its environment parameters are inherited by every process. As part of its initialization sequence, the **init** command reads the file **/etc/environment** and copies any assignments found in that file into the environment passed to all of its subprocesses. It treats the **umask** differently. If it is assigned a reasonable octal value, the **init** command does a **umask** system call for the specified value, rather than passing the value in the environment. Similarly, if **filesize** is specified, the **init** command issues a **ulimit** call with the given size as the argument.

Files

/etc/inittab
/etc/utmp
/usr/adm/wtmp
/etc/ioctl.syscon
/dev/syscon
/dev/systty

Related Information

See the following commands: "getty" in topic 1.1.189, "login" in topic 1.1.241, "pstart, penable, pshare, pdelay" in topic 1.1.338, "sh, Rsh" in topic 1.1.420, and "who" in topic 1.1.537.

See the **kill** system call and the **inittab** and **utmp** file formats in *AIX Operating System Technical Reference*.

1.1.209 install

Purpose
Installs a command.

Syntax

```
                +----------------+   +------------------+        +-------+
/etc/install ---¦                +---¦ +--------------+ +-- file --¦       +-
                +- -n directory -+   +-¦ -d mode      +-+        +- dir -+
                                       ¦ -h component ¦¦                ¦
                                      ¦¦ -i           ¦¦          +-----+
                                      ¦¦ -m           ¦¦
                                      ¦¦ -o           ¦¦
                                      ¦¦ -s           ¦¦
                                      ¦¦ -v fstore    ¦¦
                                      ¦¦ -H           ¦¦
                                      ¦¦ -M mode      ¦¦
                                      ¦¦ -O owner     ¦¦
                                      ¦¦ -G group     ¦¦
                                      ¦¦ -S           ¦¦
                                      ¦+--------------+¦
                                      +----------------+

                +- -c directory ----------+   +------------------+
/etc/install ---¦                +------+ +---¦ +--------------+ +-- file --¦
                +- -f directory -¦      +-+   ¦ ¦ -d mode      ¦ ¦
                                 +- -o -+     +-¦ -h component +-+
                                                ¦ -m           ¦¦
                                               ¦¦ -s           ¦¦
                                               ¦¦ -v fstore    ¦¦
                                               ¦¦ -H           ¦¦
                                               ¦¦ -M mode      ¦¦
                                               ¦¦ -O owner     ¦¦
                                               ¦¦ -G group     ¦¦
                                               ¦¦ -S           ¦¦
                                               ¦+--------------+¦
                                               +----------------+
```

Description
The **install** command installs a **file** in a specific place within a file system. It is most often used in **makefiles** (see "make" in topic 1.1.254). When replacing files, the **install** command copies each file into the appropriate directory, thereby retaining the original owner and permissions. A newly created **file** has permission code 755, owner **bin**, and group **bin**. The **install** command writes a message telling you exactly which files it is replacing or creating and where they are going.

If you do not supply any arguments, the **install** command searches a set of default directories (**/bin**, **/usr/bin**, **/etc**, **/lib**, and **/usr/lib**, in that order) for a file with the same name as **file**. The first time it finds one, it overwrites it with **file** and issues a message indicating that it has done so. If no match is found, the **install** command issues a message telling you there was no match and exits with no further action.

If any directories are specified on the command line, the **install** command searches them before it searches the default directories.

Note: When you use **install**, it creates a hidden directory. To remove the files you have installed, use the **rm -rf** command.

Flags

-c directory	Installs a new command file in **directory** only if that file does not already exist. If it finds a copy of **file** (as a regular file, directory or hidden file), it issues a message and exits without overwriting the file.
-d mode	Specify permission mode for installed hidden directories.
-f directory	Forces installation of **file** in **directory** regardless of whether **file** already exists. If the file being installed does not already exist, the **install** command sets the permission code and owner of the new file to **755** and **bin**, respectively.
-G group	This option may be used to specify a different group for the destination file. The group may be either a decimal group ID or a group name. The default group is **bin**.
-h component	Use or create a hidden directory with the specified component. The component is that of the machine on which **install** is invoked.
-H	Create a hidden directory rather than a flat file.
-i	Ignores the default directory list and searches only those directories specified on the command line. This flag cannot be used with the **-c** or **-f** flag.
-m	The file is moved to the directory not copied.
-M mode	This option may be used to specify the octal **mode** of the destination file.
-n directory	Installs **file** in **directory** if it is not in any of the searched directories and sets the permissions and owner of the file to **755** and **bin**, respectively. This flag cannot be used with the **-c** or **-f**. If the file is found in one of the searched directories, install places the file in the directory in which it was found.
-o	Saves the old copy of **file** by copying it to **OLDfile** in the directory in which it was found. This flag cannot be used with the **-c** flag.
-O owner	This option may be used to specify a different owner of the destination file. The owner may be either a decimal user ID or a user name. The default owner is **bin**.
-s	Suppresses display of all but error messages.
-S	Strip the symbol table out of the installed executable file.
-v fstore	Specify the **fstore** value for the installed file. (See the **chfstore** command.)

Examples

1. To replace a command that already exists in one of the default directories:

   ```
   /etc/install  fixit
   ```

   This command replaces **fixit** if it is found in the directory **/bin**, **/usr/bin**, **/etc**, **/lib**, or **/usr/lib**. Otherwise, it is not installed. For example, if **/usr/bin/fixit** exists, this file is replaced by a copy of the file **fixit** in the current directory.

2. To replace a command that already exists in a specified or default directory, and to preserve the old version of:

   ```
   /etc/install  -o  fixit  /etc  /usr/games
   ```

   This command replaces **fixit** if found in **/etc**, **/usr/games**, or one of the default directories. Otherwise it is not installed. If **fixit** is replaced, the old version is preserved by renaming it **OLDfixit** in the directory in which it was found (**-o**).

3. To replace a command that already exists in a specified directory:

   ```
   /etc/install  -i  fixit  /u/tom/bin  /u/joan/bin  /usr/games
   ```

   This command replaces **fixit** if found in the directory **/u/tom/bin**, **/u/joan/bin**, or **/usr/games**. Otherwise, it is not installed.

4. To replace a command if found in a default directory, or install it in a specified directory if not found:

   ```
   /etc/install  -n  /usr/bin  fixit
   ```

   This command replaces **fixit** if found in one of the default directories. If **fixit** is not found, it is installed as **/usr/bin/fixit** (**-n /usr/bin**).

5. To install a new command:

   ```
   /etc/install  -c  /usr/bin  fixit
   ```

   This command creates a new command by installing a copy of **fixit** as **/usr/bin/fixit**, but only if this file does not already exist.

6. To install a command in a specified directory whether it already exists:

   ```
   /etc/install  -f  /usr/bin  -o  -s  fixit
   ```

   This command forces **fixit** to be installed as **/usr/bin/fixit** whether or not it already exists. The old version, if any, is preserved by moving it to **/usr/bin/OLDfixit** (**-o**). The messages that tell where the new command was installed are suppressed (**-s**).

Related Information

See the following command: "make" in topic 1.1.254.

See the **mk** system maintenance procedure in the *AIX Operating System Technical Reference*.

1.1.210 install - BSD Version

Purpose

Installs a command - BSD version.

Syntax

```
                    +--------------+
/usr/ucb/install ---¦ +----------+ +--- binary --- destination ---¦
                    +-¦ -c       +-+
                      ¦ -m mode  ¦
                      ¦ -o owner ¦
                      ¦ -g group ¦
                      ¦ -s       ¦
                      +----------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

Binary is moved (or copied if **-c** is specified) to *destination*. If *destination* already exists, it is removed before *binary* is moved. If the destination is a directory then *binary* is moved into the *destination* directory with its original file-name.

The mode for *destination* is set to 755; the **-m** *mode* option may be used to specify a different mode.

Destination is changed to owner root; the **-o** *owner* option may be used to specify a different owner.

Destination is changed to group staff; the **-g** *group* option may be used to specify a different group.

If the **-s** option is specified, the binary is stripped after being installed.

The **install** command refuses to move a file onto itself.

Related Information

See the following commands: "chgrp" in topic 1.1.65, "chmod" in topic 1.1.67, "cp, copy" in topic 1.1.91, "mv, move" in topic 1.1.282, "strip" in topic 1.1.445, "chown" in topic 1.1.68 and "install" in topic 1.1.209.

1.1.211 install-mh

Purpose
Initializes the Message Handling (MH) environment.

Syntax

```
               +---------+
install-mh ---¦         +---¦
               +- -auto -+
```

Note: This command does not have MBCS support.

Description

The **install-mh** command is used to set up mailbox directories. This command is not designed to be run directly by the user; it is designed to be called by other programs. The **install-mh** command is part of the Message Handling (MH) package.

The **install-mh** command is issued automatically the first time you run any MH command. The command prompts you for the name of your mail directory. If the directory does not exist, you are asked if it should be created. The **install-mh** command creates the file **$HOME/.mh_profile** and places the **Path:** profile entry in it. This entry identifies the location of your mailbox.

Flag

-auto	Creates the standard MH path without prompting.

Profile Entry

Path:	Specifies your **user_mh_directory**. This entry is created by the **install-mh** command.

Files

$HOME/.mh_profile	The MH user profile.

Related Information
See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.212 installp

Purpose
Installs a Licensed Program Product (LPP).

Syntax

```
            +- -d /dev/rfd0 -+   +- -n $LOGNAME -+   +--------+
installp ---¦                +---¦               +---¦ +----+ +---¦
            +-- -d device ---+   +--- -n name ---+   +-¦ -q +-+
                                                       ¦ -  ¦¦
                                                      ¦+----+¦
                                                      +------+
```

Description

Warning: Under no circumstances attempt to run more than one installation or update process at the time. The primary site must be up when executing **installp**.

Failure to perform these steps may result in a failure in the installation or update of the active files being serviced.

The **installp** command installs an LPP. You must be operating with superuser authority to run this command.

Because more than one LPP may be on a set of diskettes, **installp** asks whether you want to install each LPP. If you do, **installp** checks to see if it is an older version than the one currently installed. If the version to be installed is older than the version on the system, **installp** informs you and asks if you want to continue.

The **installp** command makes a backup copy of the LPP history file before installation begins. If installation is not successful, it sets the Version, Release, and Level, and Fix fields of the last record of the history file to 00.00.0000.0000 and logs the exit value in the program history file. The history file remains on the system as **/usr/lpp/***lpp-idd***/lpp.hist**, where *lpp-idd* is the LPP name.

1.1.212.1 Error-Recovery

Running the service tools, may change certain key files and directories mainly:

1) The global history file **/etc/lpp/ghf**

2) The LPP's individual history file, **/usr/lpp/lpp-idd/lpp.hist**

3) **/usr/lpp.save/***

4) **/usr/lib/qproc.queue/***

If a fatal error such as a system crash occurs, these files and directories may be left in a corrupted form. An error recovery system is provided.

As the service tools proceed, they log their actions in a file:

/etc/lpp/config.log

If all goes well, the above file is removed upon completion of the service. When a detectable error occurs (or the system crashes), this file remains, and is used by the error-recovery tool **/etc/lpp/inuconfig** to set the system back to before the service command was run.

When **installp** is run, it checks for the existence of this file, if it exists **installp** will know an error had previously occurred in a service tool, **installp** will then ask the user to run **/etc/lpp/inuconfig** before running **installp**. The **inuconfig** command will then recover uncorrupted versions of history files and cleanup corrupted directories when it is done, it will remove the file:

/etc/lpp/config.log

thus enabling **installp** to proceed.

Note: All installs and updates of LPPs are recorded in the global history file **/etc/lpp/ghf**.

You cannot use INTERRUPT to stop the **installp** command. To stop **installp**, press QUIT WITH DUMP. This should be used only in extreme circumstances since the state of the system cannot be predicted and one of the following may occur:

The write-verify feature may be left on for all minidisk
All terminals other than the console may be disable
Some install control files may need to be deleted

Notes:

1. Only ordinary files with the prefix **lpp.** remain in **/usr/lpp/***lpp-idd* after completion of **installp**. All other ordinary files are removed.

2. Installing a complete program on a client where a program subset already exists will cause the history file information for the program subset to be destroyed.

Flags

-d device	Installs the LPP from the specified **device**. The default **device** is **/dev/rfd0**. If a tape device is specified, the rewind device must be specified (**/dev/rmt0rh**). The **installp** command does not support the no rewind device.
-n name	Logs the first eight nonblank characters of **name** in the LPP history file. The default **name** is the value of the environment variable **$LOGNAME**.
-q	Runs in quiet mode suppressing most of the interactive queries.
-t	Prevents **installp** from rebooting after installing on LPPs.

When the installation of an LPP completes, it is sometimes necessary for the installation tools to take special actions, such as build the kernel. In these cases, a special code must be returned to the installation tools via the installation scripts **instal** or **inst_updt.loc.** described in the *AIX Operating System Programming Tools and Interfaces*, Chapter 13. This allows several LPPs to be installed before rebooting. Codes that have local implications (2, 3, 4, and 5) are returned by **inst_updt.loc**. Code 5 is returned by **instal**. Code 0 and error codes may be returned by either. This return code signals the end of the install LPP, and determines what actions the installation tools will perform next.

Use one of the following values from a return code.

Code	**Description**
0	Successful completion. No additional action is needed.
2	Successful completion. The service tools update superblocks, the inode list, and flush the buffers (**sync**). Then they instruct the user to reboot the operating system.
3	Successful completion. The service tools build a new kernel. Then they update superblocks, the inode list, and flush the buffers. They then instruct the user to reboot the system.
4	Successful completion. The service tools build a new kernel. Then they update superblocks, the inode list, and flush the buffers. The service tools instruct the user to reboot the operating system.
5	Installation cancelled by the **instal** procedure without errors.
6	Successful completion. The service tools update superblocks and the inode list, and flushes the buffers (**sync**). Then they instruct the user to reboot the system.
other	Error. If returned by **instal**, the service tools set the **Version**, **Release Level** and fix fields of the last information record in **lpp.hist** to all zeros, and write the return code value in **lpp.hist** as an error code.

Internal Commands
Install procedures can use the internal install commands. These commands provide common code for the save and recovery functions frequently needed

by most LPP-provided procedures. They do a minimum validation of input parameters and return exit values like subcommands. You can, however, receive messages from system commands that they call. Procedures that call these commands can use the **/usr/include/inu21.h** file to define return codes.

Subtopics

1.1.212.1.1 inusave

The **inusave** command saves some or all of the files and archive files that will be changed during a LPP install or update procedure. It uses the following syntax:

inusave listfile *lpp-idd*

The *lpp-idd* parameter specifies the LPP to be installed or updated. *lpp-idd* can be a maximum of 8 characters. **listfile**, which must be a full path name, contains a list of relative path names (relative to the root) for all of the files that need to be saved. **listfile** must be in the format of an **apply list** (see *AIX Operating System Programming Tools and Interfaces*, Chapter 13, for a discussion of the format of an apply list).

The **inusave** command creates the **save directory** (**/usr/lpp/***lpp-idd***/inst_updt.save**). This is the directory in which the install and update procedures store saved files and the control list that correlates the local file names with their full path names. **inusave** uses **listfile** as a basis to determine which files need to be temporarily saved.

New Method

> Simple copying, however, could not handle non-regular files (for example, pipes and devices), **textbusy** files, and the TCF attributes of regular files such as **fstores**. Therefore, **inusave** has been modified to use the **backup** utility to save the contents of the apply list. A backup-by-name format file directory called **/usr/lpp-idd/inst_updt.save/lpp.name** stores the files from the apply list which are to be used in rejecting an update. Then **inurecv** checks to see which method was used for rejecting the update and uses copy for the old method, or restore for the new method. **inurecv** is fully compatible with the old method.

An archived constituent file is saved if there is a valid archive control file, **/usr/lpp/***lpp-idd***/lpp.acf**, for the LPP. If this file exists, **inusave** compares each of the file names in **listfile** to the constituent file names in **/usr/lpp/***lpp-idd***/lpp.acf**. When it finds a match, **inusave** uses the **ar** command to extract the constituent file from its associated archive file.

It then moves it to **/usr/lpp/***lpp-idd***/inst_updt.save/archive.n**, where **n** is an integer selected by **inusave**. **inusave** maintains a list of the extracted files that have been saved in the file **/usr/lpp/***lpp-idd***/inst_updt.save/archive.list**. The format of each entry in the list is:

archive.n cfile afile

where **archive.n** is the name of the saved file and **cfile** and **afile** are the constituent and archive files defined in the archive control file.

The **inusave** command returns the following exit values:

0	No error conditions occurred.
105	Failure occurred trying to create a save directory.
107	Copy of a file from one directory to another failed. This implies that the update apply has not yet begun and that the old level of the program is still usable.
202	One or more parameters missing.
204	Too many parameters were entered.

207 `Could not access the apply list.`

1.1.212.1.2 inurecv

The **inurecv** command recovers all files and archive-constituent files saved from the previous **inusave**. **inurecv** uses the following syntax:

inurecv *lpp-idd* **reject-flag**

It uses the control lists from the **/usr/lpp/***lpp-idd***/inst_updt.save** directory to recover the files. **inusave** creates the **/usr/lpp/***lpp-idd***/inst_updt.save** directory and control lists. **inurecv** also recovers files that may have been saved by the program-provided install or update procedure (see *AIX Operating System Programming Tools and Interfaces* for details).

The **inurecv** command has to distinguish between an immediate recovery that occurs because of an error condition during an install or update and an update rejection that occurs because a user rejects an update (**updatep -r**). If the **reject-flag** argument is **yes**, **inurecv** assumes that it is being run because of an update rejection. If the argument is **no** or if no flag is specified, **inurecv** assumes that it is being run because of an immediate recovery.

The **inurecv** command returns the following exit status values:

0	No error conditions occurred.
101	The save directory does not exist.
102	A copy of a file from one directory to another failed. This implies that the LPP could not be recovered and that it must be reinstalled and any updates reapplied.
104	A file that was saved in the save directory was not found.
205	Replacement of a constituent file in an archive file failed while attempting to recover a LPP. This implies that the LPP is no longer useable and should be reinstalled and any updates reapplied.

1.1.212.1.3 inurest

The **inurest** command does simple restores and archives. It does not do any additional processing or user interaction. **inurest** uses the following syntax:

inurest [**-ddevice**] [**-q**] **listfile** *lpp-idd*

The **listfile** is the full path name of a file containing the relative directory target path name (relative to the root), of files that a LPP needs to restore. It must be in the format of an apply list. **inurest** restores all files in the list relative to the root directory. *lpp-idd* specifies the name of the LPP to be installed or updated. It can be a maximum of 8 characters.

To archive a file, there must be an archive control file, **/usr/lpp/***lpp-idd***/lpp.acf**. If it exists, **inurest** compares each of the target names in **listfile** to the component files listed in there. Whenever **inurest** finds a match, it archives the restored file into the corresponding archive file and deletes the restored file.

The following flags modify the action of **inurest**:

-d device	Specifies the input **device**. The default device is **/dev/rfd0**.
-q	Prohibits **restore** from displaying the "insert volume 1" prompt.

The **inurest** command returns the following exit status values:

0	No error conditions occurred.
106	Failed trying to restore an updated version of files.
201	An invalid flag was specified.
202	One or more parameters missing.
204	Too many parameters were entered.
206	Failed trying to replace file in an archive file.
208	Could not access the apply list.

1.1.212.1.4 ckprereq

The **ckprereq** command determines whether the system level is compatible with the LPP to be installed or updated. It uses the following syntax:

ckprereq [-v] [-f + prerequisites]

You can run **ckprereq** only if you are operating with superuser authority. **prerequisites** is a LPP prerequisite list file. Each record in this file contains the name of a prerequisite LPP and describes the version, release, and level requirements. There is one record for each prerequisite LPP. The default **prerequisites** file is **prereq**. See *AIX Operating System Programming Tools & Interfaces* for details on the format of **ckprereq** file entries.

The **ckprereq** command tests the current version, release, and level found in the global history file **/etc/lpp/ghf** and marks each "prereq state" field of the **prereq** file with one of the following codes if the test fails:

l	The test is false for level.
f	The history file format is not fixed 80.
n	The history file was not found.
r	The test is false for release.
s	There is a syntax error in the **prereq** file.
u	The history file is in an unknown state.
v	The test is false for version.

A blank "prereq state" field indicates that the test was true. The exit value of **ckprereq** is the number of records that did not test true. If all records test true, the exit value is 0.

The following flags modify the action of **ckprereq.**:

-f prerequisites	Specifies the **prerequisites** file to use in place of **prereq**.
-v	Sends a descriptive message to standard error for each failure in the prerequisite LPP test. The messages give the same information as the prereq state field of the **prereq** file.

1.1.212.1.5 Other Internal Commands

The following extra internal service commands may also be used.

/etc/lpp/insvl listfile lpp-idd
: The local equivalent to inusave, this routine works on target directory in **/local/lpp** vs **/usr/lpp** in case of inusave.

/etc/lpp/inrcvl lpp-idd
: The local equivalent to inurecv, this routine works on target directory in **/local/lpp**/ vs **/usr/lpp** in case of inurecv.

/etc/lpp/qapp
: Append a new local action to the system queue.

  qapp [-c|-r|-u|-i|-a] lpp-iddvv.rr.llll.ffff...

  qapp accepts similar flags as **updatep** (**-a, -c, -r, -u** or **-i**), indicating the action to be taken. **lpp-iddvv.rr.llll.ffff** are an LPP name and the version, release, level, and fix numbers. It creates an entry in the system queue which, upon being processed (see **qproc** below), causes the appropriate action (apply-commit-reject-uncommit) to be done on an individuals site's local file system.

/etc/lpp/qproc
: Program to trigger the processing of all pending queue entries on an individual site.

  onsite sitenumber qproc

  qproc runs on each individual site and processes all pending updatep or installp actions for that site. **qproc** first compares the local queue pointer to the global queue pointer. If local queue pointer file is not found, it is assumed that the local queue pointer value is null, meaning the site is new and all queue entries are to be processed. If local queue pointer is less than global queue pointer, a "local" action is pending. **qproc** processes the entries one by one beginning with entry after the one pointed to by the local pointer and ending with the last queue entry.

  Options:

  -i The **-i** option of **qproc** is used to install the local portion of an **LPP** on a secondary site in a cluster. Normally this function is only used by **installp**. During **installp** you are asked to select the desired installation sites of the **LPP**. These sites are passed to **qproc** for the **LPP**s that have local actions so that the **LPP** may be serviced later on those sites.

  If you decide at a later date to install an **LPP** with local actions on some other site in the cluster, run:

  "qproc -i lpp_name" on that site.

  The **LPP** will be installed in the local of the site, and brought up to date with the current level of service for that **LPP**. **NOTE:** ALL LEVELS OF SERVICE FOR THE **LPP** MUST REMAIN AVAILABLE ON THE CLUSTER AT THE TIME THAT **QPROC** IS

CALLED. **qproc** does not need to be run for **LPP**s that do not have local action.

/etc/lpp/qinvoke
Program to coordinate when **qproc** is to run on each site of the multisite cluster. On the multisite systems, this program uses the command **at** to set up jobs for each site's **qproc** to run.

On the single site systems, **qproc** is run as a part of **installp/Updatep** because time synchronization is required for a one-site system. In such cases, **qinvoke** simply calls **qproc**.

/etc/lpp/ckstack
ckstack lpp-idd vv.rr.llll.ffff

ckstack looks into the backout stack to find the frame containing the desired backout. If there are other frames (containing other **LPPs** which, due to prerequisites, must be backed out before the desired LPP), it creates a file containing a list of these other frames. This file is called **/usr/lpp.save/list**.

If there are any relevant cleaned-up stack frames, it also warns the user to restore them using **cleanup -r** command.

Files

/usr/lpp.save/...	Backout information.
/usr/lib/qproc.queue/...	System queues directory.
/local/qproc.queue/...	Local queue directory.
/usr/lpp/prm_name/lpp.loc/...	Files needed for local installations.
/usr/include/inu21.h	Defines error codes for internal commands.
/usr/lpp/*lpp-idd***/instal**	Program installation procedure.
/usr/lpp/*lpp-idd***/inst_updt.save**	Directory for saved files.
/usr/lpp/*lpp-idd***/inst_updt/inuPIDtempn**	Temporary files.
/usr/lpp/*lpp-idd***/liblpp.a**	Central archive file.
/usr/lpp/*lpp-idd***/lpp.acf**	Archive control file.
/usr/lpp/*lpp-idd***/lpp.hist**	Program history file.
/usr/lpp/*lpp-idd***/prereq**	Program prerequisite list file.
/etc/lpp/ghf	The global history file.
/etc/lpp/config.log	The error-recovery log file.
/etc/lpp/inconfig	The error-recovery utility.

Related Information

See the following command: "updatep" in topic 1.1.499 and "cleanup" in topic 1.1.72.

See the **fork** and **exec** system calls and the **lpp.hist** file in *AIX Operating System Technical Reference*.

See the discussion of installing programs in *AIX Operating System Programming Tools and Interfaces*.

See the section on "Licensed Program Product (LPP) Service Process" in the *AIX Administration Guide for System 370*.

1.1.213 installt

Purpose
Installs Licensed Program Products (LPP) or restore images from tape media.

Syntax

```
            +- -d /dev/rst4 -+   +- -l logfile -+
installt ---¦                +---+- -q ---------+---¦
            +-- -d device ---+   +- -u ---------+
```

Note: This command is for the PS/2 only.

Description
The **installt** command installs, from either the Internal Tape Backup Unit (ITBU), or 6157 tape drive, separately installable portions of IBM LPPs that are distributed on tape, or user generated tapes created in the proper format.

The **installt** command reads a "table of contents" (TOC) file from the install media and generates a menu from which the user can select the entries that are to be installed. For each entry selected, **installt** invokes either the **installp** command or the **restore** command to perform the actual installation. The decision on which command to invoke is based on information supplied with the entry name in the TOC file. The **installt** command will install all of the entries selected by the user without prompting from **installp**, however, the user may still be prompted for input required by specific LPP installation scripts.

Tapes used with **installt** must be in a special format, described in Appendix D of the *Programming Tools and Interface Guide*.

When installing backup images that were created with relative file path names, you must be in the directory where the files are to be installed. **installp** images do not require you to be in any particular directory during installation.

Flags

-d device	Install from the device specified. This device must be a no-rewind device. The default is **/dev/rst4** (Internal Tape Backup Unit).
-l logfile	Append output from **installp** to the log file specified using the **tee** command. The log file is initially removed, and output is appended to the log file for each invocation of **installp**. The default is no log file.
-q	Install all of the packages on the tape without using the menu interface. All packages are loaded and use the default fstore values found in the TOC on the tape media. The default is to use the menu interface.
-u	Specify user created tape. The first tape file on a user created tape is the TOC. The default is that this flag is not specified and the TOC is the second tape file.

Examples

1. To invoke **installt** using the menu interface and an **ITBU** tape.

   ```
   installt
   ```

2. To use **installt** without the menu interface, using a 6157 Streaming Tape, and writing log information to a file:

   ```
   installt -d/dev/rmt4 -q -l/tmp/logfile
   ```

3. To use **installt** on an **ITBU** tape device, and the tape does not contain the initial BOS tape file:

   ```
   installt -u
   ```

Files
/dev/rmt4
/dev/rst4
/etc/fstore

Related Information
See the following commands: "backup" in topic 1.1.32, "restore" in topic 1.1.371, "installp" in topic 1.1.212, and "tee" in topic 1.1.467.

See "Installing and Updating an LPP" in *Programming Tools and Interfaces Guide*.

See "Appendix D - installt Command" in *Programming Tools and Interfaces Guide*.

See "Installing Additional Licensed Program Products" in *Installing and Customizing the Operating System for PS/2*.

1.1.214 inuconfig

Purpose

Recovers errors for installation/update tools.

Syntax

/etc/lpp/inuconfig ---¦

Description

The **inuconfig** command is the error-recovery utility for the installation and update tools (**installp** and **updatep**).

As the service tools proceed, they log their actions in a file:

/etc/lpp/config.log

If all goes well, this file is removed upon completion of the service. When a detectable error occurs (or the system crashes), this file remains and is used by the error-recovery tool **inuconfig** to set the system back to before the service command was run. The command can be run again, after the error condition (for example the root file system is full) is eliminated. The log file contains entries as shown below. Each entry indicates a certain step in the progress of service tools, and each requires a specific action by error-recovery to undo.

The entries are as follows:

00 flag Where **flag** is **a** (apply), **r** (reject), **c** (commit), or **u** (uncommit). Put in by **installp** or **inudatep**. Error-recovery will note the flag and report it to the user.

01 Global history file was backed up by the service tool. Error recovery recovers the global history file.

02 lpp_name
Where **lpp_name** is the **LPP** being processed as specified by installp, inudatep, and uninst. Individual history file will be backed-up, and error-recovery will recover **lpp.hist**. In cases where flag is a or ac, files in **/usr/lpp/ lpp_name** save are recovered by calling **/etc/inurecv**.

03 queue#
Where **queue**# is the queue entry being processed as specified by **/etc/lpp/qapp.b**. Error-recovery will remove entry number **queue# -1**. If there are more than one such entries, the queue pointer will be set to the lowest possible value.

04 frame#
Where **frame#** is the stack frame being processed. Put in by **/etc/lpp/rest**. Error-recovery will remove **/usr/sys/inst_updt.*,** then set stack pointer to **frame#**. If there are more than one such entries, the stack pointer will be set to the highest possible value.

05 frame#
Where **frame#** is the stack frame being processed. Put in by

/etc/lpp/save. Error-recovery will remove frame number **frame#**, then set stack pointer to **frame# -1**. If there are more than one such entries, the stack pointer will be set to the lowest possible value.

1.1.215 invert, lookup

Purpose
Creates and accesses an inverted index.

Syntax

```
          +-------------+
invert ---¦ +---------+ +--- file ---¦
          +-¦ -c file +-+          ¦
            ¦ -k i    ¦   +--------+
            ¦ -l i    ¦
            ¦ -p file ¦
            ¦ -%str   ¦
            ¦ -s      ¦
            +---------+
          +-------------+
lookup ---¦ +---------+ +---¦
          +-¦ -c file +-+
            ¦ -l      ¦¦
           ¦¦ -p file ¦¦
           ¦+---------+¦
           +-----------+
```

Note: This command does not have MBCS support.

Description
The **invert** command creates an inverted index to one or more files. The **lookup** command retrieves records from files for which an inverted index exists. The inverted indices are intended for use with **bib**.

The **invert** command creates one inverted index to all of its input files. The index must be stored in the current directory and may not be removed. Input files are absolute path names or paths relative to the current directory. Each input file is viewed as a set of records. Each record consists of non-blank lines and records that are separated by blank lines.

The **lookup** command retrieves records based on its input (stdin). Each line of input is a retrieval request. Records that contain all of the keywords in the retrieval request are sent to stdout. If there are no matching references, the message No References Found is sent to stdout. The lookup command first searches the user's private index (default INDEX). If no references are found, it then searches in the system index (**/usr/dict/papers/INDEX**). The system index is produced using invert with the default options. The user is advised to use the defaults.

Keywords are a sequence of non-white space character with non-alphanumeric characters removed. Keywords must be at least two characters and are truncated (default length is 6). Some common words are ignored. Some lines of input are ignored for the purpose of collecting keywords.

Flags
The following flags apply to both **invert** and **lookup**:

-c File is the name of a file containing common words, one per line. Common words are not used as keys. (Default is /usr/lib/bmac/common.)

-l Maximum length of keys. (Default is 6.)

-p	File is the name of the private index file, output of invert. (Default is INDEX.) Index must be stored in the current directory.

Note: The **lookup** command accepts only the options **c**, **l**, and **p** with the same meanings as bib. In particular, the **p** option can only be followed by a list of comma-separated index files. These are searched in order from left to right until at least one reference is found.

The following flags apply to **invert** only:

-k	The space following **-k** is optional. The maximum number of keys kept per record. (Default is 100.)
-s	Suppresses statistics.
-%str	Ignores lines that begin with **%x**, where **x** is in **str**. (Default is CNOPVX.) See the commands "bib, listrefs" in topic 1.1.40 for an explanation of field names.

Files

INDEX	Inverted index.
/usr/tmp/invertxxxx	Scratch file for invert.
/usr.lib.bmac/common	Default list of common words.
/usr/dict/papers/INDEX	Default system index.

Diagnostics

Messages indicating trouble accessing files are sent on stderr. There is an explicit message on stdout from lookup if no references are found.

The **invert** command produces a one-line message of the form:

```
  %D document  %D distinct keys %D key occurrences
```

This can be suppressed with the **-s** option.

The message

```
  locate: first key (%s) matched too many refs
```

indicates that the first key matched more references than could be stored in memory. The simple solution is to use a less-frequently

Related Information

See the **bib, listrefs** commands in the *AIX Operating System Commands Reference*.

1.1.216 ipcrm

Purpose
Removes message queue, semaphore set, or shared memory identifiers.

Syntax

```
         +---------------+
ipcrm ---¦ +-----------+ +---¦
         +-¦ -m shmid  +-+
           ¦ -q msqid  ¦¦
          ¦¦ -s semid  ¦¦
          ¦¦ -M shmkey ¦¦
          ¦¦ -Q msgkey ¦¦
          ¦¦ -S semkey ¦¦
          ¦+-----------+¦
          +-------------+
```

Description
The **ipcrm** command removes one or more message queue, semaphore set, or shared memory identifiers.

Flags

-m shmid	Removes the shared memory identifier **shmid**. The shared memory segment and data structure associated with **shmid** are also removed after the last detach.
-q msqid	Removes the message queue identifier **msqid** and the message queue and data structure associated with it.
-s semid	Removes the semaphore identifier **semid** and the set of semaphores and data structure associated with it.
-M shmkey	Removes the shared memory identifier, created with key **shmkey**. The shared memory segment and data structure associated with it are also removed after the last detach.
-Q msgkey	Removes the message queue identifier, created with key **msgkey**, and the message queue and data structure associated with it.
-S semkey	Removes the semaphore identifier, created with key **semkey**, and the set of semaphores and data structure associated with it.

The details of the remove operations are described under the **msgctl**, **shmctl**, and **semctl** in the *AIX Operating System Technical Reference*. The identifiers and keys can be found by using the **ipcs** command.

Related Information
See the following command: "ipcs" in topic 1.1.217.

See the **msgctl**, **msgget**, **msgrcv**, **msgsnd**, **semctl**, **semget**, **semop**, **shmctl**, **shmget**, and **shmop** system calls in *AIX Operating System Technical Reference*.

1.1.217 ipcs

Purpose
Reports inter-process communication facility status.

Syntax

```
        +- -m -q -s -+   +------------+   +- -C /dev/kmem -+   +---- -N /unix
ipcs ---¦   +----+   +---¦ +-- -a --+ +---¦                +---¦
        +---¦ -m +---+   +-¦ +----+ +-+   +- -C corefile --+   +- -N kernel_i
            ¦ -q ¦ ¦       +-¦ -b +-+
          ¦ ¦ -s ¦ ¦         ¦ -c ¦
          ¦ +----+ ¦        ¦¦ -o ¦¦
          +--------+        ¦¦ -p ¦¦
                            ¦¦ -t ¦¦
                            ¦+----+¦
                            +------+
```

Description
The **ipcs** command writes to the standard output information about active inter-process communication facilities. If you do not specify any flags, the **ipcs** command writes information in a short form about currently active message queues, shared memory segments, and semaphores.

The column headings and the meaning of the columns in an **ipcs** listing follow. The letters in parentheses indicate the flags that cause the corresponding heading to appear. The **all** designation means that the heading always appears. These flags only determine what information is provided for each facility; they do not determine which facilities will be listed.

T (**all**) Type of facility:

q	message queue
m	shared memory segment
s	semaphore.

ID (**all**) The identifier for the facility entry.

KEY (**all**) The key used as a parameter to the **msgget**, **semget**, or **shemget** call to make the facility entry.

Note: The key of a shared memory segment is changed to **IPC_PRIVATE** when the segment is removed until all processes attached to the segment detach it.

MODE (**all**) The facility access modes and flags. The mode consists of 11 characters that are interpreted as follows:

The first two characters can be:

R	If a process is waiting on a **msgrcv** system call.
S	If a process is waiting on a **msgsnd** system call.
D	If the associated shared memory segment has been removed. It disappears when the last process attached to the segment detaches it.
C	If the associated shared memory segment is to be cleared when the first attach is run.
-	If the corresponding special flag is not set.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next to permissions of others in the user-group of the facility entry; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.

The permissions are indicated as follows:

r	If read permission is granted
w	If write permission is granted
a	If alter permission is granted
-	If the indicated permission is **not** granted.

OWNER (**all**) The login name of the owner of the facility entry.

GROUP (**all**) The name of the group that owns the facility entry.

CREATOR (**a,c**) The login name of the creator of the facility entry.

CGROUP (**a,c**) The group name of the group of the creator of the facility entry.

Note: For the **OWNER**, **GROUP**, **CREATOR**, and **CGROUP**, the user and group IDs display instead of the login names.

CBYTES (**a,o**) The number of bytes in messages currently outstanding on the associated message queue.

QNUM (**a,o**) The number of messages currently outstanding on the associated message queue.

QBYTES (**a,b**) The maximum number of bytes allowed in messages outstanding on the associated message queue.

LSPID (**a,p**) The ID of the last process that sent a message to the associated queue.

LRPID (**a,p**) The ID of the last process that received a message from the associated queue.

STIME (**a,t**) The time when the last message was sent to the associated queue.

RTIME (**a,t**) The time when the last message was received from the associated queue.

CTIME (**a,t**) The time when the associated entry was created or changed.

NATTCH (**a,o**) The number of processes attached to the associated shared memory segment.

SEGSZ (**a,b**) The size of the associated shared memory segment.

CPID (**a,p**) The process ID of the creator of the shared memory entry.

LPID (**a,p**) The process ID of the last process to attach or detach the shared memory segment.

ATIME (**a,t**) The time when the last attach was completed to the associated shared memory segment.

DTIME (**a,t**) The time the last detach was completed on the associated shared memory segment.

NSEMS (**a,b**) The number of semaphores in the set associated with the semaphore entry.

OTIME (**a,t**) The time the last semaphore operation was completed on the set associated with the semaphore entry.

Flags

-a Uses the **-b**, **-c**, **-o**, **-p** and **-t** flags.

-b Writes the maximum number of bytes in messages on queue for message queues, the size of segments for shared memory, and the number of semaphores in each semaphores set.

-c Writes the login name and group name of the user that made the facility.

-Ccorefile Uses the file **corefile** in place of the file **/dev/kmem**. **The corefile** is a memory image file produced by the **Ctrl-**(left)**Alt-Pad7** key sequence.

-m Writes information about active shared memory segments.

-Nkernel-image Uses the specified **kernel-image** (**/unix** is the default).

-o Writes the following usage information:

Number of messages on queue
Total number of bytes in messages in queue for message queues
Number of processes attached to shared memory segments.

-p Writes the following:

Process number of the last process to send or receive a message on message queues
Process number of the process that created the shared memory segment
Process number of last process to attach or detach on shared memory segments.

-q Writes information about active message queues.

-s Writes information about active semaphore set.

-t Writes the following:

Time of the last control operation that changed the access permissions for all facilities
Time of the last **msgsnd** and last **msgrcv** system calls message queues
Time of the last **shmat** and last **shmdt** system calls on shared memory

Time of the last **semop** system call on semaphore sets.

Files

/unix	System kernel image.
/dev/kmem	Memory.
/etc/passwd	User names.
/etc/group	Group names.

Related Information

See the **ipcs**, **msgrcv**, **msgsnd**, **semop**, **shmat**, and **shmdt** system calls in *AIX Operating System Technical Reference*.

See "Memory Dump Services" in *Managing the AIX Operating System*.

1.1.218 istat

Purpose
Examines inodes.

Syntax

```
        +----- filename ------+
istat ---¦                     +---¦
        +- inumber -- device -+
```

Description
The **istat** command writes information about the inodes specified with **inumber** to standard output. Use the **istat** command to write information about the inode for a specified **file name**, or to write the contents of a specified inode, **inumber** on an arbitrary file system.

If you specify **file name**, **istat** writes the following information about the file:

The device where the file resides
The inode number of the file, on that device
Global File system number (gfs)-network-wide unique for the logica file system. Separate copies of same replicated file system have the same gfs; it is the pair (gfs number, inode number) that uniquely identifies a file.
The file type (normal, directory, block device, and so on)
What protection is on the file
The name and identification number of the owner and group
The number of links to the file
If the inode is for a normal file, the length of the file
If inode is for a device, the major and minor device designations
Inode Generation number - the number of times this Inode has bee allocated.
Version number of the file - count of number of times the file ha been updated.
Commit sequence number - number of file system changes which precede last change to this file.
fstore value - indication of which sites store the file system if th file system is replicated.
The date of the last time the inode was updated
The date of the last time the file was modified
The date of the last time the file was referenced

If you specify **inumber** and **device**, **istat** also displays, in long decimal values, the block numbers recorded in the inode. You can specify the **device** as either a device name or as a mounted-file-system name.

Examples

1. To display the information stored in a file inode:

   ```
   istat  /bin/sh
   ```

   This displays the inode information for the file **/bin/sh**. The information looks something like this:

   ```
   Inode 3812 on global file system 1 (site1 device 0/1)
   Type: file                    Protection:  r-xr-xr-x
   ```

```
   Owner:  2(bin)                 Group:       2(bin)
   Link count: 2                  Length:      83460 bytes

   Inode Generation: 91           Version:  2
   Commit Sequence:  629019
   fstore: 7000 (i386)

   Last update:    Wed Mar 15 10:09:29 1989
   Last modified:  Wed Mar 15 10:09:29 1989
   Last accessed:  Wed Mar 15 11:15:57 1989
```

2. To display inode information if given a file i-number:

   ```
   istat  34  /dev/hd0
   ```

   This displays the information contained in inode number **34** on the **/dev/hd0** device. In addition to the information shown in Example 1, this displays:

   ```
   Block pointers:
        219   220   221   222   223   224   225   226
        227   228   229    0     0
   ```

   These numbers are addresses of the disk blocks that contain the data in the file.

Related Information

See the following command: "fsdb" in topic 1.1.178.

See the **statx** system call and the **file systems** and **fs** files in *AIX Operating System Technical Reference*.

1.1.219 join

Purpose
Joins data bases files.

Syntax

```
        +---------------+   +---------+   +-------------+   +-----------------
join ---¦ +-----------+ +---¦ one of  +---¦ +---------+ +---¦
        +-¦ -e string +-+   ¦ +-----+ ¦   +-¦ -j  num +-+   +- -o --- n.num --
          ¦ -t cha    ¦¦    +-¦ -a1 +-+     ¦ -j1 nu  ¦¦                    ¦
         ¦+-----------+¦      ¦ -a2 ¦      ¦¦ -j2 num ¦¦          +--- , ----
         +-------------+      +-----+      ¦+---------+¦
                                           +-----------+
```

¦ Do not put a blank on either side of the comma.

Description
The **join** command reads **file1** and **file2**, joins lines in the files according to the flags, and writes the results to standard output. Both files must be sorted according to the collating sequence specified by the **LANG** and **LC_COLLATE** environment variables, if set, for the fields on which they are to be joined (normally the first field in each line).

One line appears in the output for each identical **join field** appearing in both **file1** and **file2**. The join field is the field in the input files that **join** looks at to determine what will be included in the output. The output line consists of the join field, the rest of the line from **file1**, and the rest of the line from **file2**. You can specify standard input in place of **file1** by substituting a **-** (minus) for the name.

Fields are normally separated by a blank or a tab character. In this case, **join** treats consecutive separators as one, and discards leading separators.

Flags

-anum	When **num** is **1**, **join** produces an output line for each line found in **file1** but not in **file2**. When **num** is **2**, **join** produces an output line for each line found in the **file2** but not in **file1**.
-e string	Replaces empty output fields with **string**.
-j[n] num	Joins the two files on the **num**th field of file **n**. **n** is **1** or **2**. If you do not specify **n**, **join** uses the **num**th field in each file.
-o n.num[,n.num...]	Makes each output line consist of the fields specified in **list**, in which each element has the form **n.num**, where **n** is a file number and **num** is a field number.
-tchar	Uses **char** as the field separator character in the input and the output. Every appearance of **char** in a line is significant. The default separator is a blank. If you specify **-t**, the sequence is that of a

plain sort. To specify a tab character, enclose it in single quotation marks (' ').

Examples

Note: The vertical alignment shown in these examples may not be consistent with your output.

1. To perform a simple join operation on two files whose first fields are the same:

```
join  phonedir  names
```

If **phonedir** contains the following telephone directory:	and **names** is this listing of names and department numbers:	then **join** disp
Brown J. 555-6235	Elder Dept. 389	Elder G. 5
Dickerson B. 555-1842	Frost Dept. 217	Grabczak P. 5
Elder G. 555-1234	Grabczak Dept. 311	McGuff M. 5
Grabczak P. 555-2240	McGuff Dept. 454	Wilde C. 5
Harper M. 555-0256	Wilde Dept. 520	
Johnson M. 555-7358		
Lewis B. 555-3237		
McGuff M. 555-5341		
Wilde C. 555-1234		

Each line consists of the join field (the last name), followed by the rest of the line found in **phonedir** and the rest of the line in **names**.

2. To display unmatched lines with the command:

```
join  -a2  phonedir  names
```

If **phonedir** contains:	and **names** contains:	then **join** displa
Brown J. 555-6235	Elder Dept. 389	Elder G. 5
Dickerson B. 555-1842	Frost Dept. 217	Frost
Elder G. 555-1234	Grabczak Dept. 311	Grabczak P. 5
Grabczak P. 555-2240	McGuff Dept. 454	McGuff M. 5
Harper M. 555-0256	Wilde Dept. 520	Wilde C. 5
Johnson M. 555-7358		
Lewis B. 555-3237		
McGuff M. 555-5341		
Wilde C. 555-1234		

This performs the same join operation as in Example 1, and also lists the lines of **names** that have no match in **phonedir**. It includes **Frost**'s name and department number in the listing, although there is no entry for **Frost** in **phonedir**:

3. To display selected fields:

```
join  -o 2.3 2.1 1.2 1.3  phonedir names
```

This displays the following fields in the order given:

Field 3 of **names**	(Department Number)
Field 1 of **names**	(Last Name)
Field 2 of **phonedir**	(First Initial)
Field 3 of **phonedir**	(Telephone Number)

If **phonedir** contains:	and **names** contains:	then **join** displa
Brown J. 555-6235 Dickerson B. 555-1842 Elder G. 555-1234 Grabczak P. 555-2240 Harper M. 555-0256 Johnson M. 555-7358 Lewis B. 555-3237 McGuff M. 555-5341 Wilde C. 555-1234	Elder Dept. 389 Frost Dept. 217 Grabczak Dept. 311 McGuff Dept. 454 Wilde Dept. 520	389 Elder 311 Grabczak 454 McGuff 520 Wilde

4. To perform the join operation on a field other than the first:

```
sort  +2 -3  phonedir | join -j1 3  - numbers
```

This combines the lines in **phonedir** and **names**, comparing the third field of **phonedir** to the first field of **numbers**.

First, this sorts **phonedir** by the third field, because both files must be sorted by their join fields. The output of **sort** is then piped to **join**. The **-** (minus sign) by itself causes the **join** command to use this output as its first file. The **-j1 3** defines the third field of the sorted **phonedir** as the join field. This is compared to the first field of **numbers** because its join field is not specified with a **-j** flag.

If **numbers** contains:	this command displays the names listed in **phonedir** for each telephone number:
555-0256 555-1234 555-5555 555-7358	555-0256 Harper M. 555-1234 Elder G. 555-1234 Wilde C. 555-7358 Johnson M.

Note that **join** lists all the matches for a given field. In this case,

join lists both **Elder G.** and **Wilde C.** as having the telephone number **555-1234**. The number **555-5555** is not listed because it does not appear in **phonedir**.

Related Information

See the following commands: "awk, nawk, oawk" in topic 1.1.29, "comm" in topic 1.1.83, "sort" in topic 1.1.432, "cut" in topic 1.1.107, and "paste" in topic 1.1.313.

See the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.220 keyboard

Purpose
Controls the delay and repetition rates of the keyboard.

Syntax

```
            +---------+
keyboard ---¦ -d rate +---¦
            ¦ -r rate ¦ ¦
          ¦ +---------+ ¦
          +-------------+
```

Note: This command is for the PS/2 only.

Description

The **keyboard** command changes the keyboard delay and repetition rates. These rates are initially set at system startup to 500 milliseconds and 11 characters per second, respectively.

Flags

-d rate	Sets the delay rate to the specified value. The **rate** can be **250**, **500**, **750**, or **1000** milliseconds.
-r rate	Sets the rate of repetition to the specified value. This **rate** can be an integer from **2** to **30**, inclusive.

Example

To change both the delay and repetition rates:

```
  keyboard  -d250  -r30
```

This command sets the keyboard to a delay of 250 milliseconds and the repetition rate to 30 characters per second.

1.1.221 kill

Purpose
Sends a signal to a running process.

Syntax

```
        +------------+
kill ---¦            +--- process-ID ---¦
        +- - signal -+                  ¦
                      +---------------+

kill --- -l ---¦
```

Description
The **kill** command sends a **signal** to a running process. The default signal is signal 15 (**SOFTWARE TERMINATE**). This default signal normally kills processes that do not catch or ignore the signal. The **process-ID** (PID or process identification number) is used to identify the process you want to send a signal to. The shell reports the PID of each process that is running in the background (unless you start more than one process in a pipeline, in which case the shell reports the number of the last process). You can also use the **ps** command to find the process ID number of commands.

A signal can be a symbolic name or a number. These symbolic names and numbers are:

Symbolic Name	**Number**	**Description**
HUP	1	Hangup
INT	2	Interrupt
QUIT	3	Quit
ILL	4	Illegal instruction
TRAP	5	Trace trap
ABRT	6	Abort process.
EMT	7	EMT instruction
FPE	8	Floating point exception
KILL	9	Kill
BUS	10	Bus error
SEGV	11	Segmentation violation
SYS	12	Bad parameter to system call
PIPE	13	Write on a pipe when there is no process to read it
ALRM	14	Alarm clock
TERM	15	Software termination signal (**default**)
URG	16	Urgent condition on I/O channel
STOP	17	Stop
TSTP	18	Interactive stop
CONT	19	Continue if stopped
CHLD	20	To parent on child stop or exit
TTIN	21	Background read attempted from control terminal
TTOU	22	Background write attempted to control terminal
IO	23	Input/output possible or completed
XCPU	24	CPU time limit exceeded
XFSZ	25	File size limit exceeded
MSG	27	HFT input data pending
WINCH	28	Window size change

PWR	29	Power failure imminent
USR1	30	User-defined signal 1
USR2	31	User-defined signal 2
PROF	32	Profiling time alarm
DANGER	33	System crash imminent
VTALRM	34	Virtual time alarm
MIGRATE	35	Migrate processes to signal sender's site
PRE	36	Programming exception
GRANT	60	HFT monitor mode granted
RETRACT	61	HFT monitor access should be relinquished
SOUND	62	An HFT sound control has completed execution

In addition, there are special **process-ID**s that cause the following special actions:

0	The **signal** is sent to all processes having a process-group ID equal to the process-group ID of the sender (except those with PIDs 0 and 1).
-1	If the effective user ID of the sender is not 0 (root), **signal** is sent to all processes with a real or effective user ID that matches the real or effective user ID of the sender (except those with PIDs 0 and 1). If the effective user ID of the sender is 0 (root), **signal** is sent to all processes, excluding numbers 0 and 1.
-process-ID	The **signal** is sent to all processes whose process-group number is equal to the absolute value of **process-ID**. When you specify a minus PID, you must also specify the **signal** to be sent, even signal 15.

See the **kill** system call in *AIX Operating System Technical Reference* for a complete discussion of the **kill** command. For a complete discussion of the signal numbers, see the **sigaction** system call in *AIX Operating System Technical Reference*.

If you do not have superuser authority, the process you wish to stop must belong to you. If you have superuser authority, you can kill any process.

Flag

-l Lists the symbolic names of signals that may be specified instead of signal numbers.

Examples

1. To stop a given process:

   ```
   kill  1095
   ```

   This command stops process **1095** by sending it the default signal, which is signal 15 (also called **TERM**). Process **1095** might not actually stop if it has made special arrangements to ignore or override signal 15.

2. To stop several processes that ignore the default signal:

   ```
   kill  -9  1034  1095
   ```

This command sends signal **9** (**KILL**) to processes **1034** and **1095**. Signal 9 is a special signal that normally cannot be ignored or overridden.

3. To stop all of your background processes:

   ```
   kill  0
   ```

   This command sends signal 15 to all members of the shell process group. This includes all background processes started with **&**. (See page 1.1.420.1 about running background processes.) Although the signal is sent to the shell, it has no effect because the shell ignores signal 15.

4. To stop all of your processes and log out:

   ```
   kill  -9  0
   ```

   This command sends signal **9** to all members of the shell process group. Because the shell cannot ignore signal 9, this command also stops the login shell and logs you out. If you are using multiple windows on a high-function terminal, this command closes the active window.

5. To kill all processes that you own:

   ```
   kill  -9  -1
   ```

   This command sends signal **9** to all processes owned by the user, even those started at other work stations and that belong to other process groups. If you are using multiple windows on a high-function terminal, this command closes all of the windows. If a listing that you requested is being printed, it is also stopped.

   Note: To send signal 15 with this form of the **kill** command, you must specify **-15** explicitly:

   ```
   kill  -15  -1
   ```

6. To send a different signal code to a process:

   ```
   kill  -30  1103
   ```

   This command sends signal **30** (**USR1**) to process **1103**.

   The name of the **kill** command is misleading because many signals, including **16**, do not stop processes. The action taken on signal **16** is defined by the particular application you are running.

Related Information
See the following commands: "csh" in topic 1.1.100, "ps" in topic 1.1.337, and "sh, Rsh" in topic 1.1.420.

Note: The **csh** command contains a built-in subcommand named **kill**. The command and subcommand do not always work the same way. For information on the subcommand, see the **csh** command.

See the **kill** and **sigaction** system calls in *AIX Operating System Technical Reference*.

1.1.222 killall

Purpose
Cancels all processes except the calling process.

Syntax

```
          +-------------+   +----------------+
killall ---¦   one of    +---¦                +---¦
          ¦ +---------+ ¦   +- kernel-image -+
          +-¦ -       +-+
            ¦ -signal ¦
            +---------+
```

Description
Warning: The **killall** command must be executed from the Bourne Shell, or you will get unexpected results.

The **killall** command cancels all processes that you started, except those producing the **killall** process. This command provides a convenient means of cancelling all processes created by the shell that you control. When started by a user operating with superuser authority, the **killall** command cancels all processes that can be canceled except those that started it.

The **kernel-image** parameter specifies the name of the system load module (by default, **/unix**).

Flags

-	Sends a **SIGTERM** signal initially and then sends a **SIGKILL** (kill) signal to all processes that survive for 30 seconds after receipt of the signal first sent. This gives processes that catch **SIGTERM** signal an opportunity to clean up. (For more information, see the **signal** system call in *AIX Operating System Technical Reference*.)
-signal	Sends the specified **signal** number. (For information about signal numbers, see the **signal** system call in *AIX Operating System Technical Reference*.)

Examples

1. To stop all background processes that have started:

   ```
   killall
   ```

   This command sends all background processes the kill signal 9 (also called **SIGKILL**).

2. To stop all background processes, giving them a chance to clean up:

   ```
   killall -
   ```

   This command sends signal 15 (**SIGTERM**), waits 30 seconds, and then sends signal 9 (**SIGKILL**).

3. To send a specific signal to the background processes:

   ```
   killall -2
   ```

This command sends signal **2** (**SIGINT**) to the background processes.

Restrictions

If the Transparent Computing Facility is installed, only those processes running on the site where **killall** is run are cancelled. To determine on which sites your processes are running, use the command **ps -r**.

Files

/unix	System kernel image.
/dev/mem	Used for reading the process table.

Related Information

See the following command: "kill" in topic 1.1.221.

See **signal** system call in *AIX Operating System Technical Reference*.

1.1.223 last

Purpose
Looks at logins and logouts for information about users.

Syntax

```
        +--------------------+
last ---¦       +----------+ +---¦
        +- -num-¦ +------+ +-+
                +-¦ name +-+
                  ¦ tt   ¦¦
                 ¦+------+¦
                 +--------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **last** command will look back in the **wtmp** file which records all logins and logouts for information about a user, a teletype or any group of users and teletypes. Arguments specify names of users or teletypes of interest. Names of teletypes may be given fully or abbreviated. For example **last 0** is the same as **last tty0**. If multiple arguments are given, the information which applies to any of the arguments is printed. For example **last root console** would list all of root's sessions as well as all sessions on the console terminal. The **last** command will print the sessions of the specified users and teletypes, most recent first, indicating the times at which the session began, the duration of the session, and the teletype which the session took place on. If the session is still continuing or was cut short by a reboot, **last** so indicates.

The pseudo-user **reboot** logs in at reboots of the system, thus

```
  last reboot
```

will give an indication of mean time between reboot.

The **last** command with no arguments prints a record of all logins and logouts, in reverse order.

If **last** is interrupted, it indicates how far the search has progressed in **wtmp**. If interrupted with an INTERRUPT signal (generated by a **Ctrl-C**), **last** indicates how far the search has progressed so far, and the search continues.

Flag

-num	Limits the report to **num** lines.

Files

/usr/adm/wtmp	Login data base.
/usr/adm/shutdownlog	Records shutdowns and reasons for logins.

Related Information

See the following commands: "ac" in topic 1.1.5 and "lastcomm" in

```
topic 1.1.224.
```

1.1.224 lastcomm

Purpose
Prints information about all the commands recorded during the current accounting file's lifetime.

Syntax

```
           +--------------------------------------------+
lastcomm ---¦                                            +---¦
           +- command-name --- user-name --- terminal-name -+
```

Description

The **lastcomm** command gives information on previously executed commands. With no arguments, the **lastcomm** command prints information about all the commands recorded during the current accounting file's lifetime. If called with arguments, only accounting entries with a matching command name, user name, or terminal name are printed. So, for example,

```
  lastcomm a.out root tty0
```

would produce a listing of all the executions of commands named **a.out** by user **root** on the terminal **tty0**.

For each process entry, the following are printed.

    The name of the user who ran the process
    Flags, as accumulated by the accounting facilities in the system
    The command name under which the process was called
    The amount of CPU time used by the process (in seconds)
    The time the process exited

The flags are encoded as follows:

"S" indicates the command was executed by the superuse

"F" indicates the command ran after a fork, but without a followin **exec**

"D" indicates the command terminated with the generation of a **core** file, and

"X" indicates the command was terminated with a signal

Files

/usr/adm/pacct	Current file for process accounting.

Related Information
See the following commands: "last" in topic 1.1.223 and "acct/*" in topic 1.1.6.

1.1.225 lconfig

Purpose

System queue debugging tool.

Syntax

onsite --- **site** --- **/etc/lpp/lconfig** ---¦

Description

The **lconfig** command is a debugging tool for when an error occurs in the processing of the system queue on a particular site. When such error occurs, **qproc** creates a file **lock** in the local queue directory. This file contains the error code received from **updtp1** and indicated an error on the site.

When invoked, **updtpl** looks in the local queue directory for the file

/local/qproc.queue/lock.

If this file is not present **lconfig** concludes that no error occurred on the site and exits.

If the lock file is present, **lconfig** reads it and the local queue pointer

/local/qproc.queue/pointer.

It then looks in the system queue directory for the file

/usr/lib/qproc.queue/pointer/info.

It uses the above information to create a list of relevant information about the error, with fields as follows:

Field	**Description**
ERROR#	Error number as returned from **updtp1**
QUEUE-ENTRY#	Queue entry where the error occurred
LPPNAME	Name of the lpp being processed
VERSION-RELEASE-LEVEL	Version, release, and level of **lpp** being processed
ACTION-TYPE	Type of action (Update-fixes or Upgrade/Install)
ACTION-PHASE	Phase of the action (apply, commit, reject, uncommit)

1.1.226 ld

Purpose
Links object files.

Syntax

```
      +----------------------------+   +-- -o a.out ---+
ld ---¦   +------------------+   +---¦               +---
      +---¦ -d -elabel -qtarg +---+   +--- -o name ---+
          ¦ -i -Bnum   -Qfile ¦ ¦                   ¦
        ¦ ¦ -K -Dnum   -Ysym  ¦ ¦       +-----------+
        ¦ ¦ -m -Hnum   -Zstr  ¦ ¦
        ¦ ¦ -n -Ldir   -usym¦ ¦ ¦
        ¦ ¦ -r -Tnum   -xa    ¦ ¦
        ¦ ¦ -s -ss            ¦ ¦
        ¦ ¦ -x -Vnum          ¦ ¦
        ¦ +------------------+ ¦
        +----------------------+
    +------------+   +----------+
 ---¦            +---¦          +---¦
    +--- file ---+   +- -l key -+
               ¦
      +--------+
```

¦ You must put a blank between these items.

Description
The **ld** command (also called the linkage editor) combines the specified object **file**s into one file, resolving external references and searching libraries. It produces an object module that can be run or that can become a **file** parameter in another call to **ld** command. In the latter case, you must use the **-r** flag to preserve the relocation bits. The **ld** command places its output in a file named **a.out**. It makes this file executable if no errors occur during the link and if the **-r** flag is not specified.

The **ld** command links object files and searches object libraries in the order specified. It links object modules unconditionally, but links from the library only those files that define an unresolved external reference. If a routine from a library calls another routine in that library, the called routine must follow the calling routine.

Unless you use the **-e** flag to specify another entry point, the first byte of the first non-null text segment (or the first byte of the data segment if all text segments are null) becomes the entry point of the output file.

The reserved 370 symbols **etext**, **edata** and **end** are set to the first location above the program, the first location above initialized data, and the first location above all data, respectively. You cannot define these symbols.

Because you can use the **ld** command to link modules intended to run on other machines, some of its action depends upon the architecture of the computer system on which you intend to run the module. The **ld** command recognizes that architecture automatically from the input modules and modifies its action accordingly. You can use some of its flags to alter the default behavior of the **ld** command for a particular architecture.

Notes:

1. On AIX/370, it is possible to convert a non-XA load module into an XA load module with the following command:

   ```
   ld -xa -o xamodule module
   ```

2. Use the **-ss** option to force segments aligned on 64-Kb boundaries (the default is 1 Mb). Without using **-ss**, you have a maximum of 5 Mb available to be allocated by the **sbrk** system call and **malloc** subroutine. Using **-ss**, you can have more than 7 Mb available.

3. If you are using **malloc** to allocate large amounts of memory, use the **lmalloc** option to include the alternate library **libmalloc.a**. The default **malloc** subroutine rounds requests to the next higher power of two bytes long, and may allocate more memory than you request. For instance, a request for 4.5 Mb is rounded to 8 Mb and then fails.

Flags
The **ld** command recognizes several flags. Except for **-l** entries, which are really abbreviations for file names, the order in which you specify flags does not affect the way they work. You can specify numeric values in either decimal, octal (with a leading **0**), or hexadecimal (with a leading **0x** or **0X**) format.

-Qfile Writes library dependency information, suitable for inclusion to make, to file **file**.

-qtarg By default the **-Q** option uses the loadmodule name specified by the **-Q** flag as the target name. If specified, **targ** is used for the dependency target in the library dependency file.

The following flags may be specified on AIX PS/2:

-Bnum Makes **num** the starting address for the uninitialized data **(bss)** segment of the output file. The default starting address is the first storage unit after the end of the data segment. Not all architectures support the separation of data and **bss** segments.

-d Defines common storage, even if you have specified the **-r** flag.

-Dnum Makes **num** the starting address for the initialized data segment of the output file. The default starting address begins at location 0 (if the **-i** flag is in effect), at the first storage unit after the end of the text segment, or, if the **-n** flag is in effect, at the next page or segment boundary.

-elabel Makes **label** the entry point of the executable output file.

Note: The **cc** command on AIX/370 prepends an underscore to symbol names, while AIX PS/2 does not.

-Hnum Makes **num** the boundary, usually the page size, to which the text segment must be padded if it has a different protection than does the data segment. Specify this parameter only to override the default value for the given architecture.

-i Assigns text and data segments to separate address spaces in memory, with the text segment read-only (if the architecture supports read-only memory) and shared among all users. The data segment starts at location 0 unless set with the **-D** flag. If the architecture does not support separate instruction and data space, this flag is treated as if it were the **-n** flag. (This option cannot be used with the **-K** flag.) The **-i** flag is useful primarily when you are linking executables (cross-compiling) on different machines.

-K Loads the **a.out** header into the first bytes of the text segment, followed by the text segments from the object modules. This flag causes pages of executable files to be aligned on pages in the filesystem so that they can be demand paged on systems that support paging. This flag provides mapped file support for the text and data segments.

Note: This flag is required for programs to execute on the PS/2 and is normally set by the **cc** command.

-n Makes the text segment read-only--if the architecture supports read-only memory--and shared among all users running the file. The data segment starts at the first segment boundary following the end of the text unless set with the **-D** flag. On architectures which only permit read-only text with separate text and data spaces, the **-n** flag is treated as if it were the **-i** flag. (This option cannot be used with the **-i** flag.)

-Tnum Makes **num** the starting address for the text segment of the output file. If not specified, the text segment begins at location zero.

-Zstr Prefixes with **str** the names specified by the **-l** key. For example, with **-Z/test** and **-lxyz**, the **ld** command looks for the file **/test/lib/llbxyz.a** or, if that file does not exist, **/test/usr/lib/libxyz.a**. The ordinary directories are not searched. This flag is most useful when cross-compiling.

The following flags may be specified on AIX/370:

-f fill Sets the default fill pattern for holes within the output section as well as initialized DSS sections. **fill** is a 2-byte constant.

-lkey Searches the specified library file, where **key** selects the file **libkey.a**. The **ld** command searches for this file in the directory specified by an **-L** flag, and then in the directories **/lib** and **/usr/lib**. It searches library files in the order that you list them on the command line.

-Ldir Looks in **dir** for files specified by **-l key**. If it does not find the file in **dir**, the **ld** command searches the standard directories.

-m Lists on standard output the names of all files and archive members used to create the output file along with a memory map.

-N Makes the text writeable.

-o name Assigns **name** rather than **a.out** to the output file.

-r	Writes relocation bits in the output file so that it can serve as a file parameter in another **ld** call. This flag also prevents common symbols from being assigned final definitions and suppresses the **undefined symbol** diagnostic messages.
-s	Strips the symbol table, line number information, and relocation information from the output. This saves space but impairs the usefulness of the debugger. Using the **strip** command has the same effect. This flag is turned off if there are any undefined symbols.
-u sym	Enters **sym** into the symbol table as an undefined symbol. This is useful when linking from only a library, since initially the symbol table is empty and an unresolved reference is needed to force the linking of the first routine.
-Vnum	Stores **num** in the **a_version** field of the output optional file header; **num** must be in the range 0 to 32767. (This option is **-VS** on the 370.)
-x	Does not enter local symbols in the output symbol table; enters only external and static symbols.
-Y[LU],dir	Looks in **dir** to find libraries. If the **L** flag is specified, the **ld** command replaces the first default directory with **dir**. If the **U** flag is specified, the **ld** command replaces the second default directory with **dir**. (**-L** is the only AIX/370 option.)
-xa	Generates an 370/XA executable module.
-ss	Generates executables aligned on a 64-Kb segment boundary (default is 1 Mb).

Note: The default is for text to be read-only and shared.

Examples

1. To link several object files and produce an **a.out** file to run under the AIX Operating System on a PS/2.

   ```
   ld 0x00400000 -K /lib/crt0.o pgm.o subs1.o subs2.o  -lc /lib/crtn.o
   ```

   A simpler way to accomplish this is to use the **cc** command to link the files as follows:

   ```
   cc  pgm.o  subs1.o  subs2.o
   ```

   Since the **cc** command automatically uses the link options and necessary support libraries, you do not need to specify them on the command line (it gets this information from the configuration file **cc.cfg**). For this reason, you should use **cc** to link files when you are producing programs that run under the AIX Operating System.

2. To specify the name of the output file:

   ```
   cc  -o pgm  pgm.o  subs1.o  subs2.o
   ```

   This command stores the linked output in the file **pgm**.

3. To conditionally link library subroutines:

```
     cc  pgm.o  subs1.o  subs2.o  mylib.a  -ltools
```

   This command links the object modules **pgm.o**, **subs1.o**, and **subs2.o** unconditionally. It then links the subroutines from **mylib.a** that are used by the preceding modules. (This is often called ***conditional linking***.) Then the **ld** command conditionally links subroutines from the library specified by **-ltools**. (This means **/lib/libtools.a**, if it exists. If the **ld** command does not find this file, it looks for **/usr/lib/libtools.a**.)

   Note: Always list libraries and **-l** flags at the end of the **ld** or **cc** command lines.

4. To generate library dependencies to file **depsfile**:

```
     cc -o pgm pgm.c -W1,-Qdepsfile
```

   This generates dependencies in the form:

```
     pgm.o: /lib/libc.a(printf.o)
```

5. To generate library dependencies with a different target name:

```
     cc -o pgm pgm.c -W1,-Qdepsfile -W1,-qpgm
```

   This generates dependencies in the form:

```
     pgm: /lib/libc.a(printf.o)
```

   This facilitates make scripts in generating the correct dependencies.

```
     pgm: $@.c
         cc -o $@ $@.c -W1,-Qdepsfile -W1,-q$@
```

Files

/lib/lib*.a	Libraries.
/usr/lib/lib*.a	Libraries.
a.out	Output file.

Related Information

See the following commands: "ar" in topic 1.1.23, "as" in topic 1.1.25, "cc" in topic 1.1.52, and "shlib2" in topic 1.1.422.

See the **a.out** file in *AIX Operating System Technical Reference*.

See the discussion of the **ld** in *AIX Operating System Programming Tools and Interfaces*.

1.1.227 learn

Purpose
Gives Computer Aided Instruction courses.

Syntax

```
         +----------------+
learn ---¦ +------------+ +---¦
         +-¦ -directory +-+
           ¦ subject    ¦
           ¦ lesson     ¦
           +------------+
```

Note: This command does not have MBCS support.

Description

The **learn** command gives Computer Aided Instruction courses and practice in the use of UNIX, the C Shell, and the Berkeley text editors. To get started simply type **learn** . If you have used the **learn** command before and left your last session without completing a subject, the program will use information in **$HOME/.learnrc** to start you up in the same place you left off. Your first time through, **learn** asks questions to find out what you want to do. Some questions may be bypassed by naming a **subject** and more yet by naming a **lesson**. You may enter the **lesson** as a number that **learn** gave you in a previous session. If you do not know the lesson number, you may enter the **lesson** as a word, and **learn** will look for the first lesson containing it. If the **lesson** is '**-**,' the **learn** command prompts for each lesson; this is useful for debugging.

The **subject**s presently handled are:

    files
    editor
    vi
    morefiles
    macros
    eqn
    C

There are a few special commands. The command **bye** terminates a **learn** session and **where** tells you of your progress, with **where m** telling you more. The command **again** re-displays the text of the lesson and **again lesson** lets you review **lesson**. There is no way for **learn** to tell you the answers it expects in English. However, the command **hint** prints the last part of the lesson script used to evaluate a response, while **hint m** prints the whole lesson script. This is useful for debugging lessons and might possibly give you an idea about what it expects.

The default directory for the script file is **/usr/lib/learn**. The **directory** option allows the user to exercise the script in a non-standard place.

Files

/usr/lib/learn	Subtree for all dependent directories and files.
/usr/tmp/pl**	Playpen directories.
$HOME/.learnrc	Startup information.

Related Information

See the following commands: "csh" in topic 1.1.100 and "ex" in topic 1.1.158.

1.1.228 leave

Purpose

Reminds you when you have to leave.

Syntax

```
         +----------------+
leave ---¦ +-----+¦       +---¦
         +-¦     +- hhmm -+
           +- + -+
```

```
-----------------
¦ Do not put a blank between these items.
```

Description

The **leave** command waits until the specified time, then reminds you that you have to leave. You are reminded 5 minutes and 1 minute before the actual time, at the time, and every minute thereafter. When you log off, the **leave** command exits just before it would have printed the next message.

The time of day is in the form hhmm where hh is a time in hours (on a 12 or 24 hour clock). All times are converted to a 12 hour clock, and assumed to be in the next 12 hours.

If the time is preceded by '+,' the alarm will go off in hours and minutes from the current time.

If no argument is given, the **leave** command prompts with "When do you have to leave?" A reply of newline causes **leave** to exit; otherwise the reply is assumed to be a time. This form is suitable for inclusion in a **login** or **profile** file.

The **leave** command ignores interrupts, quits, and terminates. To get rid of it you should either log off or use "kill -9" giving its process id.

Related Information

See the following command: "calendar" in topic 1.1.48.

1.1.229 lex

Purpose
Generates a C language program that matches patterns for simple lexical analysis of an input stream.

Syntax

```
      +------+   +--------+   +--------+
lex ---¦      +---¦ one of +---¦        +---¦
      +- -t -+   ¦ +----+ ¦   +- file -+
                 +-¦ -n +-+         ¦
                   ¦ -v ¦      +------+
                   +----+
```

Note: This command does not have MBCS support.

Description
The **lex** command reads **file** or standard input, generates a C Language program, and writes it to a file named **lex.yy.c**. This file, **lex.yy.c**, is a compilable C Language program.

The **lex** command uses rules and actions contained in **file** to generate a program, **lex.yy.c**, which can be compiled with the **cc** command. It can then receive input, break the input into the logical pieces defined by the rules in **file**, and run program fragments contained in the actions in **file**. For a more detailed discussion of the **lex** command and its operation, see *AIX Operating System Programming Tools and Interfaces*.

The generated program is a C Language function called **yylex**. The **lex** command stores the **yylex** function in a file named **lex.yy.c**. You can use the **yylex** function alone to recognize simple, one-word input, or you can use it with other C Language programs to perform more difficult input analysis functions. For example, you can use the **lex** command to generate a program that simplifies an input stream before sending it to a parser program generated by the **yacc** command.

The function **yylex** analyzes the input stream using a program structure called a "finite state machine". This structure allows the program to exist in only one state (or condition) at a time. There is a finite number of states allowed. The rules in **file** determine how the program moves from one state to another.

If you do not specify a **file**, the **lex** command reads standard input. It treats multiple files as a single file.

Note: Since the **lex** command uses fixed names for intermediate and output files, you can have only one **lex** command-generated program in a given directory.

Subtopics

1.1.229.1 Input File Format (file)

The input file can contain three sections: definitions, rules, and user subroutines. Each section must be separated from the others by a line containing only the delimiter, **%%**. Format is:

```
definitions
%%
rules
%%
user subroutines
```

The purpose and format of each section are described in the following sections.

Definitions

If you want to use variables in your rules, you must define them in this section. The are put in the left column, and their definitions are put in the right column. For example, if you wanted to define **D** as a numerical digit, you would write;

```
       D   [0-9]
```

You can use a defined variable in the rules section by enclosing the variable name in braces (**{D}**).

In the definitions section, you can set table sizes for the resulting finite state machine. The default sizes are large enough for small programs. You may want to set larger sizes for more complex programs.

%p n	Number of positions is **n** (default 2000)
%n n	Number of states is **n** (default 500)
%e n	Number of parse tree nodes is **n** (default 1000)
%a n	Number of transitions is **n** (default 3000)

If extended characters appear in regular expression strings, you may need to reset the output array size with the **%o** parameter (possibly to array sizes in the range 10,000 to 20,000). This reset reflects the much larger number of characters relative to the number of ASCII characters.

Rules

Once you have defined your terms, you can write the rules section. It contains strings and expressions to be matched in **file** to the **yylex** function, and C commands to execute when a match is made. This section is required, and it must be preceded by the delimiter **%%**, whether you have a definitions section. The **lex** command does not recognize your rules without this delimiter.

In this section, the left column contains the pattern to be recognized in an input file to the **yylex** function. The right column contains the C program fragment that is executed when that pattern is recognized. Patterns can include extended characters with one exception: these characters may not appear in range specifications within character class expressions surrounded by square brackets. The columns are separated by a tab. For example, if you wanted to search files for the keyword **KEY**, you might write:

```
       (KEY) printf("found KEY");
```

If you include this rule in **file**, the lexical analyzer **yylex** matches the pattern **KEY** and runs the **printf** command.

Each pattern may have a corresponding action, a C command to execute when the pattern is matched. Each statement must end with a semicolon. If you use more than one statement in an action, you must enclose all of them in braces. A second delimiter, **%%**, must follow the rules section if you have a user subroutine section.

When the **yylex** function matches a string in the input stream, it copies the matched file to an external character array, **yytext**, before it executes any commands in the rules section.

You can use the following operators to form patterns that you want to match:

x	Matches the character written. The **x** matches the literal character x.
[ ]	Matches any one character in the enclosed range ([.-.]) or the enclosed list ([...]). For example, **[a,b,c,x-z]** matches a,b,c,x,y, or z.
" "	Matches the enclosed character or string even if it is an operator. For example, "**$**" prevents **lex** from interpreting the character **$** as an operator.
\	Acts the same as " ". For example, **$** also prevents the shell from interpreting the character **$** as an operator.
*****	Matches zero or more occurrences of the character immediately preceding it. For example, **x*** matches zero or more repeated **x**'s.
+	Matches one or more occurrences of the character immediately preceding it.
?	Matches either zero or one occurrences of the character immediately preceding it.
^	Matches the character only at the beginning of a line. ^**x** matches an x at the beginning of a line.
[^]	Matches any character but the one following the ^. For example, [^**x**] matches any character but x.
.	Matches any character except the new-line character.
$	Matches the end of a line.
|	Matches either of two characters. For example, **x | y** matches either x or y.
/	Matches one character only when followed by a second character. It reads only the first character into the **yytext** character array. For example, **x/y** matches x when it is followed by y, and reads x into **yytext**.

( ) Matches the pattern in the parentheses. This operator is used for grouping. The parentheses reads the whole pattern into **yytext**. A group in parentheses can be used in place of any single character in any other pattern. For example, **(xyz123)** matches the pattern **xyz123** and reads the whole string into **yytext**.

{} Matches the character as you defined it in the definitions section. For example, you defined **D** to be numerical digits, **{D}** matches all numerical digits.

{m,n} Matches **m** to **n** occurrences of the character. For example, **x{2,4}** matches 2, 3, or 4 occurrences of **x**.

If a line begins with only a blank, the **lex** command copies the line to the output file, **lex.yy.c**. If the line is in the declarations section of **file**, the **lex** copies the line to the declarations section of **lex.yy.c**. If the line is in the rules section, the **lex** command copies the line to the program code section of **lex.yy.c**.

User

The **lex** library has three subroutines defined as macros, and which you can use in the rules.

input( ) Reads a character from **yyin**.

unput( ) Replaces a character after it has been read.

output( ) Writes an output character to **yyout**.

You can override these three macros by writing your own code for these routines in the user subroutines section. But if you write your own, you must undefine these macros in the definition section as follows:

```
%{
#undef input
#undef unput
#undef output
}%
```

There is no **main( )** in the file **lex.yy.c** because the **lex** library contains the **main( )** that calls the lexical analyzer **yylex**. Therefore, if you do not include **main( )** in the user subroutines section, when you compile the file **lex.yy.c**, you must enter **cc -ll lex.yy.c**, where **ll** calls the **lex** library.

External names generated by the **lex** command all begin with the prefix **yy**, as in **yyin**, **yyout**, **yylex**, and **yytext**.

Flags

-n Suppresses the statistics summary. When you set your own table sizes for the finite state machine (see page 1.1.229.1), the **lex** command automatically produces this summary if you do not select this flag.

-t Writes the file **lex.yy.c** to standard output instead of to a file.

-v Provides a one-line summary of the generated finite-state-machine statistics.

Files

/usr/lib/libl.a	Run-time library.

Related Information

See the following commmand: "yacc" in topic 1.1.547.

See the description of the **lex** command and "Programming for an MBCS Environment" in *AIX Operating System Programming Tools and Interfaces*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.230 li, di

Purpose
Lists the contents of a directory.

Syntax

```
      +-- -Oabcdfpx --+   +---------------- -R1 ----------------+
li ---¦     ¦ +-----+ +---¦ +--- -R ---+   +--------+   +-----+ +---
      +- -O --¦ a f +-+   +-¦          +---¦ one of +---¦     +-+
              ¦ b p ¦¦     +- -R num -+    ¦ +---+  ¦   +- q -+
            ¦ ¦ c x ¦¦                   +-¦ a +--+
            ¦ ¦ d   ¦¦                     ¦ p ¦
            ¦ +-----+¦                     +---+
            +--------+

    +--------- -Sn ---------+   +----------------+   +------- -In -------+
   ---¦ +- -S --+   +-------+ +---¦ +------------+ +---¦ +----+ ¦ +-----+  +---
    +-¦       +---¦ a n x +-+   +-¦ -a -n -x   +-+   +-¦ -I +---¦ a m +--+
      +- -Sr -+   ¦ c s   ¦¦      ¦ -d -s -l   ¦¦      ¦ -E ¦   ¦ c n ¦¦¦
                ¦ ¦ m u   ¦¦     ¦¦ -f -v -num ¦¦     ¦+----+ ¦ ¦ o p ¦¦¦
                ¦ +-------+¦     ¦¦ -k         ¦¦     ¦       ¦ ¦ g h ¦¦¦
                +----------+     ¦+------------+¦     ¦       ¦ ¦ i s ¦¦¦
                                 +--------------+     ¦       ¦ ¦ l u ¦¦¦
                                                      ¦       ¦ +-----+¦¦
                                                      ¦       +--------+¦
                                                      +-----------------+

    +------ . ------+
   ---¦ +-----------+ +---¦
    +-¦ file        +-+
      ¦ directory ¦¦
     ¦+-----------+¦
     +-------------+

      +------ . ------+
di ---¦ +-----------+ +---¦
      +-¦ file        +-+
        ¦ directory ¦
        +-----------+

----------------
¦ Do not put a blank between these items.
```

Description
The **li** command lists the contents of each named **directory** or archive **file** on standard output. For each non-archived **file** named, the **li** command displays the file name and any information requested. If you do not specify a **file** or **directory**, the **li** command lists the current directory.

By default, the **li** command sorts the output alphabetically and lists it in multiple columns. The collating sequence is determined by the **LANG** and **LC_COLLATE** environment variables. It displays control characters in file names in expanded form (for example, **^D**, **\177**, and so on). When you specify more than one file or directory, the **li** command sorts them appropriately, but files are listed before directories and their contents. When the date and time appear, the **LANG** and **LC_TIME** environment variables control their format.

The **di** command is equivalent to **li -Ialmops**.

If a file is a symbolic link, the **li** command lists the file name, followed by the string **->** and then the contents of the symbolic link. If a file is a hidden directory, the **li** command lists the directory name, followed by the string **@/** and then the name of the matched hidden directory component.

Subtopics

1.1.230.1 Permissions Field

The permissions field is displayed with the **-Ip** flag. This field contains 11 characters. The first character can be one of the following:

d	The entry is a directory.
b	The entry is a block-type special file.
c	The entry is a character-type special file.
h	The entry is a hidden directory, and either the file argument was escaped with a trailing @ or there was not a component that could be matched in a hidden directory.
l	The entry is a symbolic link, and the symbolic link did not point to an existing file.
p	The entry is a pipe (FIFO).
-	The entry is an ordinary file.

The next nine characters are interpreted as three sets of three bits each. The first set refers to owner permissions, the next set to permissions for others in the same group, and the last set to all others. Each of the three characters within each set indicates, respectively, permission to read, write, or execute the file. For a directory, execute permission is interpreted as permission to search the directory for a file. These permissions are indicated as follows:

r	The file is readable.
w	The file is writable.
x	The file is executable.
-	The corresponding permission is not granted.

The group-execute permission is given as **s** if the file has set-group-ID mode. The user-execute permission character is given as **s** if the file has set-user-ID mode. (For a discussion of these modes, see "chmod" in topic 1.1.67.)

The last character of the field is usually blank, but a **t** is displayed if bit 1000 of the mode is on. (See the **chmod** system call in *AIX Operating System Technical Reference* for the current meaning of this mode.)

Note: Some combinations of flags do not work well together. For example, **li -vRa** looks unusual, and **li -RSx** and **li -Sx *** are both nearly unintelligible if there are subdirectories contained in the current directory, due to confusion about what level is being listed.

Flags

Flags are grouped into five classes: fields (the **I** or **E** flag), restrictions (the **O** flag), recursion (the **R** flag), sort orders (the **S** flag), and miscellaneous. The following flags modify the action of **li**:

-I [hiplogcsmaun]
-E [hiplogcsmaun]
Requests the inclusion (**-I**) or exclusion (the **-E**) of certain fields. These fields are selected by the flags in the subset **hiplogcsmaun**.

-I This flag includes the selected fields in the order in which they appear in the argument list; the **-E** flag excludes these fields. For example, **-l -Ep** excludes the protections **(P)** field, while **-Ep -l** includes it, since **-l** (the equivalent of **-Icglmop**) follows **-Ep**.

The only field included by default is the name (**n**) field. If you include any other fields, **li** lists the output in single-column rather than multiple-column format. The **li** command lists the following fields in the following order:

h	Headers
i	I-number
p	Protections
l	Link count
o	Owner (name or UID)
g	Group (name or GID)
c	Character count
s	Size in blocks
m	Modified time
a	Accessed time
u	Updated (inode modified) time
n	Name

If the file is a special file, the size (**s**) field contains the major and minor device numbers. If you select the **c** (character count) or **s** (size in blocks) flags, the **li** command writes the total number of blocks for each directory and the grand total when appropriate.

-O [abcdfpx]
Requests that the listing be restricted to files of certain types. These types are selected from the subset **abcdfpx**. The possible types are:

a	Archives
b	Block devices
c	Character devices
d	Directories
f	Files (normal, not special)
p	Pipes (FIFOs)
x	Executable files (any file with execute permission)

-R[num]apq
Lists recursively to **num** levels deep. The default depth is infinite. This flag normally displays a single column, with a two-column indentation for each level of the directory structure. When the **li** command reaches a directory with no subdirectories, it lists the contents of that directory in multiple-column form. Specifying either **-Ra** or **-Rp** suppresses the indentation and multiple-column display. These flags display either the full (**-Ra**) or relative (**-Rp**) path names of each file found. The **Rq** flag also lists the contents of archive files. (See the archive file format in *AIX Operating System Technical Reference.*)

li command
Reaches a directory with no subdirectories, it lists the contents of that directory in multiple-column form. Specifying either **-Ra** or **-Rp** suppresses the indentation and multiple-column display. These flags display either the full (**-Ra**) or relative (**-Rp**) path names of each file found. The **-Rq** flag also lists the contents of archive files. (See the archive file format in *AIX Operating System Technical Reference*.)

-S [acmnrsux]
Describes the order in which the listing is to be displayed. The default order is by name (**n**). The **-Sx** flag specifies no sorting. Choosing a flag from the subset **acmnsu** selects which of the following

fields, the listing is sorted by.

a	Accessed time, latest first
c	Character count, largest first
m	Modified time, latest first
n	Name
s	Size (same as character count)
u	Updated time, latest first

If you include the **r** flag with any of these flags, the **li** command reverses the order of the sort.

The miscellaneous flags are:

-a Lists all entries, including those beginning with **.** (dot).

-d Lists only the name, not the contents, of directories.

-f Forces the **li** command to interpret each **file** as a directory and to list the name found in each slot. All flags requiring information not found in directory entries are turned off and the **-a** flag is turned on. Names are listed in the order that they appear in the directory.

-l Uses a listing that is equivalent to **li -Icglmop** (the long form listing). That is, it lists the permission code, link count, owner, group, character count, time of last modification, and name of each file.

-n Inhibits the interpretation of control characters in file names. This flag is useful for generating lists of file names for program input or for editing into per-file commands.

-s Provides a listing similar to that of the **-v** flag, except that the distinguishing marks for file types do not affect sorting (a sortable verbose list). Subdirectories appear in the listing as **name**/, files with execute permission as **name*** , and special files as **name**?.

-v Lists files in a way that visually differentiates file types (a verbose visual listing). With this flag, the **li** command lists subdirectories as [**name**], files with execute permission as <**name**>, and special files as ***name***. This differentiation occurs before the **-S** sort. Thus, different types of files are sorted into different parts of the listing.

-x Displays every available field except headers (an extended form listing). This is equivalent to specifying **li -Iacglimopsu**.

-num Lists with a maximum of **num** columns. If **num** is unreasonable, the **li** command picks its own **num**. This flag can be used as in **li -1** to make shell files or in **li-Io9** to force the **li** command to display its output in multiple columns. A number appearing in any flag argument is assumed to be the number of columns unless it follows the **-R** flag.

Examples

1. To list the files in the current directory in alphabetical order:

```
li
```

2. To list all files in the current directory, including those with names

beginning with a **.** (dot):

```
li -a
```

3. To display detailed information:

   ```
   li -l chap1 .profile
   ```

   This command displays a long listing with detailed information about **chap1** and **.profile**. It lists all the information that you probably need to see. However, the **li** command can supply even more information with the **-x** flag.

4. To display detailed information about a directory:

   ```
   li -d -l . manual manual/chap1
   ```

   This command displays a long listing for the directories **.** and **manual**, and for the file **manual/chap1**. Without the **-d** flag, this command would list the files in **.** and **manual** instead of the detailed information about the directories themselves.

5. To list the files in order of modification time:

   ```
   li -Sm -l
   ```

   This command displays a long listing of the files that were modified most recently, followed by the older files.

6. To include extra information in the listing:

   ```
   li -Ichil
   ```

   In addition to the file name, this command lists the character count (**-Ic**), i-number (**-Ii**), and link count (**-Il**) for each file in the current directory. The **-Ih** flag tells the **li** command to write a heading at the top of each column of information.

7. To list the contents of each directory in a tree:

   ```
   li -R manual
   ```

   This command lists the names in each subdirectory of the tree that starts with **manual**.

Files

/etc/passwd	Contains user names for **li -Io**.
/etc/group	Contains group names for **li -Ig**.

Related Information

See the following commands: "ls, lf, lr" in topic 1.1.252.

See the **chmod** system call and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.231 line

Purpose
Reads one line from standard input.

Syntax

```
line ---¦
```

Description
The **line** command copies one line from standard input and writes it to standard output. It returns an exit value of 1 on an end-of-file and always writes at least a new-line character. Use this command within a shell command file to read from your work station.

Example

To read a line from the keyboard and append it to a file:

```
  echo 'Enter comments for the log: \c'
  line >>log
```

This shell procedure displays the message:

```
  Enter comments for the log:
```

then reads a line of text from the work station keyboard and adds it to the end of **log**. The **: \c** command displays a colon prompt. See "echo" in topic 1.1.146 for information about the **\c** escape sequence.

Related Information

See the following commands: "sh, Rsh" in topic 1.1.420.

See the **read** system call in *AIX Operating System Technical Reference*.

1.1.232 link, unlink

Purpose
Performs a link or unlink system call.

Syntax

```
link -- file1 -- file2 --¦

unlink -- file --¦
```

Description
The **link** and **unlink** commands perform the corresponding system calls of the same name on the specified file, abandoning all error checking. These commands can be run only by a user operating with superuser authority (see "su" in topic 1.1.449). You should be familiar with the **link** and **unlink** system calls described in *AIX Operating System Technical Reference*.

The **link** and **unlink** commands do not issue error messages when the associated system call fails; you must check the exit value to determine if the command completed normally. Each command returns a 0 if it succeeds, a 1 if you specify too few or too many parameters, or a 2 if its system call fails.

Warning: The **link** and **unlink** commands allow the superuser to deal with unusual problems, such as moving an entire directory to a different part of the directory tree. They also permit you to create directories that cannot be reached or escaped from. Be careful to preserve directory structure. To preserve directory structure observe the following rules:

Be certain every directory has a **.** (dot) link to itself.
Be certain every directory has a **..** (dot dot) link to its parent directory.
Be certain every directory has no more than one link to it from above
Be certain every directory is accessible from the root of its fil system.

Note: If the . (dot) entry has been destroyed and the **fsck** command is unable to repair it (a rare occurrence), you can use the **link** command to restore the . (dot) entry of the damaged directory with the command **link dir dir/.**, where **dir** is the name of the damaged directory. However use this command only as a last resort; that is, when the directory is destroyed and **fsck** is unable to fix it.

Related Information
See the following commands: "ln" in topic 1.1.234 and "fsck, dfsck" in topic 1.1.177.

See the **link** and **unlink** system calls in *AIX Operating System Technical Reference*.

1.1.233 lint

Purpose
Checks C programs for potential problems.

Syntax

```
        +-----------------+
lint ---¦ +-------------+ +--- file ---¦
        +-¦ -a -lkey -u +-+           ¦
          ¦ -b -n -c -v ¦   +--------+
         ¦¦ -h -p    -x ¦¦
         ¦¦ -Nnnum   -z ¦¦
         ¦¦ -o lib      ¦¦
         ¦¦ -T file     ¦¦
         ¦+-------------+¦
         +---------------+
```

Note: This command does not have MBCS support.

Description
The **lint** program checks C-language source code for coding and syntax errors and for inefficient or nonportable code. You can use this program to

Identify source code and library incompatibilit
Enforce type checking rules more strictly than the compile
Identify potential problems with variable
Identify potential problems with function
Identify problems with flow contro
Identify legal constructions that may produce errors or be inefficien
Identify possibly nonportable code

The **lint** command assumes that **file** names ending in **.c** are C Language source files. It assumes that those ending in **.ln** are the result of an earlier running of **lint** with either the **-c** or the **-o** flag used. These **.ln** files are analogous to the **.o** (object) files produced by the **cc** command when given a **.c** file as input. The **lint** command warns you about files with other suffixes and ignores them.

The **lint** command takes all the **.c** and **.ln** files and the libraries specified by **-l** flags and processes them in the order that they appear on the command line. By default, it adds the standard **lint** library (**llib-lc.ln**) to the end of the list of files. However, when you select the **-p** flag, **lint** uses the portable library **llib-port.ln**. By default, the second pass of **lint** checks this list of files for mutual compatibility; however, if you specify the **-c** flag, **lint** ignores the **.ln** and **lib-lx** files.

The **-c** and **-o** flags allow for incremental use of the **lint** command on a set of C Language source files. Generally, you use **lint** once for each source file with the **-c** flag. Each of these runs produces a **.ln** file that corresponds to the **.c** file and writes all messages that pertain to that source file only. After you run all source files separately through **lint**, you run the command again, without the **-c** flag, listing all the **.ln** files with the needed **-l** arguments. This writes all inter-file inconsistencies. This procedure works well with the **make** command, allowing it to run **lint** on only those source files that have been modified since the last time that set of source files was checked.

The following comments in a C source program change the way that the **lint** command operates when checking the source program:

/*NOTREACHED*/ Suppresses comments about unreachable code.

/*VARARGSn*/ Suppresses checking the following function declaration for varying numbers of arguments but does check the data type of the first **n** arguments. If you do not include a value for **n**, **lint** checks no arguments (**n**=0).

/*ARGSUSED*/ Turns on the **-v** flag for the next function.

/*LINTLIBRARY*/ If you place this comment at the beginning of a file, **lint** does not identify unused functions in the file.

The **lint** command first writes messages about each source file as it processes the file. It collects messages about included files and writes those after it has gone through all the source files. Finally, if you have not specified the **-c** flag, it collects information gathered from all input files and checks it for consistency. At this point, if it is not clear whether a message stems from a given source file or from one of its included files, **lint** displays the source file name followed by a question mark.

Flags

-a Suppresses messages about assignments of long values to variables that are not long.

-b Suppresses messages about unreachable break statements.

-h Does not try to detect bugs, improve style, or reduce waste.

-c Causes **lint** to produce a **.ln** file for every **.c** file on the command line. These **.ln** files are the product of the first pass of **lint** only and are not checked for inter-function compatibility.

-lkey Includes the additional **lint** library **llib-lkey.ln**. You can include a **lint** version of the math library **llib-lm.ln** by specifying **-lm** on the command line or **llib-ldos.ln** by specifying **-ldos** on the command line. Use this flag to include local **lint** libraries when checking files that are part of a project having a large number of files. This flag does not prevent **lint** from using the **llib-lc.ln** library.

-n Does not check for compatibility with either the standard or the portable **lint** libraries.

-Nnnum Increases the size of the symbol table. The default size is 1500.

-o lib Causes **lint** to create a lint library with the name **llib-llib.ln**. The **-c** flag nullifies any use of the **-o** flag. The **lint** library produced is the input that is given to the second pass of **lint**. The **-o** flag simply causes this file to be saved in the named **lint** library. To produce a

llib-llib.ln without extraneous messages, use the **-x** flag. The **-v** flag is useful if the source files for the lint library are just external interfaces (for example, the way the file **llib-lc** is written). These flag settings are also available through the use of lint comment lines.

-p	Checks for portability to other C dialects.
-T file	Use file as intermediate file.
-u	Suppresses messages about functions and external variables that are either used and not defined or defined and not used. Use this flag to run **lint** on a subset of files of a larger program.
-v	Suppresses messages about function parameters that are not used.
-x	Suppresses messages about variables that have external declarations but are never used.
-z	Suppresses messages about undeclared structures.

In addition, the **lint** command recognizes the following flags of the **cpp** command (macro preprocessor):

-Dname[=def]	Defines the **name**, as if by a **#define** directive. The default **def** is **1**.
-Idir	Adds **dir** to the list of directories in which **lint** searches for **#include** files.
-Uname	Removes any initial definition of **name**, where **name** is a reserved symbol that is predefined by the particular preprocessor.

Examples

1. To check a C program for errors:

   ```
   lint  program.c
   ```

2. To suppress some of the messages:

   ```
   lint  -v  -x  program.c
   ```

   This command checks **program.c**, but does not display error messages about unused function parameters (**-v**) or unused externals (**-x**).

3. To check the program against an additional lint library:

   ```
   lint  -lsubs  program.c
   ```

   This command checks **program.c** against both the standard **lint** library **/usr/lib/llib-lc.ln** and **/usr/lib/llib-lsubs.ln**.

4. To check against the portable library and an additional library:

   ```
   lint -lsubs -p program.c
   ```

This command checks **program.c** against both the portable lint library **/usr/lib/llib-port.ln** and **/usr/lib/llib-lsubs.ln**.

5. To check against a nonstandard library only:

```
lint -lsubs -n program.c
```

   This command checks **program.c** against only **/usr/lib/llib-lsubs.ln**.

Files

/usr/lib/lint[12]	Programs.
/usr/lib/llib-lc.ln	Declarations for standard functions (binary format).
/usr/lib/llib-lc	Declarations for standard functions (source).
/usr/lib/llib-port.ln	Declarations for portable functions (binary format).
/usr/lib/llib-port	Declarations for portable functions (source).
/usr/lib/llib-lm.ln	Declarations for standard math functions (binary format).
/usr/lib/llib-lm	Declarations for standard math functions (source).
/usr/lib/llib-ldos.ln	Declarations for DOS Services functions (binary format).
/usr/lib/llib-ldos	Declarations for DOS Services functions (source).
/usr/tmp/*lint*	Temporary files.

Related Information

See the following command: "cc" in topic 1.1.52.

See "Checking C Programs" in *AIX Operating System Programming Tools and Interfaces*.

See "Overview of International Character Support" in *Managing the AIX Operating System*.

1.1.234 ln

Purpose
Links files.

Syntax

```
              +-- file --- directory --+
ln ----------¦         ¦              +---¦
    +- -s -+ ¦ +-------+              ¦
              +--- file --- newname ---+
```

Description
The **ln** command links **file** to **newname** (in the current directory), or to the same name (**file**) in another existing **directory**.

There are two types of links supported by the **ln** command: hard links and symbolic links.

If you are linking a file to a new name, you can list only one **file**. If you are linking to a **directory**, you can list more than one file. If **file** is a symbolic link, **ln** creates a second symbolic link called **newname** that points to the same file or directory as **file**.

Notes:

1. You cannot create hard links between files across file systems.

2. When creating a symbolic link, **file** need not exist.

3. When creating a symbolic link, **file** may name a directory.

The **ln** command does not link a file to a hidden directory. Hidden directory components must be named explicitly. For example, the following is a correct specification:

```
  ln hidden.file@/i386 new.file
```

Flag

-s   Creates a symbolic link rather than a hard link.

Examples

1. To create another name (also called an alias) for a file:

    ```
    ln  chap1  intro
    ```

    This command links **chap1** to the new name **intro**. If **intro** does not already exist, the file name is created. If **intro** does exist, the file is replaced by a link to **chap1**. Now **chap1** and **intro** are two file names that refer to the same file. Any changes made to one also appear in the other. If one name is deleted with the **del** or **rm** command, the file is not actually deleted, but remains under the other name.

2. To link a file to the same name in another directory:

    ```
    ln  index   manual
    ```

This command links **index** to the new name **manual/index**.

Note that **intro** in the example above, (creating an alias) "Chap 1" is the name of a file, while in this example **manual** is a directory that already exists.

3. To link several files to names in another directory:

   ```
   ln  chap2  tom/chap3  /u/manual
   ```

   This command links **chap2** to the new name **/u/manual/chap2** and **tom/chap3** to **/u/manual/chap3**.

4. To use pattern-matching characters:

   ```
   ln  manual/*  .
   ```

   This command links all files in the directory **manual** into the current directory (.), giving them the same names they have in **manual**. You must type a space between the asterisk and the period.

5. To create a symbolic link:

   ```
   ln -s /bin/who who
   ```

   This command creates a symbolic link in the current directory **who** that points to the file directory **/bin/who**.

6. To create a symbolic link into the <LOCAL> filesystem:

   ```
   ln -s "<LOCAL>/utmp" /etc/utmp
   ```

   This command creates a symbolic link **/etc/utmp**, which points at a different file **utmp**, on each site in a TCF cluster. The use of quotes is required because **<** and **>** have special meanings to the shells.

Related Information

See the following commands: "rm, delete" in topic 1.1.375, "mv, move" in topic 1.1.282, and "cp, copy" in topic 1.1.91.

See the **chmod** and **link** system calls in *AIX Operating System Technical Reference*.

1.1.235 lnetstat

Purpose
Provides network statistics.

Syntax

```
            +-----------------+   +------------- -p -----------------+
lnetstat ---¦                 +---¦ +------+  +--------------------+ +---¦
            +- -s --- site ---+   +-+- -S -+--¦      +------------+ +-+
                            ¦       +- -R -+  +- -t -¦            +-+
                   +--------+                        +- interval -+
```

Description
The **lnetstat** command provides statistics about TCF message traffic. If no arguments are specified, **lnetstat** displays network information about all sites in the current cluster except the local site. The local site network message statistics are ignored if the site number is specified or when all site information is printed. This information includes site number, name, and how long the site has been up.

If TCP/IP message traffic is required, use the **netstat** command.

The **site** parameter values define the site to which network messages are routed, the timeout value for transmissions, the number of retries (re-transmissions) for a network message, the acknowledgement windows for normal and special messages, and whether check summing is on (1) or off (0).

Site status information describes the status of network messages coming from and going to a specific site, whether or not the site is accepting messages, messages that have been acknowledged or dropped, re-transmitted messages, and time in contact with foreign sites.

Message statistics for each site include statistics for the number of incoming and outgoing messages and their response time for various classes of network messages handled by the site.

Flags

-p	Prints the site parameters; these are values used by the kernel in handling all traffic to and from a given site.
-R	Displays re-transmission counts to all sites (default) or to those specified with the **-s** flag.
-s site	Restricts display to the specified sites. Sites can be specified either by name or number, and should be separated with a space. If the local site is specified, it is silently ignored.
-S	Summarizes; message counts and times are summed over all sites (default) or those specified with the **-s** flag.
-t interval	Displays new traffic once every **interval** seconds. If **interval** is not specified, it defaults to 10.

Examples

1. The command

   ```
   lnetstat -p -s myoko
   ```

   displays the following statistics:

   ```
   site   5    myoko
   route:   5       timeout:    4       retries:        3
   window:  6       s_window:   2       checksum:       1
   ```

2. The command

   ```
   lnetstat -s myoko
   ```

   displays the following statistics:

   ```
   site  5 myoko                      connected for 0d 01h 03m 58s
   SITE STAT               MSG_STAT        msgin    msgout       rsptime
   =====================================================================
   chan_stat       open    misc           164123    127103          12.5
   site_stat         up    nops            85027     89789           0.2
   in_pkts       763636    read            77238     81979          10.5
   out_pkts      761604    writes             51     10117          19.1
   dups             352    open, stat      78942    122325          16.0
   retrans          585    rcd ops           448       132           8.3
   msgdrops           2    synch               0         0           -
   bufdrops           0    proc crt/dst       13        11         116.3
   xmterrs            0    proc sig/sts       18         8          10.0
   crcerrs            0    cls/cmt/trunc   78985     81124         923.8
   badmsg             0    read rsp        94410     77517           9.7
   badsite            2    other rsp      174095    168027           7.7
   badseq           103    css-ss              4        24           5.0
   ack_blocks       382    topchg/update      18        25       25548.8
   sack_blocks        3    netwrk maint     9934     10062           6.2
   chan_opens        28    user-1              0         0           -
   channel          409    user-2              0         0           -
                           user-3              0         0           -
                           user-4              0         0           -
   untimed msgs      18    user-5              0         0           -
   ```

3. The command

   ```
   lnetstat -S -s myoko
   ```

   displays the following statistics:

   ```
   Summary of site(s) 5
   SITE STAT               MSG_STAT        msgin    msgout       rsptime
   =====================================================================
   chan_stat    xxxxxxx    misc           164126    127106          12.5
   site_stat    xxxxxxx    nops            85028     89790           0.2
   in_pkts       763646    read            77239     81980          10.5
   out_pkts      761614    writes             51     10117          19.1
   dups             352    open, stat      78943    122326          16.0
   retrans          585    rcd ops           448       132           8.3
   msgdrops           2    synch               0         0           -
   bufdrops           0    proc crt/dst       13        11         116.3
   xmterrs            0    proc sig/sts       18         8          10.0
   crcerrs            0    cls/cmt/trunc   78986     81125         923.8
   badmsg             0    read rsp        94411     77518           9.7
   ```

```
  badsite           2    other rsp       174097   168029            7.7
  badseq          103    css-ss               4       24            5.0
  ack_blocks      382    topchg/update       18       25        25548.8
  sack_blocks       3    netwrk maint      9934    10062            6.2
  chan_opens       28    user-1               0        0            -
  channel     xxxxxxx    user-2               0        0            -
                         user-3               0        0            -
                         user-4               0        0            -
  untimed msgs      0    user-5               0        0            -
```

4. The following is an example of **inetstat** information for a site called **sonic**:

```
  site  2 sonic                     connected for 13d 10h 49m 46s
  SITE STAT              MSG_STAT        msgin    msgout        rsptime
  =====================================================================
  chan_stat      open    misc             3735      1807             21
  site_stat        up    nops            13371     44918              1
  in_pkts      115059    read              309      1873             84
  out_pkts     144272    writes             35      1152             24
  dups              0    open, stat        724      6762             47
  retrans           8    rcd ops          2775     32764             16
  msgdrops          0    synch               0         0              0
  bufdrops          0    proc crt/dst      421       462             26
  xmterrs           0    proc sig/sts      308       710            117
  crcerrs           0    cls/cmt/trunc     429      3106            157
  badmsg            0    read rsp         4510       309             85
  badsite           0    other rsp       77672     10655             23
  badseg            0    css-ss              0       606             13
  ack_blocks        0    topchg/update     212       219             24
  sack_blocks       0    netwrk maint     9580     38731             13
  chan_opens        1
  channel         194


  untimed msgs      4
```

The SITE STAT column describes the site status along with associated information in the next column. The MSG_STAT column describes the message status type. msgin gives the number of messages in, msgout gives the number of messages out, rsptime gives the response time.

Each site has a chart of SITE STAT and MSG_STAT information, which indicates the following:

SITE STAT

chan_stat	Channel status (open, close, attempt).
site_stat	Status of other site (up or down).
in_pkts	Number of TCF messages received.
out_pkts	Number of TCF messages sent.
dups	Number of TCF messages received (three are counted as two, one worked).
retrans	Number of TCF messages sent (three are counted as two, one worked).
msgdrops	When TCF message was received there were no free netmsgs available, so the TCF msg was dropped.
bufdrops	When long message was received, there were no free filesystem buffers, so TCF msg was dropped.
xmterrs	Transmit errors; no TCF msg may have been transmitted;

	error in lower-level protocol.
crcerrs	Checksum error in TCF header.
badmsg	Malformed message (too short, consistency check bad).
badsite	TCF site number in TCF msg was not the same as the TCF site that received it for previous **n** messages had yet been received*.
badseq	Bad sequence, not in order.
ack_blocks	Number of times message ready to be sent but could not be sent, as no acknowledgement.
sack_blocks	Number of times special message (topchange) ready, not sent.
chan_opens	Number of times channel established or attempted.
channel	Channel number (incremented after it goes down).

MSG_STAT

misc	Messages not included in any other category.
nops	No operations, as it carried no information (only an acknowledgement).
read	Number of filesystem reads.
write	Number of filesystem writes.
open, stat	Number of open and stat calls.
rcd, ops	Remote character device terminal operations reads, writes, I/O controls.
proc crt/dat	Process creation and destruction messages.
proc sig/sts	Process signal and status message.
cls/cmt/truc	Filesystem (not remote terminal) close, commit, or truncated.
read rsp	Read response (on filesystem, not on remote terminal).
other rsp	Responses other than read for filesystem messages.
css-ss	Controlling storage site and storage site message.
topchg/update	Topology change related.
netwrk maint	Channel opens, probes, responds to IP level of TCF protocol.

* where **n** is the window size

Note the distinction between filesystem operations and remote terminal operations. The key to understanding how to use the **Inetstat** parameters is to watch for patterns in network traffic every day and then be aware of the correlation between the various parameters and the changes that the cluster is undergoing. It is also useful to check the TCP/IP traffic (using **netstat**) and correlate TCP/IP changes with what is going on with TCF changes. For example, if **drops** are higher (say 500 in one hour, where normally you have 100 per day or so), this indicates insufficient resources and therefore more **netmsgs** need to be configured in **/etc/system**. (If **netparams** are not listed, then check the default **/etc/master** file for this number.)

If **dups** and **drops** go up simultaneously, then one side is losing messages. If **dups** increase, but **drops** do not the site is too slow and is losing messages.

If **retrans** is high, the site is sending lots of messages out. This could be caused by no buffers, hardware being wedged, or a cable problem.

If **dups** and **retrans** constitute a large percentage of **inpkts** and **out_pkts**, then further analysis is warranted to find out why transmissions are not working.

High **crcerrs** is due to a checksum problem, which is usually a hardware problem.

High **badmsg** counts are often due to a hardware or kernel problem. Look at other indicators for more information.

badseq, **dups**, and **retrans** are all related, since bad messages give out-of-sequence errors and numbers.

Files
/usr/include/sys/netstat.h
/usr/include/sys/netctrl.h

Related Information

In this book, see the following command: "crash" in topic 1.1.96.

See the *AIX/370 Diagnosis Guide*.

1.1.236 loads

Purpose
Displays the load average of each site in the network.

Syntax

```
         +----------------+
loads ---¦ +------------+ +---¦
         +-¦ sitename   +-+
           ¦ sitenumber ¦
          ¦¦ sitetype   ¦¦
          ¦+------------+¦
          +--------------+
```

Description
For each site in the current partition, the **loads** command prints out the number of users, the 1 minute, 5 minute, and 15 minute load averages, and the length of time the site has been up. If site names, site numbers or site-type names (see /etc/site) are specified as arguments, the output from **loads** is further limited to the indicated sites. Normally, the **loads** command reads its information from files generated by various daemons.

Flags

sitename	Displays statistics for sitename only.
sitenumber	Displays statistics for sitenumber only.
sitetype	Displays statistics for site type only.

The possible **sitetype** values are:

i386	An AIX PS/2 system.
i370	An AIX/370 system.
xa370	An XA AIX/370 system.
s370	A non-XA AIX/370 system.

Files

/etc/local_loads	Loads information for local site.
/etc/loads	Load information for all sites.
/etc/loadserver	Daemon which updates **/etc/local_loads, and /etc/loads**.

Diagnostics
The **loads** command silently ignores any site which is down.

Related Information
See the following command: "fast, fastsite" in topic 1.1.161.

1.1.237 loadserver

Purpose
Local site load daemon.

Syntax

loadserver ---¦

Description

The **loadserver** daemon is automatically started by the **/etc/inittab** file when the system reaches a run-level of 4. The **loadserver** daemon is not normally started from the command line. See **init** in the *AIX Operating System Commands Reference* and **inittab** in the *AIX Operating System Technical Reference* for more information.

The **loadserver** command collects the local load information and places it in the file **/etc/local_loads**. The **/etc/local_loads** file is really a symbolic link into the local file system of the site. The **loadserver** command extracts the load information and user information from the kernel and the **/etc/utmp** files of the local site. If the **loadserver** command detects that the kernel load module has been modified lately, it researches the load module symbol table for the addresses of the kernel variables used to extract the load information.

On one site in a TCF cluster, the site with a writable local copy of **/etc/loads**, **loadserver** collects data from the **<LOCAL>/local_loads** file on each site and places this information in the file **/etc/loads**. This file is then consulted by the **fast**, **fastsite** and **loads** command.

Files

/unix	Kernel load module.
/etc/local_loads	Local load information.
/etc/loads	Global load information.

Related Information

See the following commands: "fast, fastsite" in topic 1.1.161 and "loads" in topic 1.1.236.

1.1.238 locator

Purpose
Controls the sample rate of the locator.

Syntax

locator -- -rrate --¦

Note: This command does not have MBCS support.

Description
The **locator** command sets the per-second, **rate** at which the system checks the cursor position controlled by the mouse. You can specify any of the following for **rate**: **10**, **20**, **40**, **60**, **80**, or **100**. At system startup, this rate is set at **60**.

Note: You can run the **locator** command only from the system console (PS/2 only).

Flag

-rrate Sets the sampling **rate** to the specified value.

Example
To set the locator rate to **40**

```
  locator  -r40
```

1.1.239 lock

Purpose
Requests a password from the user and verifies.

Syntax

```
        +-----------+
lock ---¦           +---¦
        +- -number -+
```

Description

The **lock** command requests a password from the user, reads it again for verification and then it normally will not relinquish the terminal until the password is repeated. There are three other conditions under it will terminate. It will time out after some interval. It may be killed by somebody with the appropriate permission. On being given the root password, the lock is broken, but the program logs the initial user out.

The default time limit for termination is 15 minutes but it may be changed with the **-number** option, where **number** is the time limit in minutes.

1.1.240 logger

Purpose
Provides a program interface to the **syslog** system log module.

Syntax

```
          +---------+   +-----------+
logger ---¦ -t tag  +---¦           +---¦
          ¦ -p pr   ¦ ¦ +- message -+
        ¦ ¦ -i      ¦ ¦
        ¦ ¦ -f file ¦ ¦
        ¦ +---------+ ¦
        +-------------+
```

Description

The **logger** command provides a program interface to the **syslog** system log module.

A message can be given on the command line, which is logged immediately, or a file is read and each line is logged.

Flags

-t tag
Mark every line in the log with the specified **tag**.

-p pri
Enter the message with the specified priority. The priority may be specified numerically or as a "facility.level" pair. For example, "-p local3.info" logs the message(s) as **info**rmational level in the **local3** facility. The default is "user.notice."

-i
Log the process ID of the logger process with each line.

-f file
Log the specified **file**.

message
The **message** to log; if not specified, the **-f** file or standard input is logged.

Examples

1. To reboot the system.

```
logger
logger -p local0.notice -t HOSTIDM -f /dev/idmc
```

Related Information
See the following command: "syslogd" in topic 1.1.457.

1.1.241 login

Purpose
Logs you in to the system.

Syntax

```
        +----------+    ¦
login ---¦          +---¦
        +- -r node -+
```

```
----------------
¦ This command is not normally entered on the command line.
```

Description
The **login** program logs you into the system. Its primary functions are:

To validate your passwor
To make the required accounting and log entrie
To set up your processing environmen
To run the command interpreter that is specified in the password file which is usually the **sh** program.

A logger process, which initially runs the **getty** program, is started for each enabled port. The **getty** program reads a login name and sets work station modes (see "getty" in topic 1.1.189). Then it runs the **login** program, which may ask you for a password. If you do not have a password, you will be immediately logged in.

Your login attempt can fail for the following reasons:

Your login name/password pair does not match an entry in the passwor file.
Your password has expired. This can happen if your system require that you change your password after a specified number of days. If your password expires, the **login** program, instead of letting you log in runs the **passwd** command. (For more information, see "passwd, chfn, chsh" in topic 1.1.312.) After you change your password, you can attempt to log in again.
The system has reached the limit of simultaneously logged-in users Each AIX kernel sets a limit on the number of concurrent log ins by non-privileged users; this limit may be one. A ***privileged user*** is a user who has a user ID between 0 and 20. A privileged user can log in at any time.

In one special case, the **login** program does not ask for a user name and password pair: when the login port is the console and the file **/etc/autolog** contains a valid user name. In this case, the **login** program automatically creates a login session for that user. Other processing by the **login** program proceeds normally.

When a user logs in successfully, the **login** program makes entries in the file **/etc/utmp**, which contains a list of users logged into the system. This program also makes entries in the file **/usr/adm/wtmp**, if it exists, which is used for accounting. When an invalid login is attempted (due to an incorrect login name or password), the **login** program makes entries in the **/etc/.ilog** file.

After you log in to the system, the **login** program lists the system message of the day (which is stored in the **/etc/motd** file) and informs you if you have mail. These messages are suppressed if there is a **.hushlogin** file in your home directory. If a user file size limit has been specified in the **passwd** file, the limit is set with **ulimit** system call. When you log in as user **root** or **su** and the **/etc/.ilog** file is not empty, you see a message advising the you to check the **/etc/.ilog** file for a record of unsuccessful login attempts.

The **login** program sets the **LOGNAME** and **HOME** environment variables from information in the password file. Environment variables (such as those specified in **/etc/environment**) that are inherited from the **getty** and **init** commands are kept. You may expand or modify the environment by supplying additional parameters to the **login** program when it requests your login name. These may take the form **xxx** or **xxx=yyy**. Parameters without an equal sign are placed in the environment as **Lnum=xxx**, where **num** is a number starting at 0 and incremented each time a new variable name is required. Parameters containing an equal sign are placed into the environment without modification. If they already exist, the new assignment replaces the older value. There are two exceptions: you cannot change the shell variables **PATH** and **SHELL**. (This restriction prevents people logging into restricted environments from spawning secondary shells that are not restricted.) Both the **login** and **getty** understand simple single-character quoting conventions. Typing a \ (backslash) in front of a character quotes it and allows you to include such items as spaces and tab characters.

Note: The default language is selected from **/etc/ports**.

The **login** command changes the current directory to your HOME directory, changes the ownership of the port (work station) to the user logging in, sets the user- and group-IDs of the process, and then runs the program specified for the user in the **password** file, which is usually the shell (**/bin/sh**). The **login** program calls this program with a name consisting of a - (minus) followed by the last segment of its path name. An instance of the shell can therefore determine from its invocation name whether it is a login shell or a subshell.

The **/etc/passwd** file entry may include parameters that are always passed to the shell program. For more details, see the **passwd** file in *AIX Operating System Technical Reference*.

If the file **/etc/nologin** exists, the **login** program prints its contents on your terminal and exits. This is used by the **shutdown** program to stop users from logging in when the system is about to go down.

Flags

-r node	Identifies the login as a remote login and specifies the node requesting the login.

Files

/etc/utmp	Accounting file.
/usr/adm/wtmp	Accounting file.
/etc/.ilog	Accounting file.
/etc/autolog	Login ID for automatic login.
/usr/motd	Message of the day.
/etc/passwd	Password file.
.hushlogin	Suppresses login messages.

/etc/nologin	Stops logins.

Related Information

See the following commands: "adduser, users" in topic 1.1.15, "csh" in topic 1.1.100, "getty" in topic 1.1.189, "init, telinit" in topic 1.1.208, "passwd, chfn, chsh" in topic 1.1.312, and "pstart, penable, pshare, pdelay" in topic 1.1.338, "Restricted Shell" in topic 1.1.420.25, "sa" in topic 1.1.403, "sh, Rsh" in topic 1.1.420, and "su" in topic 1.1.449.

See the **passwd** and **utmp** files in *AIX Operating System Technical Reference*.

See the discussion of login sessions in *Managing the AIX Operating System*.

1.1.242 logname

Purpose
Displays your login name.

Syntax

```
logname ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **logname** command writes to standard output the name you used to log into the system. The command is the contents of the environment variable **$LOGNAME**, which is set when you log into the system.

Related Information
See the following commands: "env, printenv" in topic 1.1.151 and "login" in topic 1.1.241.

See the **logname** subroutine, **usrinfo** file and **environ** special facility in *AIX Operating System Technical Reference*.

1.1.243 look

Purpose
Consults a sorted **file** and prints all lines that begin with **string**.

Syntax

```
        +-------------+              +------------+
look ---¦ +---------+ +--- string ---¦            +---¦
        +-¦ -d      +-+              +--- file ---+
          ¦ -       ¦¦                          ¦
         ¦¦ -t char ¦¦                 +--------+
         ¦+---------+¦
         +-----------+
```

Note: This command does not have MBCS support.

Description

The **look** command consults a sorted **file** and prints all lines that begin with **string**. It uses binary search.

The options **-d** and **-f** affect comparisons as in **sort**.

If no **file** is specified, **/usr/dict/words** is assumed with collating sequence **-df**.

Flags

-d "Dictionary" order: only letters, digits, tabs and blanks participate in comparisons.

-f Fold. Uppercase letters compare equal to lower case.

-t char Use **char** as the tab character.

Related Information

See the following commands: "sort" in topic 1.1.432 and "grep, egrep, fgrep" in topic 1.1.193.

1.1.244 lookbib, indxbib

Purpose
Makes an inverted index.

Syntax

```
           +------+
lookbib ---¦      +--- database ---¦
           +- -n -+

indxbib --- database ---¦
```

Note: This command does not have MBCS support.

Description

The **indxbib** command makes an inverted index to the named **databases** (or files) for use by **lookbib** and **refer**. These files contain bibliographic references (or other kinds of information) separated by blank lines.

A bibliographic reference is a set of lines, constituting fields of bibliographic information. Each field starts on a line beginning with a "%," followed by a key-letter, then a blank, and finally the contents of the field, which may continue until the next line starting with "%."

The **indxbib** command is a shell script that calls **/usr/lib/refer/mkey** and **/usr/lib/refer/inv**. The first program, **mkey**, truncates words to 6 characters, and maps uppercase to lowercase. It also discards words shorter than 3 characters, words among the 100 most common English words, and numbers (dates) < 1900 or > 2000. These parameters can be changed; see page 4 of the **refer** document by Mike Lesk. The second program, **inv**, creates an entry file **.ia**, a posting file **.ib**, and a tag file **.ic**, all in the working directory.

The **lookbib** command uses an inverted index made by **indxbib** to find sets of bibliographic references. It reads keywords typed after the ">" prompt on the terminal, and retrieves records containing all these keywords. If nothing matches, nothing is returned except another ">" prompt.

The **lookbib** command will ask if you need instructions, and will print some brief information if you reply "y".

It is possible to search multiple data bases, as long as they have a common index made by **indxbib**. In that case, only the first argument given to **indxbib** is specified to **lookbib**.

If the **lookbib** command does not find the index files (the data base **.i[abc]** files), it looks for a reference file with the same name as the argument, without the suffixes. It creates a file with a **.ig** suffix, suitable for use with **fgrep**. It then uses this fgrep file to find references. This method is simpler to use, but the data base **.ig** file is slower to use than the data base .i[**abc**] files, and does not allow the use of multiple reference files.

Flag

-n turns off the prompt for instructions.

Related Information

See the following commands: "refer" in topic 1.1.365, "addbib" in topic 1.1.14, "sortbib" in topic 1.1.433, "roffbib" in topic 1.1.382 and "lookbib, indxbib."

1.1.245 lorder

Purpose
Finds the best order for files in an object library.

Syntax

```
lorder -- file --¦
                 ¦
        +------+
```

Description
The **lorder** command reads one or more object or library archive **file**s, looking for external references and writing a list of paired file names to standard output. The first file in each pair contains references to identifiers that are defined in the second file. You can send this list to the **tsort** command to find an ordering of a library member file suitable for one-pass access by the **ld** command.

The **lorder** command accepts any object or archive file regardless of its suffix as long as multiple input files are specified. A single file must have a **.o** suffix.

Example

To create a subroutine library:

```
  lorder charin.o scanfld.o scan.o scanln.o | tsort | xargs ar qv libsubs.a
```

This command creates a subroutine library named **libsubs.a** that contains the files **charin.o**, **scanfld.o**, **scan.o**, and **scanln.o**.

The ordering of the object modules in the library is important. The **ld** command requires that each module precede all the other modules that it calls or references. The **lorder** and **tsort** commands together add the subroutines to the library in the proper order.

Suppose that the file **scan.o** calls the files **scanfld.o** and **scanln.o**. The file **scanfld.o** also calls the file **charin.o**. First, the **lorder** command creates a list of pairs that shows these dependencies:

```
  charin.o charin.o
  scanfld.o scanfld.o
  scan.o scan.o
  scanln.o scanln.o
  scanfld.o charin.o
  scan.o scanfld.o
  scan.o scanln.o
```

Next, the | (vertical bar) sends this list to the **tsort** command, which converts it into the ordering we need:

```
  scan.o
  scanfld.o
  scanln.o
  charin.o
```

Each module precedes the module it calls. The file **charin.o**, which does not call another module, is last.

The second | (vertical bar) then sends this list to the **xargs**, command which constructs and runs the following **ar** command:

```
  ar  qv  libsubs.a  scan.o  scanfld.o  scanln.o  charin.o
```

This **ar** command creates the properly ordered library.

Files

/tmp/sym*	Temporary files.

Related Information
See the following commands: "ar" in topic 1.1.23, "ld" in topic 1.1.226, "nm" in topic 1.1.298 "tsort" in topic 1.1.486, and "xargs" in topic 1.1.544.

See the **ar** file in *AIX Operating System Technical Reference*.

1.1.246 lp, cancel

Purpose

Sends or cancels a request to an **LP** line printer.

Syntax

```
      +--------------+
lp ---¦ +----------+ +--- file ---¦
      +-¦ -        +-+          ¦
    ¦   ¦ -        ¦¦           ¦
    ¦  ¦¦ -nnumber ¦¦           ¦
    ¦  ¦¦ -ttitle  ¦¦           ¦
    ¦  ¦+----------+¦           ¦
    ¦  +------------+           ¦
    +---------------------------+

cancel --- file ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **lp** command accesses the **print** command for printing a file on a line printer. The **cancel** command accesses the print command to cancel a print order.

Flags

The following flags for the **lp** command may appear in any order and may be intermixed with filenames.

-c Make copies of the **files** to be printed immediately when **lp** is invoked. Normally, **files** will not be copied, but will be linked whenever possible. If the **-c** option is not given, then the user should be careful not to remove any of the **files** before the request has been printed in its entirely. In the absence of the **-c** option, any changes made to the named **files** after the request is made but before it is printed are reflected in the printed output.

-m Send mail (see **mail**) after the files have been printed. By default, no mail is sent upon normal completion of the print request.

-nnumber Print **number** copies (default of 1) of the output.

-ttitle Print **title** on the banner page output.

Related Information

See the following commands: "qdaemon, lp" in topic 1.1.347.

1.1.247 lpq

Purpose
Displays the status of the queues and printers.

Syntax

```
lpq ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **lpq** command displays the current queue and printer status.

Related Information

See the following commands: "lpr" in topic 1.1.248, "lprm" in topic 1.1.249 and "print" in topic 1.1.326.

1.1.248 lpr

Purpose
Off line print.

Syntax

```
       +-----------------------------+
lpr ---¦ +----+  +-------+           +--- file ---¦
       +-¦ -h +--¦ -#num +-- -Ttitle -+          ¦
         ¦ -m ¦ +-------+¦             +--------+
        ¦¦ -r ¦¦+---------+
        ¦+----+¦
        +------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **lpr** command accesses the **print** command and converts an **lpr** invocation into a **print** call.

Flags

The single letter options have the following meaning.

-r Remove the file upon completion of spooling or upon completion of printing.

-m Send mail upon completion.

-h Suppress the printing of the burst page.

-T title **Title** is used in place of the file name as the title.

-# num Produces **num** copies of the output. For example,

```
lpr -#3 foo.c bar.c more.c
```

produces 3 copies of the file foo.c, then 3 copies of the file bar.c, then 3 copies of the file more.c.

Related Information

See the following commands: "lpq" in topic 1.1.247, "pr" in topic 1.1.322 and "print" in topic 1.1.326.

1.1.249 lprm

Purpose
Remove jobs from the line printer spooling queue.

Syntax

```
lprm ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

AIX does not implement the 4.3 BSD **lprm** command. When the **lprm** command is entered on the command, the user is warned to use the **print** command.

Related Information

See the following command: "print" in topic 1.1.326.

1.1.250 lpstat

Purpose

Displays the status of the queues and printers.

Syntax

```
lpstat ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **lpstat** command displays the current queue and printer status.

Related Information

See the following commands: "lp, cancel" in topic 1.1.246 and "print" in topic 1.1.326.

1.1.251 lptest

Purpose
Generates line printer ripple pattern.

Syntax

```
          +-----------------------+
lptest ---¦          +---------+ +---¦
          +- length -¦         +-+
                     +- count -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **lptest** command writes the traditional "ripple test" pattern on standard output. In 96 lines, this pattern will print all 96 printable ASCII characters in each position. While originally created to test printers, it is quite useful for testing terminals, driving terminal ports for debugging purposes, or any other task where a quick supply of random data is needed.

The **length** argument specifies the output line length if the default **length** of 79 is inappropriate.

The **count** argument specifies the number of output lines to be generated if the default count of 200 is inappropriate. If **count** is to be specified, **length** must also be specified.

1.1.252 ls, lf, lr

Purpose
Displays the contents of a directory.

Syntax

```
      +---------------+   +--------+   +--------+   +--------+
ls ---¦   +-------+   +---¦ one of +---¦ one of +---¦ one of +---
      ¦   ¦ -a -R ¦   ¦   ¦ +----+ ¦   ¦ +----+ ¦   ¦ +----+ ¦
      +---¦ -d -r +---+   +-¦ -F +-+   +-¦ -b +-+   +-¦ -C +-+
          ¦ -i -s ¦         ¦ -p ¦       ¦ -q ¦       ¦ -x ¦
          ¦ -A -L ¦         +----+       +----+       ¦ -m ¦
          ¦ -N -H ¦                                   ¦ -1 ¦
          ¦ -v -S ¦                                   +----+
          ¦    -G ¦
          ¦    -z ¦
          +-------+
    +----------------------------------+   +-------- . --------+
 ---¦   one of                         +---¦   +-----------+   +---¦
    ¦   +----+   +----+                ¦   +---¦ file      +---+
    ¦ +-¦ -c +---¦ -l +--------------+ ¦       ¦ directory ¦
    ¦ ¦ ¦ -u ¦   ¦ -t ¦ ¦            ¦ ¦     ¦ +-----------+ ¦
    +-¦ +----+ ¦ +----+ ¦            +-+     +---------------+
      ¦        +--------+            ¦
      ¦ +---------------+   +------+ ¦
      +-¦    one of     +---¦      +-+
        ¦   +-------+   ¦   +- -t -+
        +---¦ -g -n +---+
            ¦ -k -o ¦
          ¦ ¦ -l    ¦ ¦
          ¦ +-------+ ¦
          +-----------+
             +------------+   +--------+   +-----------------+
ls --- -f ---¦   +----+   +---¦ one of +---¦                 +---¦
             +---¦ -d +---+   ¦ +----+ ¦   +--- directory ---+
                 ¦ -i ¦       +-¦ -C +-+                   ¦
               ¦ ¦ -s ¦ ¦       ¦ -x ¦     +-------------+
               ¦ +----+ ¦       ¦ -m ¦
               +--------+       +----+
```

Description
The **ls** command writes to standard output the contents of each specified **directory** or the name of each specified **file**, along with any other information you ask for with the flags. If you do not specify a **file** or **directory**, **ls** displays the contents of the current directory. The **lf** command is equivalent to **ls -F**. The **lr** command is equivalent to **ls -R**.

Note: Users running in different locales see different listings on the screen. For example, suppose two users are listing the contents of a single directory. One user is operating in an English or European locale, and one in a Japanese locale.

The user running in the English or European locale sees mostly ASCII characters with some extended characters mixed in. File names in ASCII characters are readable, as are file names with extended characters. File names in Japanese characters are difficult or impossible to read.

The user running in the Japanese locale usually sees a mixture of ASCII and Japanese characters. (System files retain their ASCII names in order to work properly, even if all user files are given Japanese names.) File names in ASCII characters are readable in the Japanese locale, as are file names in Japanese characters. File names with English or European extended characters are difficult or impossible to read.

By default, **ls** displays all information in alphabetic order by file name. The collating sequence is determined by the **LC_COLLATE** environment variable or, in the absence of **LC_COLLATE**, by the **LANG** environment variable. Individual file names are listed before directory names.

There are three main ways to format the output:

List one entry per line. This is the default format unless output i directed to a terminal, in which case **-C** is the default.
List entries in multiple columns by specifying either the **-C** or **-x** flags.
List entries in a comma-separated series by specifying the **-m** flag.

To determine the number of character positions in the output line, **ls** uses the environment variable **COLUMNS**. If this variable is not set, it reads the **terminfo** file. If **ls** cannot determine the number of character positions by either of these methods, it uses a default value of 80.

The mode displayed with the **-l** flag is interpreted as follows:

If the first character is:

d	The entry is a directory.
b	The entry is a block special file.
c	The entry is a character special file.
h	The entry is a hidden directory and either the **-H** flag was specified, the file argument was escaped with a trailing @, or there was not a component that could be matched in a hidden directory.
l	The entry is a symbolic link and either the **-N** flag was specified or else the symbolic link did not point to an existing file.
p	The entry is a first-in first-out (FIFO) special file.
s	The entry is a socket.
-	The entry is an ordinary file.

The next nine characters are divided into three sets of three characters each. The first three characters show the owner's permission. The next set of three characters show the permission of the other users in the group. The last set of three characters show the permission of any one else with access to the file. The three characters in each set show read, write and execute permission of the file. Execute permission of a directory lets you search a directory for a specified file.

Permissions are indicated as follows:

r	You can read the file.
w	You can edit (write) the file.
x	You can execute the file or search the directory.
-	You do not have permission to the file.

The group-execute permission character is **s** if the file has set-group ID mode. The user-execute permission character is **s** if the file has set-user-ID mode. The last character of the mode (normally **x** or -) is **t** if the 1000 (octal) bit of mode is set; see "chmod" in topic 1.1.67 for the meaning of this mode. The indications of set-ID and 1000 bit of the mode are capitalized (**S** and **T** respectively) if the corresponding execute permission is not set.

When the size of the files in a directory are listed, the **ls** command displays a total count of blocks, including indirect blocks.

The environment variables **LANG** and **LC_TIME** control the format of the date and time.

Flags

-A Lists all entries, except **.** (dot) and **..** (dot dot). This flag is implied if the effective user ID is 0 (root).

-a Lists all entries in the directory including the entries that begin with a **.** (dot).

-b Displays nonprintable characters in an octal \nnn notation.

-c Uses the time of last modification of the inode (file created, mode changed, and so on) for sorting (when used with **-t**) or for displaying (when used with **-l**). This flag has no effect when not used with either **-t** or **-l** or both.

-C Sorts output vertically in a multi-column format.

-d Displays only the information for the directory named. This is useful with the **-l** flag to get the status of a directory. See also the note under the **i** option below.

-f Lists the name in each slot for each named **directory**. This flag turns off **-l**, **-t**, **-s**, and **-r**, and turns on **-a**; the order is the order in which entries appear in the directory.

-F Lists all directories with a trailing **/**, executable files with a trailing *****, hidden directories without a matching component with a trailing **/**, and symbolic links with a trailing @.

Symbolic links are shown with the trailing @ only if the **-N** flag is used or if the link points to a non-existent file; otherwise, information about the target file is displayed. This option can also be invoked by entering the **lf** command.

-g Displays the same information as with **-l**, except for the owner.

-G Displays the global file system numbers of the files.

-H Treats all hidden directories as regular directories, and suppresses the "sliding through" that would otherwise occur.

-i Displays the i-number in the first column of the report for each file.

Note: The inode number of a hidden directory is provided by **ls -i** or **ls -di** if the directory is empty. If it has components (such as i386), each of the above commands displays the inode numbers of the components instead of the hidden directory.

-k Displays the mode, site, owner, group, size (in bytes), and time of last modification for each file.

-L If an argument is a symbolic link, lists the file or directory the link references rather than the link itself. This is the default, but, can be used to override **-N**.

-l Displays the mode, number of links, owner, group, size (in bytes), and time of last modification for each file. If the file is a special file, the size field will instead contain the major and minor device numbers. This option can also be invoked by entering the **if** (type "if").

-m Uses stream output format (a comma-separated series).

-n Displays the same information as with **-l**, except that it displays the user and the group IDs instead of the user and group names.

-N Symbolic links are not followed when determining the status of a file. Using this flag with other flags gives the user BSD semantics for symbolic links.

-o Displays the same information as with **-l**, except for the group.

-p Puts a slash after each file name if that file is a directory. This is useful when you pipe the output of **ls** to the **pr** command as follows:

```
ls -p | pr -5 -t -w80
```

-q Displays nonprintable characters in file names as the character ?.

-r Reverses the order of the sort, giving reverse alphabetic or the oldest first, as appropriate.

-R Lists all subdirectories recursively. This option can also be invoked by entering the **lr** command.

-s Gives size in kilobytes (including indirect blocks) for each entry.

-S Include in the list the site name of the current synchronization site (CSS) for the file being displayed.

-t Sorts by time of last modification (latest first) instead of by name.

-u Uses the time of the last access instead of time of the last modification for sorting (when used with **-t**) or for displaying (when used with **-l**). This flag has no effect when not used with either **-t** or **-l** or both.

-v Specifies the version of the file (count of the number of times the file has been updated). Must be used with the **-l** option.

-x Sorts output horizontally in a multi-column format.

-z Include an octal representation of the replicated storage mask (fstore value) in the listing of the file being displayed. To have this information displayed in a symbolic form, use the **where** command.

-1 Forces one entry per line output format; this is the default when output is not to a terminal.

Examples

1. To list all files in the current directory:

   ```
   ls  -a
   ```

   This lists all files, including . (dot), .. (dot-dot), and other files with names beginning with a dot.

2. To display detailed information:

   ```
   ls  -l  chap1  .profile
   ```

   This displays a long listing with detailed information about **chap1** and **.profile**.

3. To display detailed information about a directory:

   ```
   ls  -d  -l  .  manual  manual/chap1
   ```

   This displays a long listing for the directories . and **manual**, and for the file **manual/chap1**. Without the **-d** flag, this would list the files in **.** and **manual** instead of the detailed information about the directories themselves.

4. To list the files in order of modification time:

   ```
   ls  -l  -t
   ```

   This displays a long listing of the files that were modified most recently, followed by the older files.

Files

/etc/passwd	Contains user IDs.
/etc/group	Contains group IDs.
/usr/lib/terminfo/*	Contains terminal information.

Related Information

See the following commands: "chmod" in topic 1.1.67, "ctab" in topic 1.1.103, "find" in topic 1.1.165, and "where" in topic 1.1.534.

See the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

See "Overview of International Character Support" in *Managing the AIX Operating System*.

1.1.253 mail, Mail

Purpose
Sends and receives mail.

Syntax

```
+------+   +----------------+   +-----------+   +----------------------------
¦ mail +---¦ +------------+ +---¦           +---¦                  +-----------
¦ Mail ¦   +-¦ -file name +-+   +- user-ID -+   +- @ system-name -¦
+------+     ¦ -i         ¦                                        +--- .domai
             ¦ -n         ¦
             ¦ -N         ¦                                          +--------
             ¦ -s subject ¦
             ¦ -u user    ¦
             ¦ -v         ¦
             +------------+

    +----------------------------------------+
  ---¦     +------------+   +-------------+ +---
     +- : -¦            +---¦             +-+
           +- uucphost -+   +- ! user-ID -+

    +--------------------------------------------------------+
  ---¦                               +--------------------+ +---¦
     +- : --- user-ID @ system_name -¦                    +-+
                                     +--- .domain_name ---+
                                                         ¦
                                       +-----------------+
```

Description

The **mail** program allows you to:

    Compose a message and send i
    Receive a message and look at i
    Store received messages in your mailbox or in folder
    Discard messages

To send a message to one or more persons, enter **mail** or **Mail** on the command line with arguments that are the network addresses of the people who are to receive the message. When the **mail** command starts, you can type the message using an editor similar to **ed**. When you are finished with the message, press the **Enter** key at the end of a line, and use the **Ctrl-D** (EOF) sequence at the beginning of the next line to exit the editor and send the message.

Mail standards limit mail to 7-bit ASCII characters. The mail utilites usually do not handle multibyte characters except in the names of files on the local system. However, multibyte characters included in a mail message are sent reliably if the sender and the receiver are using the same locale and are in the same AIX cluster.

When you have messages in your mailbox, the system displays the default message:

```
  [YOU HAVE NEW MAIL]
```

To look at the contents of your mailbox, enter the **mail** command without arguments on the command line. The program displays a listing of the messages in your mailbox and allows you to look at them, reply to them, or dispose of them.

Subtopics

1.1.253.1 Reading Incoming Mail

To receive and read incoming mail, use the **mail** command with no arguments:

```
  mail
```

The **mail** command then checks your system mailbox (**$HOME/.newmail**) and displays a one-line entry for each message in the system mailbox similar to:

```
  "$HOME/.newmail": 2 messages, 2 new
  >N  1 amy      Thu Sep 17 14:36  13/359 "Dept Meeting"
   N  2 amy      Thu Sep 17 16:28  13/416 "Dept Meeting Delayed"
  &
```

The **>** symbol indicates the ***current message***, or the message that commands act on if you do not specify a message number or list of message numbers. The other fields, in order, in the listing represent:

1. Message number
2. User address of the sender
3. Date the message was sent
4. Size of the message in lines/characters
5. The subject of the message (if one is included in the message).

From the **mail** command prompt **&**, you can enter commands to look at, reply to, save, discard, or manage the contents of the mailbox. To display a summary of some of the commands that you can use to handle mail in your mailbox, enter **?** at the **mail** command prompt. For more information on those commands and information on additional commands, refer to the table on the following page.

Many mailbox commands allow you to specify groups of messages upon which you can apply the command. Commands that allow you to specify groups of messages use the parameter **msg_lst** in the command format. For example, the format of the **f** command (display information about messages) appears as:

```
  & f msg_lst
```

In this format **msg_lst** can be one of the following:

One or more message numbers separated by space

```
    & f 1 2 4 7
```

A range of message numbers indicated by the first and last numbers i the range separated by a dash

```
    & f 2-5
```

is the same as

```
    & f 2 3 4 5
```

One or more addresses separated by spaces to apply the command t messages received from those addresses

```
    & f amy geo@zeus
```

The characters entered for an address need not match the address exactly. They must only be contained in the address field of the messages in either uppercase or lowercase. Therefore, the request for address **amy** matches all of the following addresses (and many others):

- **amy**
- **AmY**
- **amy@zeus**
- **hamy**

A string, preceded by a slash, to match against the **Subject:** field of the messages

```
& f /meet
```

In this example, the command applies to all messages whose **Subject:** field contains the letters **meet** in uppercase or lowercase. The characters entered for a match pattern do not need to match the **Subject:** field exactly. They must only be contained in the Subject: field of the messages in either uppercase or lowercase. Therefore, the request for subject **meet** matches all of the following subjects (and many others):

- **Meeting on Thursday**
- **Come to meeting tomorrow**
- **MEET ME IN ST. LOUIS**

The special character * (asterisk) addresses all messages, and (caret) addresses the first message, and $ (dollar sign) addresses the last message.

The following table lists the **mail** commands and describes their functions.

Command	**Function**
=	Echoes the number of the current message.
#	Comment character for writing comments in mail script files.
-n	Goes to the previous message and displays it. If given number argument **n**, goes to the **n**th previous message and displays it.
?	Displays a brief summary of commands.
!sh_cmd	Executes the AIX shell command specified by **sh_cmd**.
alias	(**a**) With no arguments, displays all currently defined aliases. With one argument, displays alias. With more than one argument, creates a new or changes an old alias.
alternates alt_list	(**alt**) The **alternates** command is useful if you have accounts on several machines. Use it to inform the **mail** program that the addresses listed in **alt_list** all refer to you. Then, when you reply to messages, **mail** does not send a copy of the message to any of the addresses given in **alt_list**. If you enter the **alternates** command with no argument, **mail** displays the

current set of alternate names.

chdir dir	(**cd**) Changes your working directory to the directory **dir**. If no directory is given, it changes to your login directory.
copy msg_lst file	(**c**, **co**) Appends each message in **msg_lst** in turn to the end of **file**. Displays the file name in quotes, followed by the line count and character count, on the user's terminal. Does not mark the messages it is used on to be deleted when you quit.
delete msg_lst	(**d**) Marks the messages in **msg_lst** to be deleted when you quit the **mail** program. Deleted messages are neither saved in your personal mailbox (see the **mbox** command) nor are they available for most other commands. However, you can restore messages that you have deleted while in the same mailbox session (see the **undelete** mail command).
discard [**fld_lst**]	(**di**) Identical to the **ignore** command.
dp	Deletes the current message and displays the next message. If there is no next message, the **mail** program displays one of the following messages:
	at EOF (There are active messages, but no more at the end.)
	no more messages (No active messages.)
dt	Deletes the current message and displays the next message. If there is no message, the **mail** program displays one of the following:
	at EOF (There are active messages, but no more at the end.)
	no more messages (No active messages.)
echo string	Displays the character string **string** on the command line.
edit msg	(**e**) Activates the editor that you define with the **set EDITOR=** statement and loads message **msg** into the editor. When you exit the editor, the saved message is replaced in the mailbox being processed.
exit	(**ex** or **x**) Exits to the shell without changing the mailbox being processed. The mailbox returns to the condition that it was in when the **mail** program was started. Messages marked to be deleted are not deleted.
file [**name**]	(**fi**) Identical to the **folder** command.
folder [**name**]	(**fo**) Switches to a new mail file or folder. With no arguments, displays the name of the mailbox that you are currently reading. If an argument is included, it stores the current mailbox with changes (such as

	messages deleted) and reads in the new mailbox specified by the **name** parameter. The following special conventions are recognized for **name**: # refers to the previous file % refers to the system mailbox & refers to your personal mailbox (**$HOME/mbox**) +**name** refers to a file in your folder directory.
folders	Lists the names of the folders in your folder directory.
from msg_lst	(**f**) Displays the headings of messages in **msg_lst**.
group	(**g**) Identical to the **alias** command.
headers	(**h**) Lists the headings in the current group of messages (each group of messages contains 20 messages by default; change this with the **set screen=** statement).
help	Identical to question mark (?).
hold msg_lst	(**ho**) Marks each message in **msg_lst** to be saved in your system mailbox instead of in **mbox**. Does not override the **delete** command.
if condition else endif	Construction for conditional execution of **mail** commands. Commands following **if** are executed if **condition** is true. Commands following **else** are executed if **condition** is not true. The **else** is not required. The **endif** ends the construction and is required. The **condition** can be **receive** (receiving mail) or **send** (sending mail).
ignore [**fld_lst**]	Adds the header fields in **fld_lst** to the list of fields to be ignored. Ignored fields are not displayed when you look at a message with the **t** or **p** commands. Use this command to suppress machine-generated header fields. Use the **Type** and **Print** commands to print a message in its entirety, including ignored fields. If **ignore** is executed with no arguments, it lists the current set of ignored fields.
list	(**l**) Displays a list of valid **mail** commands.
local	Lists other names for the local host.
mail addr_lst	(**m**) Activates the mail editor to allow you to create and send a message to people specified in **addr_lst**.
mbox msg_lst	Indicates that the messages in **msg_lst** be sent to your personal mailbox when you quit. This operation is the default action for messages that you have looked at if you are looking at your system mailbox and the **hold** option is not set.
more msg_lst	(**mo**) Displays the messages in **msg_lst** using the defined pager program to control display to the

screen.

More msg_lst	(**Mo**) Like **more**, but also displays ignored header fields. See **more** and **ignore**.
new msg_lst	Identical to the **unread** command.
New msg_lst	Identical to the **Unread** command.
next [**msg**]	(**n**) Makes the next message in the mailbox the current message and displays that message. With an argument list, it displays the next matching message.
page msg_lst	(**pa**) Identical to the **more** command.
Page msg_lst	(**Pa**) Identical to the **More** command.
preserve	(**pre**) Identical to the **hold** command.
print msg_lst	(**p**) Displays the messages in **msg_lst**.
Print msg_lst	(**P**) Like **print**, but also displays ignored header fields. See **print** and **ignore**.
quit	(**q**) Ends the session and returns to the shell. Before ending, **mail** saves all messages that have not been deleted or saved in your personal mailbox (**$HOME/mbox**). It keeps all messages marked with **hold** or **preserve** and those messages that have not been looked at in the system mailbox. It removes all other messages from the system mailbox. If given while editing a mailbox file with the **-f** flag, the edit file is saved with changes. If the edit file cannot be saved, **mail** does not exit. Use the **exit** command to exit without saving the changes.
reply msg	Allows you to create and send mail to the people who sent and received the message specified in **msg**.
Reply msg	Allows you to create and send mail only to the person who sent the message specified in **msg**.
respond msg	Identical to the **reply** command.
Respond msg	Identical to the **Reply** command.
retain [**fld_lst**]	Adds the header fields in **fld_lst** to the list of fields to be retained. Retained fields are displayed when you look at a message with the **t** or **p** commands. Use this command to define which header fields you want displayed. Use the **Type** and **Print** commands to print a message in its entirety, including fields that are not retained. If **retain** is executed with no arguments, it lists the current set of retained fields.
save msg_lst file	(**s**) Appends the messages specified in **msg_lst** to **file**. Displays the file name and the size of the file when the operation is complete. If you save a message to a file, that message is neither returned to the system

	mailbox nor saved in your personal mailbox when you quit the **mail** program.
set [**option**]	(**se**) With no arguments, prints all variable values. Otherwise, sets an option as specified in **option**. The **option** field can be either the name of a **binary** option (an option that is either set or not set) or a statement of the form: `option=value` that assigns a value to a valued option. Binary and valued options are described later in this command description.
shell	(**sh**) Invokes an interactive version of the shell.
size msg_lst	Displays the sizes in lines/characters of the messages in **msg_lst**.
source file	(**so**) Reads **mail** commands from **file**.
top msg_lst	Displays the top few lines of the messages specified by **msg_lst**. The number of lines displayed is determined by the valued option **toplines** and defaults to five.
touch msg_lst	When operating with your system mailbox, this command marks the messages in **msg_lst** to be moved to your personal mailbox when you quit the **mail** program, even though you have not read the listed messages. The messages appear in your personal mailbox as unread messages. When you use **touch**, the last message in **msg_lst** becomes the current message.
type msg_lst	(**t**) Identical to the **print** command.
Type msg_lst	(**T**) Identical to the **Print** command.
unalias al_lst	Removes the defined aliases specified in **al_lst**.
undelete msg_lst	(**u**) Removes the messages in **msg_lst** from the list of messages to be deleted when you quit **mail**.
unread msg_lst	(**U**) Marks each message in **msg_lst** as **not** having been read.
Unread msg_lst	Identical to the **unread** command.
unset option_lst	Discards the values of the options specified in **option_lst**. This action is the inverse of the **set** command.
version	(**ve**) Displays the version banner for the **mail** program.
visual msg	(**v**) Activates the editor that you define with the **set VISUAL=** statement and loads message **msg** into the editor. When you exit the editor, the saved message is replaced in the mailbox being processed.

write msg_lst file	(**w**) Appends the messages specified in **msg_lst** to **file**. Displays the file name and the size of the file when the operation is complete. Does not include message headers in the file.
xit	(**x**) Identical to the **exit** command.
z [+] [-]	Changes the current message group (group of 20 messages) and displays the headings of the messages in that group. If a + (plus) or no argument is given, the headings in the next group are shown. If a - (minus) argument is given, the headings in the previous group are shown.

1.1.253.2 Forwarding Mail

You can have mail forwarding by including a **.forward** file in your home directory.

The **.forward** file contains a single line containing one of the following:

Mail address for user

A list of user names, separated by commas

An absolute pathname where incoming messages are appended

A ¦ (vertical bar) followed by an absolute pathname. Forwarde messages are executed by the program specified by the pathname.

1.1.253.3 Handling Outgoing Mail

To compose and send a message, use **mail** with the following format:

```
  mail addr_lst
```

In this format **addr_lst** is a list of user addresses separated by spaces. This command activates the mail editor so that you can compose a message to be sent to the specified addresses.

By default, **mail** treats lines beginning with the character ~ (tilde) as special while you are composing a message. For instance, typing **~m** on a line by itself places a copy of the current message into the response, shifting it to the right by one tab stop.

Other escapes set up subject fields, add and delete recipients of the message, and allow the user to escape to an editor to revise the message, or to a shell to run other commands. You can change the escape character to something other than a tilde with the **set escape=** statement. To view a summary of many useful commands, enter **~?** on a line by itself while in the mail editor.

The following table shows a summary of the mail editor commands. Use these commands only while in the mail editor. The editor recognizes commands only if you enter them at the beginning of a new line.

Command	**Function**
~**!cmd**	Executes the shell command **cmd** and returns to the message.
~b addr_lst	Adds names in **addr_lst** to the list of people to receive blind copies of the message.
~c addr_lst	Adds names in **addr_lst** to the list of people to receive copies of the message.
~d	Reads the file **dead.letter** from your home directory into the message.
~e	Activates the editor that you have specified with the **set EDITOR=** statement using the message text in the current message. When you exit that editor, you return to the mail editor to continue appending the changed message, or to send the message by exiting the **mail** program.
~f msg_lst	Reads the named messages into the message being sent. If no messages are specified, reads the current message. This command works only if you entered the mail editor from the mailbox listing using the **m** or **r** mailbox commands.
~h	Allows you to edit the message header fields by typing each one in turn. Allows you to append text to the end or modify the field using the current terminal erase and kill characters.
~m msg_lst	Reads the named messages into the message being sent, shifted right one tab. If no messages are specified,

	reads the current message. This command works only if you entered the mail editor from the mailbox listing using the **m** or **r** mailbox commands.
~p	Displays the message as it currently exists, prefaced by the message header fields.
~q	Aborts the message being created without sending it. Saves the message in **dead.letter** in your home directory if the **save** option is set.
~r file name	Reads the named file into the message.
~s string	Changes the **Subject:** field to the phrase specified in **string**.
~t addr_lst	Adds the addresses in **addr_lst** to the **To:** field of the message.
~v	Activates the editor that you have specified with the **set VISUAL=** statement using the message text in the current message. When you exit that editor, you return to the mail editor to continue appending to the changed message, or to send the message by exiting the **mail** program.
~w file name	Writes the message to the named file.
~|cmd	Pipes the message through the command **cmd** as a filter. If **cmd** gives no output or terminates abnormally, it retains the original text of the message. Otherwise, the output of **cmd** replaces the current message. The command **fmt** is often used as **command** to rejustify the message.
~~	Allows you to use the character **~** (tilde) in a message without its being interpreted as a command prefix. The sequence **~~** results in only one ~ being sent in the message. If you have changed the escape character, double that character instead of ~ to use the new escape character as a single character.

You can end a **mail** session with the **quit (q)** command. Messages that you have looked at go to your personal mailbox. Messages that you have marked to be deleted are deleted. Messages that you have not looked at go back to your system mailbox.

1.1.253.4 Customizing the Mail Program

The **mail** command has a number of options that you can set to customize the mail system for your particular use. Use the **set** command to enable options, and the **unset** command to disable options. You can also use the **set** command to assign a value to an option.

The format for using the **set** command to enable options not requiring a value is:

```
  set [option_list]
```

The **option_list** can be one or more options that you want to enable. To set options so that they are valid each time you use **mail**, put the commands in **.mailrc** in your **$HOME** directory. To set options so that they are valid for all users on the system, put the commands in **/usr/lib/Mail.rc**. The following table lists the binary options (those that need only be set or unset).

Option	**Function**
append	Causes messages saved in **mbox** to be appended (added to the end) rather than prepended (added to the beginning).
ask	Causes **mail** to prompt you for the subject of each message you send. If you respond with a new line (carriage return), no subject field is set.
askcc	Causes you to be prompted for the addresses of people to receive copies of the message. Responding with a new line indicates your satisfaction with the current list.
autoprint	Causes the **delete** command to behave like the **dp** command. Thus, after deleting a message, the next one is typed automatically.
debug	Same as specifying **-d** on the command line. Causes **mail** to display debugging information. The **mail** command does not send mail while in debug mode.
dot	Causes **mail** to interpret a period alone on a line as the terminator of a message you are sending.
hold	Holds messages in the system mailbox by default.
ignore	Causes interrupt signals from your terminal to be ignored and echoed as @'s.
ignoreeof	Related to the **dot** option. Makes **mail** refuse to accept an **Ctrl-D** as the end of a message. **ignoreeof** also applies to **mail** command mode.
metoo	Usually, when an alias containing the sender is expanded, the sender is removed from the expansion. Setting this option causes the sender to be included in the alias expansion (and thus receives copies of messages).

nosave	Normally, when a message is terminated with two interrupt sequences (**Ctrl-C**), **mail** copies the partial letter to the file **dead.letter** in your home directory. Setting the binary option **nosave** prevents this.
Replyall	Reverses the sense of the **reply** and **Reply** mailbox commands.
quiet	Suppresses the printing of the program banner when **mail** starts.
verbose	Same as using the **-v** flag on the command line. When mail runs in verbose mode, the actual delivery of messages is displayed on the user's terminal.

The format for using the **set** command for options requiring a value is:

```
  set option=value
```

Note: Do not use a space before or after the equal sign.

The following table lists the valued options (those that need to be assigned a value).

Option	**Function**
EDITOR	Path name of the text editor to use in the **edit** command and ~**e** escape. If not defined, a default editor (**/bin/ex**) is used.
PAGER	Path name of the paging program to use for the **more** command or when the **crt** variable is set. If you do not specify a value for PAGER, the system uses **/bin/pg**.
SHELL	Path name of the shell to use in the ! command and the ~! escape. A default shell is used if this option is not defined.
VISUAL	Pathname of the text editor to use in the **visual** command and ~**v** escape. The default path name is **/bin/vi**.
crt=n	Calls the **pg** command to display the message when the message exceeds **n** lines.
escape	If defined, the first character of this option gives the character to use in the place of ~ to denote escapes.
folder	Defines the name of the directory to use for storing folders of messages. If this name begins with a **/** (slash), **mail** considers it to be an absolute path name; otherwise, the folder directory is found relative to your home directory.
record	If defined, gives the path name of the file used to record all outgoing mail. If this name begins with a **/** (slash), **Mail** considers it an absolute path name;

	otherwise it is relative to **$HOME**. If not defined, outgoing mail is not saved. Do not include the home directory as part of the path name.
screen	If defined, controls the size of the window for message headers. You can set this option to show the number of lines on the screen. For example, the entry **screen=22** causes the system to scroll for 22 lines and then pause.
toplines	If defined, gives the number of lines of a message to be printed out with the **top** command; normally, the first five lines are printed.

Flags

-v	Puts **mail** into verbose mode. Details of delivery are displayed on the user's terminal.
-i	Causes tty interrupt signals to be ignored. Useful when using **mail** on noisy phone lines.
-n	Inhibits the reading of **/usr/lib/Mail.rc**.
-N	Suppresses the initial printing of headers.
-s subject	Specifies a subject for a message to be created.
-f name	Causes **mail** to read in the contents of your **mbox** or the specified file for processing. When you quit, **mail** writes undeleted messages back to this file.
-u user_id	Short way of doing **mail -f/~userid/.newmail**. Activates **mail** for a specified users mailbox. You must have access permission to the specified mailbox.

Files

$HOME/mbox	Your personal mailbox.
$HOME/.forward	Forwards your mail as specified.
$HOME/.mailrc	File containing mail commands to customize **mail** to a specific user.
$HOME/.newmail	Your system mailbox.
/tmp/R#	Temporary for editor escape.
/usr/lib/Mail.help	Help file for mailbox commands.
/usr/lib/Mail.tildehelp	Help file for mail editor commands.
/usr/lib/Mail.rc	File containing mail commands to change **mail** for all users on the system.

Related Information

See the following commands: "bellmail" in topic 1.1.38, "sendmail, mailq, newaliases" in topic 1.1.417, and "uucp" in topic 1.1.506.

See the chapter about sending and receiving mail in *Using the AIX Operating System*.

See also the chapters about Mail Handler (an alternative to **mail**) and about managing the mail system in *Managing the AIX Operating System*.

1.1.254 make

Purpose
Maintains up-to-date versions of programs.

Syntax

```
        +------------------+   +-------------------+
make ---¦      one of      +---¦                   +---
        ¦      +----+      ¦   +--- -f makefile ---+
        ¦      ¦ -n ¦      ¦                       ¦
        ¦ +----¦ -t +----+ ¦     +---------------+
        +-¦    +----+    +-+
          ¦ +----------+ ¦
          +-¦ -m -d -e +-+
            ¦ -k -s -p ¦¦
           ¦¦ -i -q -r ¦¦
           ¦+----------+¦
           +------------+

    +----------------+   +--------------+
 ---¦                +---¦              +---¦
    +--- name=def ---+   +--- target ---+
                    ¦                  ¦
      +------------+       +----------+
```

Note: This command does not have MBCS support.

Description
The **make** command reads **makefile** for information about the specified **target** files and for the commands necessary to update them. **make** does not change the **target** if you have not changed any of the source files since you last built it. It considers a missing file to be a changed file (out of date).

You can also include macro definitions on the command line after all of the flags. Macro definitions have the form:

macro-name = **string**

See "Macros" in topic 1.1.254.5 for more information about macros and their uses.

The **make** command considers all entries on the command line that follow the flags and that do not contain an equal sign to be target file names.

Subtopics

1.1.254.1 Description File

The description file contains a sequence of entries specifying the files that the target files depend on. The general form of an entry is:

```
  targ [targ]...:[:][file]...[; cmd]...[#]
    [cmd]... [#]
```

The first line of an entry (called the **dependency line**), contains a list of targets followed by a : (colon) and an optional list of prerequisite files or dependencies. If you put shell commands on the dependency line, they must be preceded by a **;** (semicolon). All commands that follow the semicolon and all following lines that begin with a tab contain shell commands that **make** uses to build the target.

To specify more than one set of commands, you must enter more than one dependency definition. In this case, each definition must have the target name followed by two colons (**::**), a dependency list, and a command list.

The first line that does not begin with a tab or **#** (hash sign) begins a new dependency or a macro definition. Command lines are performed one at time, each by its own subshell. Thus, the effect of some shell commands, such as **cd**, does not extend across new-line characters. You can, however, put a \ (backslash) at the end of a line to continue it on the next physical line. A comment begins with a **#** and ends with a new-line character.

The first one or two characters in a command can be one of the following special characters:

-	Ignores errors returned by the command on this line.
@:	Does not display this command line.
-@ @-	Does not display this command line and ignores errors.

1.1.254.2 Suffixes

The **make** command has default rules that govern the building of most standard files. These rules depend on the standard suffixes used by the system utility programs to identify file types. These rules define the starting and ending file types so that, for example, given a specified **.o** file, **make** can infer the existence of a corresponding **.c** file and knows to compile it using the **cc -c** command.

A rule with only one suffix (that is, **.c:**) defines the building of **prog** from all its source files. Use a ~ (tilde) in the suffix to indicate a SCCS file. For example, the **.c~.o** rule governs changing an SCCS C source file into an object file. You can define rules within the description file. **make** recognizes as a rule any target that contains no slashes and starts with a dot.

You can also add suffixes to the list of suffixes recognized by **make** and add to the default dependency rules. Use the target name **.SUFFIXES** followed by the suffixes you want to add. Be careful of the order in which you list the suffixes. **make** uses the first possible name for which both a file and a rule exist. The default list is:

```
  .SUFFIXES: .o .c .c~ .y
  .y~ .l .l~ .s .s~
  .sh .sh~ .h .h~
```

You can clear the list of suffixes by including **.SUFFIXES:** with no following list.

1.1.254.3 Special Target Names

You can use some special target names in the description file to tell **make** to process the file in a different manner. The special target names are:

.DEFAULT	The commands that appear after this name in the description file tell **make** what to do if it can find no commands or default rules to tell it how to create a specific file.
.IGNORE	If this name appears on a line by itself, **make** does not stop when errors occur. Using a **-** (minus) as the first character on a line in the description file tells **make** to ignore errors for the command on that line.
.PRECIOUS	The files named on the same line as this special name are not removed when **make** is interrupted.
.SILENT	If this name appears on a line by itself, **make** does not display any of the commands that it performs to build a file.
.SUFFIXES	Use this name to add more suffixes to the list of file suffixes that **make** recognizes.

1.1.254.4 Environment

When you run **make**, it reads the environment and treats all variables as macro definitions. The **make** command processes macro definitions in the following order: internal (default) definitions, shell environment variables, description file macros, then command line macro definitions. Therefore, macro assignments in a description file normally override duplicate environment variables. The **-e** flag instructs **make** to use the environment variables instead of the description file macro assignments. Command line macro definitions always override definitions in the environment, descriptions file, and the internal definitions.

The **make** command recognizes a macro **MAKEFLAGS**, which can be assigned any **make** command line flag except **-f**, **-p**, and **-d**. When **make** begins, it assigns the current flags to **MAKEFLAGS**. It passes this variable to any commands it invokes, including additional invocations of **make** itself. Thus you can perform a **make -n** recursively on a software system to see what would have been performed. The **-n** is put in **MAKEFLAGS** and passed to further copies of the shell that runs the next level of **make** commands. In this way, you can check all of the description files for a software project without actually compiling the project.

MFLAGS are the same as **MAKEFLAGS** except **MFLAGS** includes a leading "-" (dash) if there is at least one flag present.

1.1.254.5 Macros

Entries of the form **string1 = string2** are macro definitions. **string2** can consist of all characters that can occur on a line before a comment character (#) or before a new-line character that is not a continuation line. After this macro definition, **make** replaces each **$(string1)** in the file with **string2**. You do not have to use the parentheses around the macro name if the macro name is only one character long and there is no substitute sequence (see the next paragraph). If you use the following form, you can also replace characters in the macro string with other characters for one time that you use the macro:

$(**string1**[:**subst1**=[**subst2**]])

The optional **:subst1 = subst2** + specifies a substitute sequence. If you specify a substitute sequence, **make** replaces each **subst1** in the named macro with **subst2** (if **subst1** does not overlap with another **subst1**). Strings in a substitute sequence begin and end with any of the following: a blank, tab, new-line character, or beginning of line. See "Libraries" in topic 1.1.254.7 for an example of the use of the substitute sequence.

Note: Because **make** uses the dollar sign symbol (**$**) to designate a macro, do not use that symbol in file names of targets and parents, or in commands in the description file unless you are using a defined **make** macro.

1.1.254.6 Internal Macros

The **make** command has five internal macros. It assigns values to these macros under one or more of the following conditions:

When it uses an internal rule to build a file
When it uses a **.DEFAULT** rule to build a file.
When it uses rules in the description file to build a file
When the file is a library member

They are defined as follows:

$*	The file name (without the suffix) of the source file. The **make** command only evaluates this macro when applying inference rules.
$@	The full target name of the current target. $$@ expands to the current target name when specified to the right of **:** on dependency lines.
$<	The source files of an out-of-date module. **make** evaluates this macro when applying inference rules or the **.DEFAULT** rule. For example: `.c.o:` `cc -c $<` Here, **$<** is the equivalent of **$*** and refers to the **.c** file of any out-of-date **.o** file.
$?	The list of out-of-date files. **make** evaluates this macro when it evaluates explicit rules from **makefile**.
$%	The name of an archive library member. **make** evaluates this macro only if the target is an archive library member of the form **lib(file.o)**. In this case, **$@** evaluates to **lib** and **$%** evaluates to the library member, **file.o**. $$% is recognized to the right of **:** on dependency lines to be the library member. $$% is related to $% as $$@ is related to $@. This allows for explicit **.c.a** rules to be specified.

You can add an uppercase **D** or **F** to indicate "directory part" or "file part", respectively, to all internal macros except for **$?**. Thus, **$(@D)** refers to the directory part of the name **$@**. If there is no directory part, **make** uses **./**.

1.1.254.7 Libraries

If a target name contains parentheses, **make** considers it an archive library. The string within parentheses refers to a library member. Thus, **lib(file.o)** and **$(LIB)(file.o)** both see an archive library which contains **file.o**. (You must have defined the **LIB** macro already.) The expression **$(LIB)+ (file1.o file2.o)** is not legal.

Rules that apply to archive libraries have the form **x.a**, where **.x** is the suffix of the file you want to add to an archive library. For example, **.c.a** indicates a rule that changes any C source file to a library file member. The following lines give the default rule for this change:

```
  lib:   lib(file.o) lib(file.o) lib(file.o)
         @echo lib is now up to date
  .c.a:
         $(CC) -c $(CFLAGS) $<
         ar rv $@ $*.o
         rm -f $*.o
```

.x must be different from the suffix of the archive member. Therefore, you cannot have **lib(file.o)** depend upon **file.o**.

Another, but more limited, example of an archive library maintenance rule follows:

```
  lib:   lib(file.o) lib(file.o) lib(file.o)
         $(CC) -c $(CFLAGS) $(?:.o=.c)
         ar rv lib $?
         rm $?   @echo lib is now up to date
  .c.a:;
```

This example rule uses a substitute sequence (**.o=.c**) to replace with **.c** files all **.o** files generated by the **$?** macro. The **$?** list is the set of object file names (inside **lib**) with C source files that are out of date. The macro substitution translates **.o** to **.c**.

If this rule appears in your description file, it disables the default **.c.a:** rule, which creates each object file one by one. This type of organization speeds up archive library maintenance, but becomes hard to use if the archive library contains a mix of assembly programs and C programs.

The $$% macro allows the **.c.a:** rule to be specified as follows:

```
  lib(file1.o) lib(file2.o) lib(file3.o):  $$(%:.o=.c)
         $(CC) -c $(CFLAGS) $(%:.o=.c)
         ar rv $@ $%
         rm $%
```

This is equivalent to specifying:

```
  lib(file1.o): file1.c
         $(CC) -c $(CFLAGS) $(%:.o=.c)
         ar rv $@ $%
         rm $%

  lib(file2.o): file2.c
         $(CC) -c $(CFLAGS) $(%:.o=.c)
         ar rv $@ $%
```

```
        rm $%

  lib(file3.o_: file3.c
        $(CC) -c $(CFLAGS) $(%:.o=.c)
        ar rv $@ $%
        rm $%
```

Make recognizes two commands which affect make file processing:

```
  include file
  oinclude file
```

When make encounters these commands it opens the name **file** and reads **file** as if the contents of the file were actually present in the original file. This process can be nested.

If the file is not present, **oinclude** will disregard the command as if it were not specified, while the **include** command will cause make to exit with a fatal error.

Flags

-d	Displays detailed information about the files and times that **make** examines (debug mode).
-e	Uses environment variables in place of any assignments made within description files. These assignments normally replace environment variables.
-f makefile	Reads **makefile** for a description of how to build the target file. If you give only a **-** (minus) for **makefile**, **make** reads standard input. If you do not use the **-f** flag, **make** looks in the current directory for a description file named **makefile**, **Makefile**, **s.makefile**, or **s.Makefile**. You can specify more than one description file by entering the **-f** flag more than once (with its associated **makefile** parameter).
-i	Ignores error codes returned by commands. **make** normally stops if a command returns a nonzero code. Use this flag to compile several modules only if you want **make** to continue when an error occurs in one of the modules. Do not link the resulting modules when you use this flag.
-k	Stops processing the current target if an error occurs but continues with other branches that do not depend on that target.
-m	Prints memory map when exiting program.
-n	Displays commands but does not run them. Displays lines beginning with an @ (at sign). If the command in the description file contains the string **$(MAKE)**, perform another call to **make** (see the discussion of the **MAKEFLAGS** macro on page 1.1.254.4). Use this flag to preview the performance of **make**.
-p	Displays the complete set of macro definitions and

target descriptions before performing any commands.

-q	Returns a zero status code if the target file is up to date; returns a nonzero status code if the target file is not up to date.
-r	Does not use the default rules.
-s	Does not display commands on the screen as they are performed.
-t	Changes only the date of the files, rather than performing the listed commands. Use this flag if you have made only minor changes to a source file that do not affect anything outside of that file. This flag changes the date of all target files that appear on the command line or in the description file.
-T	Directs make to print to standard out.

Examples

1. To make the file specified by the first entry in the description file:

   ```
   make
   ```

2. To display, but not run, the commands that **make** would use to make a file:

   ```
   make  -n  search.o
   ```

   You may want to do this to verify that a new description file is correct before using it.

3. To save the internal rules in a file:

   ```
   make  -p  -f  /dev/null  2> /dev/null  > defaults
   ```

   This lists the internal rules and macros and saves them in the file **defaults** for viewing or editing. All exported shell environment variables are included in the list of macro definitions.

4. To generate dependency trigger information:

   ```
   make -T -f makefile
   ```

   If file **stdio.h** and **h.c** were newer than target **execute** the following makefile rules:

   ```
   all:     default
   default: execute

   execute: x.o h.o
              cc -o execute x.o h.o

   x.o: /usr/include/stdio.h x.c
     cc -c x.c
   ```

```
      h.o: h.c /usr/include/params.h
        cc -c h.c
```

would generate the following trigger information:

```
      /usr/include/stdio.h<-x.o<-execute<-default<-all
      /source/h.c<-h.o<-execute<-default<-all
```

Files

Makefile
makefile
s.Makefile
s.makefile

Related Information

See the discussion of **make** in *AIX Operating System Programming Tools and Interfaces*.

1.1.255 makedbm

Purpose
Makes an NIS **dbm** file.

Syntax

```
                       +----------+   +--------+
/etc/yp/makedbm ---¦          +---¦ +----+ +---¦
                       +- infile -+   +-¦ -l +-+
                                        ¦ -o ¦¦
                                       ¦¦ -d ¦¦
                                       ¦¦ -m ¦¦
                                       ¦¦ -u ¦¦
                                       ¦+----+¦
                                       +------+
```

Note: This command does not have MBCS support.

Description

Note: Do not run the **makedbm** command on a client machine. The command should be used only on servers.

The **makedbm** command takes **infile** and converts it to a pair of files in **dbm** format, namely **outfile**.pag and **outfile**.dir. Each line of the input file is converted to a single **dbm** record. All characters up to the first tab or space form the key, and the rest of the line is the data. If a line ends with \, the data for that record is continued on to the next line. It is left for the clients of the NIS to interpret #; the **makedbm** command does not itself treat it as a comment character. The **infile** can be - (minus), in which case standard input is read.

The **makedbm** command is meant to be used in generating **dbm** files for the NIS, and it generates a special entry with the key **yp_last_modified**, which is the date of **infile** (or the current time, if **infile** is -).

Flags

-i	Create a special entry with the key **yp_input_file**.
-o	Create a special entry with the key **yp_output_name**.
-d	Create a special entry with the key **yp_domain_name**.
-m	Create a special entry with the key **yp_master_name**. If no master host name is specified, **yp_master_name** is sent to the local host name.
-u	Undo a **dbm** file. That is, print out a **dbm** file one entry per line, with a single space separating keys from values.

Example
It is easy to write shell scripts to convert standard files such as **/etc/passwd** to key value form used by the **makedbm** command. For example:

```
  #!/bin/awk -f
  BEGIN { FS=":";OFS="\t";} { print$1,$0}
```

takes the **/etc/passwd** file and converts it to a form that can be read by the **makedbm** command to make the NIS file **passwd.byname**. That is, the key is a user name, and the value is the remaining line in the **/etc/passwd** file.

Related Information

See the following command: "yppasswd" in topic 1.1.552.

1.1.256 makekey

Purpose
Generates an encryption key.

Syntax

/usr/lib/makekey ---¦

Description
The **makekey** command generates an encryption key to use with programs that perform encryption. Its input and output are usually pipes.

The **makekey** command reads 10 characters from standard input and writes 13 characters to standard output. The first 8 of the 10 input characters can be any sequence of ASCII characters. The last two input characters (the ***salt***), are best chosen from the set [**a-zA-Z0-9.,/**]. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt characters and constitute the output key that you use as the **key** parameter to programs that perform encryption.

Example

Note: In the following example, do **not** press **Ctrl-D** after typing the input key **1234567890** because this ends your shell session. Also, the shell prompt appears immediately after the generated key, instead of appearing on a separate line as it usually does. This is normal.

To generate an encryption key:

```
  /usr/lib/makekey
  1234567890
```

This generates an encryption key based on the string **1234567890**. The key **90y744T/NXwlU** is displayed at the work station.

1.1.257 makemotd

Purpose
Makes the system message-of-the-day on multiple sites.

Syntax

```
            +------+   +- /etc/MOTD -+
makemotd ---¦      +---¦             +---¦
            +- -L -+   +- filename --+
```

Description

The **makemotd** command makes a new system message-of-the-day, possibly on multiple sites. The system message-of-the-day for each site (**<LOCAL>/motd**), displayed when each user.logs in, is built from four parts. These are the system identifier, the load module date, the system-wide message-of-the-day, and a site-specific message-of-the-day.

To change the system message-of-the-day, one changes one of these four parts and then rebuilds <LOCAL>/motd using /etc/makemotd. Changes made to <LOCAL>/motd directly will be undone when /etc/makemotd is next run (an action typically accompanying a system reboot) and should therefore be done only when the message is to expire when the system reboots. A typical example of a direct modification to <LOCAL>/motd is the addition of a message announcing a scheduled down-time.

The system identifier is obtained from the file <LOCAL>/ident. The load module date is obtained from the file <LOCAL>/moddate. Each of these are included if the particular file is available and readable.

If file name is specified, then that file is used as the text of the system wide message-of-the-day; otherwise, the file /etc/MOTD is used.

A site-specific message-of-the-day, if present, is obtained from the file <LOCAL>/MOTD.

The -L flag included in the command line causes only the local site's message-of-the-day to be modified. Otherwise, the message of the day is updated on all sites.

The command is only available to the super user.

To summarize, there are several ways to change the system message-of-the-day. To change the message-of-the-day displayed on all sites, a system administrator can change the file /etc/MOTD and run /etc/makemotd. To put a special message on one site in addition to the standard system messages, add the message to <LOCAL>/MOTD and run /etc/makemotd or /etc/makemotd -L. To put a special message on one site which only applies until the next system reboot, the system administrator can simply change <LOCAL>/motd directly, and this change will be undone when this file is recreated on the next system reboot.

Note: If a message-of-the-day is established on a system where users may have selected different character code sets for the display of text in different languages, the message should be written using ASCII so that it is displayed consistently for all users.

Files

<LOCAL>/motd	File containing the local site system message-of-the-day.
/etc/MOTD	Default text for the system wide message-of-the-day.
<LOCAL>/MOTD	Text for a site-specific message-of-the-day.
<LOCAL>ident	Local site system identification.
<LOCAL>/moddate	Local site kernel modification date.

Related Information

See the following command: "login" in topic 1.1.241.

Restriction

The **makemotd** command stops reading **<LOCAL>/ident** and **<LOCAL>/moddate** at the first newline found.

1.1.258 man

Purpose
Displays manual entries on-line.

Syntax

```
           +-----+  +- /user/man -+  +-----------+  +-----------+
        +-¦     +--¦             +--¦          +--¦          +---+
        ¦ +- - -+  +- -Mpath ----+  +- section -+  +-- title --+   ¦
man --¦                                                   ¦    +--¦
        ¦                                            +---------+    ¦
        +------------- -k --- keyword ----------------------------¦
        ¦                             ¦                             ¦
        ¦                  +-----------+                            ¦
        +------------- -f --- file --------------------------------+
                                  ¦
                         +--------+
```

Note: This command does not have MBCS support.

Description

The **man** command provides online access to sections of the printed manuals, displaying information about commands, system calls, file formats, and other AIX technical information. With the **man** command, you can ask for one-line descriptions of commands specified by name, or for all commands whose description contains any of a set of keywords.

The **man** command formats a specified set of manual pages. If a **section** argument is given, the **man** command looks in that section of the manual for the given **title**. The **section** argument can be either a section number or one of the following characters:

Character	**Meaning**
C	Commands including system management commands.
F	Files.
L	Library functions.
n	New.
l	Local.
o	Old.
p	Public.

A section number can be followed by a single letter classifier (for example, **1g** indicates a graphics program in section 1). If **section** is omitted, the **man** command searches all sections of the manual, giving preference to commands over subroutines in system libraries, and printing the first section it finds, if any.

Unless the standard output is a teletype or the **-** flag is used, the **man** command pipes its output through the **more** command with the **-s** option to eliminate useless blank lines and to stop after each page on the screen. When viewing documentation, you can press the space bar to continue to the next page and **Ctrl-D** to scroll 11 more lines.

Flags

-f file   Attempts to locate manual sections related to the specified files, displaying the table of contents lines for those sections.

-k keyword	Displays a one-line synopsis of each manual section whose listing in the table of contents contains one of the keywords.
-Mpath	Specifies a search path that **man** searches for manual information (normally **/usr/man**). The search path is a colon-separated (:) list of directories in which manual subdirectories can be found; for example **/usr/local:/usr/man**. If the environment variable **MANPATH** is set, its value is used for the default path. If the **-k** or **-f** flag is used, the **-M** flag must precede it.

Files

/usr/man/cat?/*	Directories that **man** searches for preformatted pages.
/usr/man/whatis	Keyword data base.

Related Information

See the following commands: "apropos" in topic 1.1.22, "more, page" in topic 1.1.277, "whereis" in topic 1.1.535, and "catman" in topic 1.1.50.

1.1.259 mark

Purpose
Creates, modifies and displays message sequences.

Syntax

```
    +-----------------------------------------------------------------------
 ---¦                   +-------------+   +-------------+   +---------------+
    +- -sequence name --¦   one of    +---¦   one of    +---¦    one of     +-
                        ¦ +---------+ ¦   ¦ +---------+ ¦   ¦ +-----------+ ¦
                        +-¦ -add    +-+   +-¦ -zero   +-+   +-¦ -public   +-+
                          ¦ -delete ¦       ¦ -nozero ¦       ¦ -nopublic ¦
                          +---------+       +---------+       +-----------+

    +---------+
 ---¦         +---¦
    +- -list -+

mark --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **mark** command is used to create a sequence, delete a sequence, add messages to a sequence, and delete messages from a sequence. The **mark** command is also used to list messages in a sequence and list sequences in a folder. The **mark** command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **mark** command specified with only a folder name lists the sequences defined for that folder and the messages that comprise each of the sequences. If you specify a new sequence name with the **-sequence** flag, **mark** creates a new sequence. You can use the **-add** flag to add messages to a sequence and the **-delete** flag to remove message from a sequence. When all messages are deleted from a sequence, **mark** removes the sequence name from the folder.

Flags

-add Adds messages to the specified sequence. If messages are specified **-add** is the default.

-delete
Deletes messages from the specified sequence.

+*foldermsgs*
Specifies messages that you want **mark** to select from. *msgs* can be several messages, a range of messages, or a single message. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

If the **-list** flag is used, the default for *msgs*is **all**. Otherwise, the default is the current message. The default folder is the current folder. If you specify a folder, that folder becomes the current folder.

-help Displays help information for the command.

-list Displays the messages in the specified sequence. If you do not specify a sequence, **-list** displays all sequence names defined for the folder and the messages in each sequence.

-nopublic
Restricts the specified sequence to your usage. **-nopublic** does not restrict the messages in the sequence, only the sequence. This option is the default if the folder is write-protected from other users.

-nozero
Modifies the sequence by adding or deleting only the specified messages (see the **-zero** flag). This flag is the default.

-public
Makes the specified sequence available to other users. **-public** does not make protected messages available, only the sequence. This flag is the default if the folder is not write-protected from other users.

-sequence *.name*
Specifies a sequence for the **-list**, **-add**, and **-delete** operations. You must specify this flag for the **-add** and **-delete** operations.

-zero Clears the specified sequence of all messages before adding any other messages. When the **-delete** flag is also specified, **-zero** places all of the messages from the folder into the sequence before deleting any messages.

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.

Files

$HOME/.mh_profile	The MH user profile.

Related Information

See the MH command "pick" in topic 1.1.316.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.260 MasterInstall

Purpose
Installs one or more **LPP** programs from 9-track tape.

Syntax

MasterInstall ---¦

Note: This command is for the System/370 only.

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **MasterInstall** program is a menu-oriented user interface that provides the user with the ability to select the installation of one or more **LPP** program packages from an **LPP** tape. **MasterInstall** arranges for each selected **LPP** to be loaded from the tape and installed on the system.

The following **LPP** program packages may be installed by **MasterInstall**.

1 aadu -AIX Access for DOS Users

2 adt -Application Development Toolkit

3 ate -Asynchronous Terminal Emulation

4 osext -OS Extensions

5 fortran -FORTRAN

6 gsl -Graphics Support Library

7 ined -INed editor

8 inmail -INmail/INnet/INftp

9 merge DOS Merge

10 pascal -Pascal

11 tcp/ip -TCP/IP

12 tfs -Text Formatting System

13 use -Usability Services

14 whip Workstation Host Interface

15 x.25 -X.25

Other packages may be defined at some future date.

Related Information

See the following commands: "installp" in topic 1.1.212 and "update" in topic 1.1.497.

See the *AIX Operating System AIX/370 Installation and Customization Guide* for further details.

1.1.261 mdrc

Purpose
Reinstalls a user-created minidisk.

Syntax

For AIX PS/2

```
        +-------------------+    +------+
mdrc ---¦                   +----¦      +---¦
        +--- -h hdisknum ---+    +- -v -+
                          ¦
               +----------+
```

For AIX/370

```
        +-----------------+    +------+
mdrc ---¦                 +----¦      +---¦
        +--- -h chdnum ---+    +- -v -+
                fhdnum  ¦
               +--------+
```

Description
The **mdrc** command provides access to user-created minidisks. You should run this command if you have reinstalled the AIX Operating System or if you have had to replace the **/etc/system** or **/etc/filesystems** files with copies that do not contain stanzas describing any user-installed minidisks. The system uses the information in these stanzas to configure the minidisks at system startup, and **mdrc** recreates the necessary stanzas. The **mdrc** command uses the VTOC (volume table of contents) on the hard disk to recreate the necessary stanzas.

If the **mdrc** command cannot recreate the original **/etc/filesystems** stanza for AIX Operating System minidisks, it assigns attributes of **Auto Mount=no**, **Read/Write Status=R/W**, and **Mount Directory=/tmp/directory/hdn** to the minidisk. In this case, you should then run the **minidisk** command to change the attributes to the values you want. You might also need to run the **mkdir** command to create the mount directory, if you reinstalled the entire AIX Operating System.

You must have superuser authority or be a member of the system group to run the **mdrc** command.

Flag

-h	Specifies any disks that have been removed or damaged and tells the **mdrc** command to remove the minidisk configuration entries for these disks. If you do not specify this flag and a disk is not configured, **mdrc** ignores entries in the configuration files for the disk's minidisks. Use **chdnum** for ckd drives and **fhdnum** for fda drives on AIX/370s.
-v	Verbose option.

Files

/etc/filesystems	Descriptions of the mountable file systems.

/etc/system	Default system configuration file.

Related Information
See the following commands: "minidisks" in topic 1.1.266 and "mkdir" in topic 1.1.268.

See the **filesystems** and **system** files in *AIX Operating System Technical Reference*.

1.1.262 mesg

Purpose
Allows or prevents incoming **write** command messages.

Syntax

```
        +--------+
mesg ---¦ one of +---¦
        ¦ +---+  ¦
        +-¦ n +--+
          ¦ y ¦
          +---+
```

Description
The **mesg** command controls whether other users on the system can send messages to your work station with the **write** or **talk** command. Called without arguments, **mesg** displays the current work station message-permission setting. The shell startup process permits messages by default. You can override the default by including the line **mesg n** in the **$HOME/.profile** file (for **sh** users), or in the file **$HOME/.login** (for **csh** users). A user with superuser authority can send **write** messages to any work station, regardless of its message permission setting. Message permission has no effect on messages delivered through the electronic mail system (**sendmail**).

Note: The **mesg** command does not affect the mail notification (**biff** command).

Flags

n	Disables incoming **write** messages. Use this form of the command to prevent your display from being cluttered with incoming messages.
y	Permits incoming **write** command messages.

Files

/dev/tty*	Default terminal devices.

Related Information
See the following commands: "write" in topic 1.1.541, "sendmail, mailq, newaliases" in topic 1.1.417, and "biff" in topic 1.1.41.

1.1.263 mhl

Purpose
Produces formatted listings of messages.

Syntax

```
                         +--------------+   +-------------------+
/usr/lib/mh/mhl ---¦                +---¦                   +---
                         +- -form file -+   +- -folder +folder -+

    +-----------------------------------------------------------------------
 ---¦ +---- -moreproc cmd ---------------------------------------------------
    +-¦                 +--- -bell ---+   +-- -noclear --+   +---------------+
      +- -nomoreproc --¦   one of    +---¦    one of    +---¦               +-
                        ¦ +---------+ ¦   ¦ +----------+ ¦   +- -length num -+
                        +-¦ -bell   +-+   +-¦ -clear   +-+
                          ¦ -nobell ¦       ¦ -noclear ¦
                          +---------+       +----------+

 --- file ---¦
          ¦
  +--------+

/usr/lib/mh/mhl --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **mhl** command is used to create formatted lists of messages. The **mhl** command is part of the Message Handling (MH) package and can be used with other MH and AIX commands. **mhl** is usually invoked through the profile entry **showproc:** or through the **-showproc** flag in other MH commands.

The **mhl** command uses the formatting directions listed in the format file to display the message information about each of the specified messages. If you specify more than one message, **mhl** provides a prompt before displaying each screen of messages or with **-nomoreproc**, before displaying each message. If you specify **-nomoreproc**, press **Enter** or **END OF FILE** to see the next message. Press **INTERRUPT** to stop the current message output and to receive a prompt for the next message. Press **QUIT** to stop the command output.

Flags

-bell	Produces an audible indicator at the end of each page. This flag affects **mhl** only if the output device is a display and the **moreproc:** entry is defined and empty. This flag is the default.
-clear	Clears the screen at the end of each page when the output device is a display. When the output device is not a display, **-clear** inserts a formfeed character at the end of each message. If the output device is a display, **mhl** uses the **$TERM** and **$TERMCAP** environment variables to determine the type of display. This flag affects **mhl** only if the **moreproc:** entry is defined and is empty.

-digest	Creates a digest format for the message output.
-folder +*folder*	Specifies the folder to be used for the **mhl.format messagename** entry. The default is the value of the **$MHFOLDER** environment variable.
-form *file*	Uses the format contained in the specified file. If you do not specify this flag, **mhl** uses the format described in **user_mh_directory/mhl.format**. If this file does not exist, **mhl** uses the system default format described in **/usr/lib/mh/mhl.format**.
-forward	Places a blank after the leading dashes (-) at the start on an output line.
-forwall	Forwards message.
-help	Displays help information for the command.
-length *num*	Sets the length of the output. The default value is the value indicated by **$TERM**. If that value is not appropriate, the default value is 40.
-moreproc *cmd*	Uses *cmd*instead of the value of the **moreproc:** entry specified in **$HOME/.mh_profile**.
-nobell	Does not produce an audible indicator at the end of each page. This flag affects **mhl** only if the output device is a display and the **moreproc:** entry is defined and is empty.
-noclear	Does not clear the screen at the end of each page when the output device is a display. When the output device is not a display, **-clear** does not insert a formfeed character at the end of each message. This flag affects **mhl** only if the **moreproc:** entry is defined and is empty. This flag is the default.
-nomoreproc	Sets **moreproc:** as an empty value.
-width *num*	Sets the width of the output. The default value is the value indicated by **$TERMCAP**. If that value is not appropriate, the default value is 80.

Profile Entry

moreproc:	Specifies the interactive program for communicating with user.

Files

/usr/lib/mh/mhl.format	The default MH message template.
user_mh_directory/mhl.format	The user's default message template. (If it exists, it overrides the default MH message template.)
$HOME/.mh_profile	The MH user profile.

Related Information

See other MH commands: "ap" in topic 1.1.20, "dp" in topic 1.1.138, "next" in topic 1.1.293, "prev" in topic 1.1.323, and "show" in topic 1.1.423.

See the **mh-format** and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.264 mhmail

Purpose
Sends or receives mail.

Syntax

```
mhmail ---¦

                 +------------------+   +-------------+   +-------------
mhmail --- user ---¦                  +---¦             +---¦
               ¦ +--- -cc --- user ---+   +- -from user -+   +- -subject st
       +--------+                  ¦
                          +--------+

    +---------------+
 ---¦               +---¦
    +- -body string -+

mhmail --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **mhmail** command is used to incorporate messages and compose messages. The **mhmail** command is part of the Message Handling (MH) package and can be used with MH and AIX commands.

The **mhmail** command entered by itself incorporates messages from your mailbox. If you specify user addresses, **mhmail** accepts text from your terminal and composes a message. You can end the message text by pressing the **END OF FILE** key. The **mhmail** command sends a copy of the message to each specified address.

Flags

-body *string*	Sends a message with *string* as the body. When you specify **-body**, the **mhmail** command does not accept text from the terminal.
-cc users	Sends a copy of the message to the specified **users**. The **mhmail** command puts the addresses in the **cc:** field of the message.
-from *user*	Places the specified **user** address in the **From:** field of the message.
-help	Displays help information for the command.
-subject *string*	Places the specified text **string** in the **Subject:** field of the message.

Files

$HOME/$USER	The location of the mail drop.

Related Information

See other MH commands: "inc" in topic 1.1.206 and "post" in topic 1.1.320.

See the **mh-alias**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.265 mhpath

Purpose
Prints full path names of messages and folders.

Syntax

```
          +-----------+   +---------------------------------------------------
mhpath ---¦           +---¦ +-------- all ------------------------------------
          +- +folder -+   +-¦ +---------- new --------------------------------
                            +-¦   +------------ sequence ------------+   +----
                              +---¦  one of                          +---¦
                                   ¦ +-------+   +------------------+ ¦ ¦ +- ne
                                ¦ +-¦ num   +---¦       one of      +-+ ¦
                                ¦   ¦ first ¦   ¦¦+--------------+ ¦    ¦
                                ¦   ¦ prev  ¦   +-¦ :num   -prev +-+    ¦
                                ¦   ¦ cur   ¦     ¦ :+num  -cur  ¦      ¦
                                ¦   ¦ .     ¦     ¦ :-num  -.    ¦      ¦
                                ¦   ¦ next  ¦     ¦ -num   -next ¦      ¦
                                ¦   ¦ last  ¦     ¦ -first -last ¦      ¦
                                ¦   +-------+     +--------------+      ¦
                                +---------------------------------------+

mhpath --- -help ---¦


-----------------
¦ Do not put a blank between these items.
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **mhpath** command is used to list the path names of folders and messages. This command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **mhpath** command lists the path names of all specified messages. If you do not specify any messages, **mhpath** lists the path name of the folder. If you do not specify messages or a folder, **mhpath** lists the path name of the current folder.

Flags

+*folder msgs* Specifies the folder or the messages for which you want to list path names. *msgs* can be several messages, a range of messages, or a single message. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
.	**next**	**last**
all	**sequence**	

You cannot use **new** in a message range.

If you do not specify a message, **mhpath** lists the path name of the specified folder. The default folder is the current folder.

-help	Displays help information for the command.

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.

Files

.profile	
$HOME/.mh_profile	The MH user profile.

Related Information

See the **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.266 minidisks

Purpose
Adds, deletes, changes and displays minidisks.

Syntax

```
minidisks ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **minidisks** command lets you add, delete, show, or change characteristics of a minidisk. To use the **minidisks** command, you must be a member of the system group or have superuser authority. You may change minidisks only when the primary site of the root file system is in the cluster.

The **minidisks** command is menu-driven. For information on how to use it, see *Installing and Customizing the AIX Operating System*.

Files

/dev	Directory.
/tmp	Directory.
/etc/ddi	Directory.
/etc/master	Default master configuration file.
/etc/system	Default system configuration file.
/etc/mdkaf	Default **minidisk** configuration file.
/etc/filesystems	Descriptions of mountable file systems.

Related Information
See "mkdir" in topic 1.1.268 and "mdrc" in topic 1.1.261.

See the discussion of **minidisks** in *Installing and Customizing the AIX Operating System*.

1.1.267 mkcatdefs

Purpose

Processes symbolic message identifiers in message text source files.

Syntax

```
                                                            +------------+
mkcatdefs --- -h --- catname --- msg-file --- cmd-name --¦            +-¦
                                                            +- descfile -+
```

Description

The **mkcatdefs** command provides a mechanism for specifying message set numbers and message identification numbers symbolically within a user program and within the message catalog descriptor file used as input to the **gencat** command. The **mkcatdefs** command preprocesses a descriptor file containing ASCII symbolic message identifiers (set and message numbers), producing a descriptor file that has these symbols replaced by numeric values for the set and message numbers. A header file is also produced and can be included within a user program. The header file contains all the symbolic-to-numeric definitions (in the form **#define symbol value**) required to reference the message symbolically.

The **-h** option is used to suppress a header file. The **catname** parameter is the prefix used to create the header file (for example, **catname-msg.h**). The **msg-file** parameter contains the message text source file with symbolic identifiers.

The **cmd-name** parameter generates a name for the header file (**cmd-name_msg.h**). The **cmd-name** must be an ASCII file name. The **descfile** parameter specifies a descriptor file containing the symbolic definitions in place of the set and message numbers (by convention the name of **descfile** is **cmd-name.msg**). In addition to the header file created by **mkcatdefs**, the modified descriptor file (with numeric values in place of symbolic values) is written to standard output.

The header file then can be included in programs that access the messages within the catalog symbolically rather than numerically. The descriptor file written to standard output should be given to the **gencat** command to produce the actual catalog.

Related Information

See the following command: "gencat" in topic 1.1.184.

See the **catgets** and **NLgetamsg** subroutines in the *AIX Technical Reference*.

See the "Message Catalog Generation" section of the "International Character Support" chapter of the *AIX Operating System Programming Tools and Interfaces*.

See the "Messages" chapter in the *AIX MBCS Guide*.

1.1.268 mkdir

Purpose
Makes a directory.

Syntax

```
       +------+
mkdir -¦      +-- directory --¦
       +- -h -+               ¦
                +-----------+
```

Description
The **mkdir** command makes a new directory. The read, write, and execute permissions of the new directory will depend on the current umask value. You can change the default permissions the command sets with the **umask** command (see page 1.1.490). The **mkdir** command also creates, by default, the standard entries **.** (dot) for the directory itself and **..** (dot dot) for its parent.

If you specify unsupported options, the **mkdir** command creates a directory whose name consists of the unsupported option. For example, **mkdir -sdlkif** creates a directory named **-sdlkif**. This allows you to include a minus sign in the directory name.

Note: To make a new directory, you must have write permission in the parent directory.

Flags

-h Creates a hidden directory instead of an ordinary directory.

Related Information
See the following commands: "sh, Rsh" in topic 1.1.420, "rm, delete" in topic 1.1.375, and "umask" in topic 1.1.490.

See the **mkdir** system call in *AIX Operating System Technical Reference*.

1.1.269 mkfs

Purpose
Makes a file system.

Syntax

```
                   +--------------------+   +- -p/stand/boot -+
mkfs --- device ---¦ +---------------+ +---¦                 +---
                   +-¦ -ffilesystem  +-+   +--- -pprogram ---+
                     ¦ -scyl:ski     ¦¦
                    ¦¦ -vvolume      ¦¦
                    ¦¦ -l            ¦¦
                    ¦¦ -r            ¦¦
                    ¦¦ -ggfs         ¦¦
                    ¦¦ -Pgfspack     ¦¦
                    ¦+---------------+¦
                    +-----------------+

    +--------------------+
 ---¦       one of       +---¦
    ¦ +---------------+ ¦
    +-¦ blocks        +-+
      ¦ blocks:inodes ¦
      ¦ prototype     ¦
      +---------------+
```

Description
The **mkfs** command makes new file systems. This command initializes the volume label and file system label, the startup block, and the bad-block list. It also interleaves the free list in accordance with the flags or with defaults found in the **/etc/filesystems** file.

The **mkfs** command creates the new file system on the **device** specified on the command line. The **device** can be a block device name, raw device name, or file system name. If **device** is a file system name, **mkfs** uses this name as the **filesystem** and uses the following parameters from the applicable stanza in the file **/etc/filesystems**:

dev	Device name.
cyl	See the **-s** flag on page 1.1.269.3.
skip	See the **-s** flag on page 1.1.269.3.
vol	Volume ID.
bad	List of bad blocks separated by commas.
size	File system size.
boot	Program to be installed in the startup block.
gfs	Global file system number to be assigned to this file system.
gfspack	Global file system pack number to be assigned to this file system if it is to be replicated (used with **-r** option).

Subtopics

1.1.269.1 File System Size

You can specify the size of a new file system in the following ways:

- On the command lin
- In the **prototype** file
- In the **/etc/filesystems** entry for the given file system.

If the size is not specified in any of these places, **mkfs** takes it from the **devinfo** structure for the block device associated with the file system being generated. (See the **ioctl** system call and the **devinfo** file in *AIX Operating System Technical Reference*.) The size provided in the **devinfo** structure is the maximum size of the file system in any case. If you specify a file system size larger than that returned in the **devinfo** structure, the **devinfo** value is used instead. Assuming you do not specify a size larger than the size provided in **devinfo**, a size specification on the command line overrides any defaults found in **devinfo** or in the file **/etc/filesystems**.

1.1.269.2 Global File System Numbers

Each file system must be given a global file system number (gfs) to distinguish it from other file systems mounted on that machine, or, if the Transparent Computing Facility is installed, to distinguish it from other file systems mounted in the same cluster. If its value is not 0, the global file system number can be in the range of 1 to a system-configurable value, which defaults to 300. A gfs value of 0 can be used to indicate that the system should assign a gfs number each time it is mounted. This is intended primarily for use for file systems created on a removable medium and may not be used for replicated file systems.

You can specify the gfs number of a new file system in the following ways:

On the command line with the **-ggfs** option.

In the **gfs** entry for this file system in the **/etc/filesystems** file.

If the gfs value not specified in one of these ways, **mkfs** prompts the user for it.

In addition to the global file system number, if the file system is created to be a replicated file system with the **-r** option, a global file system pack number (gfspack) number must be specified. The global file system pack number can be in the range of 1 to 32. Each copy of a replicated file system must have a unique number. The global file system pack number can be specified on the command line (**-Pgfspack** option) or in the **/etc/filesystems** file (**gfspack** entry). If it is not specified in one of these places, **mkfs** prompts the user for it.

1.1.269.3 Prototype Files

To initialize the contents of a new file system in accordance with a prototype, specify the name of a **prototype** file on the command line. The **proto** command can be used to construct prototype files from existing file systems.

The **prototype** file contains tokens separated by spaces or new-line characters. The first token is the name of a file to be copied onto block 0 as the bootstrap program. The second token is a number specifying the size of the created file system. Typically, it is the number of blocks on the device, perhaps diminished by space for paging. The next token is the number of inodes in the i-list. (The **mkfs** command rounds this to fill out the appropriate number of blocks.) The next set of tokens contains the specifications for the root file. File specifications consist of tokens giving the mode, the user name, the group name, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a six-character string. The first character specifies the type of the file. (The characters **-**, **b**, **c**, **s**, and **d** specify regular, **b**lock special, **c**haracter special, **s**ymbolic link, and **d**irectory files, respectively.) The second character must be either **u** or **-**. If **u** is used, the set-user-ID mode is specified; if **-** is used, the set-user-ID mode is not specified. The third character must be either **g** or **-** for specifying the set-group-ID mode. The rest of the mode is a three-digit octal number giving the owner, group, and other read, write, execute permissions (see "chmod" in topic 1.1.67).

Two decimal number tokens come after the mode. They specify the user and group names of the owner of the file. If the file is a regular file, the next token is a path name from which the contents and size are copied. If the file is a block or character special file, two decimal number tokens follow, giving the major and minor device numbers. If the file is a symbolic link, the next token is the path name that is to be used as the contents of the symbolic link.

If the file is a directory, **mkfs** makes the entries **.** (dot) and **..** (dot dot) and then recursively reads a list of names and file specifications for the entries in the directory. The scan is ended with the token **$** (dollar sign).

The **mkfs** command can also be run noninteractively, taking its directions from a "here" document or an input file (rather than the command line). When **mkfs** is run noninteractively, it assumes "yes" answers to the following questions for which it would normally prompt you:

```
  mkfs: destroy DEVICE (y)?
  inode count OK:(y/n)
```

A "yes" answer to the first question means that **mkfs** automatically overwrites any pre-existing file system with the name specified on the command line. A "yes" answer to the inode-count question means **mkfs** has calculated an inode count and assumes it is correct.

Flags

-ffilesystem Specifies the file system label for the new file system. The label can be up to 32 characters. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the

	32-character limit may be reduced by as much as 50%, depending on the character code set being used.
-ggfs	The global file system number to be assigned to this file system. The overriding value, if any, is found in the file **/etc/filesystems**.
-l	Creates a **lost+found** directory for the file system. The allows the **fsck** command to take corrective actions when certain types of file system inconsistency problems are discovered.
-pprogram	Specifies the name of a program to be installed in block 0 of the new file system. The default bootstrap program is **/stand/boot**.
-Pgfspack	The global file system pack number to be assigned to this file system (meaningful only if the file system is replicated). The overriding value, if any, is found in the file **/etc/filesystems**.
-r	Creates a replicated file system.
-scyl:skip	Specifies an interleaving of the free list. (Interleaving the free list can improve the speed of disk I/O.) The **cyl** argument is the number of blocks per cylinder, and **skip** is the number of blocks to skip.
-vvolume	Specifies the volume label for the new file system. The **volume** argument can be up to 8 characters.
blocks[:inodes]	A size specification where **blocks** is the number of 1024-byte blocks in the file system. When **inodes** is specified, it determines the number of inodes on the system. If **inodes** is not specified, a number suitable for the size of the file system is used. The number of inodes is rounded up so that the inode area occupies an integral number of blocks.

Examples

(Note: The following two examples do not apply to AIX/370.)

1. To create an empty file system on a diskette:

```
mkfs  /dev/fd0
```

2. To specify volume and file system names for a new filesystem:

```
mkfs  /dev/fd0  -fWORKFS  -vVOL001
```

   This command creates an empty file system on the diskette, giving it the volume serial **VOL001** and file system name **WORKFS**.

Related Information
See "fsck, dfsck" in topic 1.1.177.

See the **ioctl** system call and the **devinfo**, **dir**, **filesystems**, and **fs** files in *AIX Operating System Technical Reference*.

1.1.270 mklost+found

Purpose

Creates a directory with empty files for **fsck**.

Syntax

```
/etc/mklost+found ---¦
```

Description

A directory **lost+found** is created in the current directory. A number of empty files are created therein and then removed so that there are empty slots for the **fsck** command. This command should not normally be needed since the **mkfs** command automatically creates the **lost+found** directory when a new file system is created.

Related Information

See the following commands: "fsck, dfsck" in topic 1.1.177 and "mkfs" in topic 1.1.269.

1.1.271 mknod

Purpose
Creates a special file.

Syntax

```
                         +--- c ---+
                       +-¦         +- major --- minor -+
                       ¦ ¦ +-----+ ¦                   ¦   +---------------+
mknod --- device ---¦ +-¦  b  +-+                   +---¦ +-----------+ +---¦
                       ¦   ¦  c  ¦                     ¦   +-¦           +-+ ¦
                       ¦   +-----+                     ¦ ¦   +- sitename -+   ¦
                       +------------- p -------------+ +--------------------+
```

Description
The **mknod** command makes a directory entry and corresponding inode for a special file. The first parameter specifies the name of the entry **device**.

The **mknod** command has two forms. In the first form, the second parameter is **b** or **c**. The **b** flag indicates the special file is a block-oriented device (disk, diskette, tape). The **c** flag indicates the special file is a character-oriented device (other devices). The next two parameters are numbers specifying the **major** device, which helps the operating system find the device driver code; and the minor device, that is, the unit drive, or line number, which can be either decimal or octal. The final optional parameter, **sitename**, specifies the cluster of sites where the device is located. If no site name is specified, the newly created **specid** file refers to a device on the site where this command is run.

The assignment of major device numbers is specific to each system. Device numbers are determined by examining the system file **/etc/master**.

Note: If you change the contents of **/etc/master** to add a device driver, you must rebuild the operating system. See the discussion of device drivers in *AIX Operating System Technical Reference*.

The second form of **mknod** is used to create FIFOs (named pipes). The **p** flag after **device** indicates you are creating a named pipe. See the *AIX Operating System Technical Reference* for an explanation of FIFOs and named pipes.

Flags

b Indicates the special file is a block-oriented device.

c Indicates the special file is a character-oriented device. This is the default.

p Creates a named pipe.

Examples

To create the special file for a new diskette drive:

```
  mknod  /dev/fd2  b  1  2
```

This creates the special file **/dev/fd2**, which is a block special file with major device number **1** and minor device number **2**, and refers to a device on the local cluster site.

Related Information

See the **mknod** system call and discussion of device drivers in *AIX Operating System Technical Reference*.

1.1.272 mkpasswd

Purpose

Generates hashed password table.

Syntax

```
            +------+
mkpasswd ---¦      +--- passwdfile ---¦
            +- -v -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **mkpasswd** command is used to generate the hashed password data base used by the subroutines **getpwnam** and **getpwuid**. If the **-v** option is supplied, each entry is listed as it is added. The **passwdfile** file is usually **/etc/ptmp** (invoked by the **vipw** command, and in any case must be in the format of **/etc/passwd**). The **mkpasswd** command generates data base files named **passwdfile.pag** and **passwdfile.dir**. The **mkpasswd** command exits with a non-zero exit code if any errors are detected.

Note: This command requires that the *passwd.dir* and *passwd.pag* files are removed before running.

Files

passwdfile.pag	Data base file
passwdfile.dir	Data base file

Related Information

See the following commands: "passwd, chfn, chsh" in topic 1.1.312, "vipw" in topic 1.1.523, "yppasswd" in topic 1.1.552 and "adduser, users" in topic 1.1.15.

See the **passwd** file format and the **getpwuid** and **getpwnam** subroutines in the *AIX Technical Reference*.

1.1.273 mkstr

Purpose
Creates files of error messages.

Syntax

```
        +-----+
mkstr ---¦     +--- messagefile --- prefix --- file ---¦
        +- - -+                                       ¦
                                               +--------+
```

Note: This command does not have MBCS support.

Description

The **mkstr** command is used to create files of error messages. It can make programs with large numbers of error diagnostics smaller and reduce system overhead since the error messages do not have to be swapped constantly.

The **mkstr** command processes each specified **file**, placing a messaged version of the input file in a file whose name consists of the specified **prefix** and the original name. A typical usage of the **mkstr** command would be:

```
  mkstr pistrings xx *.c
```

This example causes all error messages from the C source files in the current directory to be placed in the file **pistrings** and processed copies of the source for these files to be placed in files whose names are prefixed with **xx**.

To process the error messages in the source to the message file, the **mkstr** command keys on the string **%'error("'** in the input stream. Each time the string occurs, the C string starting at **'"'** is placed in the message file, followed by a null character and a new-line character. The null character terminates the message so it can be used easily when retrieved. The new-line character makes it possible to use the **cat** command to display the contents of the error message file.

The messaged copy of the input file then contains an **lseek** pointer into the file, which can be used to retrieve the message. For example:

```
  char efilname[] =  "/usr/lib/pi_strings"
  int    efil = -1;

  error(a1, a2, a3, a4)
  {
      char buf[256];
      if (efil < 0) {
          efil = open(efilname, 0);
          if (efil < 0) {
  oops:
              perror(efilname);
              exit(1);
          }
      }
      if (lseek(efil, (long) a1, 0) || read(efil, buf, 256) <= 0)
          goto oops;
      printf(buf, a2, a3, a4);
  }
```

The optional **-** causes the error messages to be placed at the end of the specified message file for recompiling part of a large **mkstr** program.

Related Information

See the following command: "xstr" in topic 1.1.546.

1.1.274 mm, checkmm

Purpose
Displays documents formatted with the memorandum macros.

Syntax

```
      +-------------------+   +---- - -----+
mm ---¦ +---------------+ +---¦            +---¦
      +-¦ -12           +-+   +--- file ---+
        ¦ -c            ¦¦             ¦
       ¦¦ -e            ¦¦      +--------+
       ¦¦ -E            ¦¦
       ¦¦ -t            ¦¦
       ¦¦ -Tworkstation ¦¦
       ¦+---------------+¦
       +-----------------+


           +--------+
checkmm ---¦        +---¦
           +- file -+ ¦
         ¦            ¦
         +------------+
```

Note: This command does not have MBCS support.

Description
Using the **nroff** command and the memorandum macros text-formatting package, the **mm** command writes files to standard output. If you specify a **-** (minus) in place of the **file** parameter, the command reads standard input. Do not specify both file names and standard input.

The **mm** command has flags to specify preprocessing by the **tbl** and/or **eqn** commands and post-processing by various work station-oriented output filters. The **mm** command generates the proper pipelines and required parameters for the **nroff** command, depending on the flags selected.

The **checkmm** command checks the contents of the specified **file**s for errors in the use of the memorandum macros and some **eqn** and **neqn** command constructions. The **checkmm** command skips all directories. If you do not specify a **file** parameter, the **checkmm** command reads standard input.

Notes:

1. Use the **-olist** argument of the **nroff** command to specify ranges of pages for output. However, invoking **mm** with one or more of the **-e**, **-t**, and **-** minus flags together with the **nroff -olist** parameter may cause a harmless **broken pipe** diagnostic if the last page of the document is not specified in the **list** variable.

2. The **mm** command calls **nroff** with the **-h** flag. The **nroff** command assumes the work station has tabs set every eight character positions.

3. If you use the **-s** flag of the **nroff** command to stop between pages of output, use a line feed rather than **Enter** or a new-line character to restart the output. The **-s** flag of **nroff** is not compatible with the **-c** flag of the **mm** command; nor is it compatible with the **mm** command if **mm** automatically calls the **col** command.

4. Providing inaccurate information to **mm** about the kind of work station its output is to be printed on yields unsatisfactory results. If you are redirecting output to a file, use the **-T37** flag and then use the appropriate work station filter when you actually print the file.

Flags

Any flags on the command line not listed below are passed to the **nroff** command or to memorandum macros. The flags can occur in any order but they must come before the **file** parameter.

-c	Invokes the **col** command, which is invoked automatically by the **mm** command unless the **workstation** variable is one of the following: 300 300s 450 37 4000a 382 4014 tek 1620 X
-e	Invokes the **neqn** command.
-E	Invokes the **-e** flag of the **nroff** command.
-t	Invokes the **tbl** command.
-Tworkstation	Specifies the work station. To display a list of recognized values for the **workstation** variable, enter: `help term1` By default, the **mm** command uses the value of the shell variable **$TERM** from the environment as the value of **workstation**. If **$TERM** is not set, the **mm** command uses the **lp** command. If several work station types are specified, the last one listed takes effect.
-12	Specifies 12-pitch font, which can be used when **$TERM** is set to 300, 300s, 450, or 1620. The pitch switch on the DASI 300 and 300s work stations must be manually set to 12 if this flag is used.

Related Information

See the following commands: "col" in topic 1.1.77, "env, printenv" in topic 1.1.151, "eqn, neqn, checkeq" in topic 1.1.152, "mmt, mant, mvt" in topic 1.1.275, "nroff, troff" in topic 1.1.301, and "tbl" in topic 1.1.463.

See the **profile** file and the **eqnchar**, **mm**, and **term** miscellaneous facilities in *AIX Operating System Technical Reference*.

See the discussion of the **mm** command in *Text Formatting Guide*.

1.1.275 mmt, mant, mvt

Purpose
Typesets documents, manual pages, view graphs, and slides.

Syntax

```
 one of
+------+   +------------+   +-----------+   +----- - -----+
¦ mmt  +---¦   one of   +---¦ +-------+ +---¦             +---¦
¦ mant ¦   ¦ +--------+ ¦   +-¦ -c -e +-+   +--- file¦ ---+
¦ mvt  ¦   +-¦ -T4014 +-+     ¦ -t -a ¦¦              ¦
+------+     ¦ -Ttek  ¦      ¦+-------+¦      +----------+
             +--------+      +---------+
```

```
----------------
¦ If no files are given, these commands display their flags.
```

Note: This command does not have MBCS support.

Description
These commands are similar to the **mm** command except they typeset their input via the **troff** rather than **nroff** command. The **mvt** and **mant** commands are links to **mmt**. The **mmt** command uses the memorandum macro package (see the **mm** facility in *AIX Operating System Technical Reference*). The **mvt** command uses the macro package for view graphs and slides (see the **mv** facility in *AIX Operating System Technical Reference*). The **mant** command uses the manual page macros.

These commands have flags to specify preprocessing by the **tbl**, **cw**, or **eqn** commands. The **mmt** command generates the proper pipelines and required parameters for the **troff** command and the macro package used, depending on the flags selected. If you specify a **-** (minus) in place of the **file** variable, these commands read standard input.

The **checkmm** command can be used to check the input to **mmt**.

If the input contains a **troff** command comment line consisting solely of the string **'\" x** (single quotation mark, backslash, double quotation mark, space, **x**), where **x** is any combination of the three letters **c**, **e**, and **t**, and where there is exactly one space between the double quotation mark and **x**, the input is processed through the appropriate combination of the **cw**, **eqn**, and **tbl** commands, respectively, regardless of the command line parameters.

Note: Use the **-olist** parameter of the **troff** command to specify ranges of pages for output. However, calling these commands with one or more of the **-c**, **-e**, **-t**, and **-** flags together with **troff -olist** may cause a harmless **broken pipe** diagnostic if the last page of the document is not specified in the **list** variable.

Flags

Flags other than the ones listed below are passed to the **troff** command or macro package. All flags must appear before the **file** variable. If you do not provide any parameters, these commands print a list of their flags.

-a	Invokes the **-a** flag of the **troff** command.

-c	Preprocesses the input files with the **cw** command.
-e	Preprocesses the input files with the **eqn** command.
-t	Preprocesses the input files with the **tbl** command.
-T4014	Directs the output to a Tektronix 4014 work station via the **tc** command.
-Ttek	Same as **-T4014** flag.

Related Information

See the following commands: "env, printenv" in topic 1.1.151, "eqn, neqn, checkeq" in topic 1.1.152, "mm, checkmm" in topic 1.1.274, "tbl" in topic 1.1.463, "tc" in topic 1.1.464, and "troff" in topic 1.1.302.2.

See the **profile** file and the **environ**, **mm**, and **mv** miscellaneous facilities in *AIX Operating System Technical Reference*.

1.1.276 moo

Purpose
Plays a number-guessing game.

Syntax

/usr/games/moo ---¦

Note: This command does not have MBCS support.

Description

The **moo** command picks a random four-digit decimal number with nonrepeating digits. You guess four digits and score a "cow" with a correct digit in an incorrect position and a "bull" with a correct digit in a correct position. The game continues until you guess the number. To quit the game, press **Interrupt** or **Ctrl-D**.

1.1.277 more, page

Purpose
Acts as the perusal filter for screen viewing.

Syntax

```
 one of
+------+   +----------------+
¦ more +---¦ +-------------+ +--- filename ---¦
¦ page ¦   +-¦ -c       -n +-+              ¦
+------+     ¦ -d       -p ¦¦  +------------+
            ¦¦ -f       -s ¦¦
            ¦¦ -l       -u ¦¦
            ¦¦ +linenumber ¦¦
            ¦¦ +/pattern   ¦¦
            ¦+-------------+¦
            +---------------+
```

Description

The **more** command is a filter that allows examination of a continuous text, one screen at a time. The program normally pauses after each screen and displays the word **more** at the bottom of the screen. If the user then types a carriage return, one more line is displayed. Pressing the space bar displays another screen of information.

If the program is invoked as **page**, the screen is cleared before each screen is printed (but only if a full screen is being printed), and **k**-1 rather than **k**-2 lines are printed in each screen, where **k** is the number of lines the terminal can display.

The **more** command looks in the file **/etc/termcap** to determine terminal characteristics and to determine the default window size. On a terminal capable of displaying 24 lines, the default window size is 22 lines.

The **more** command looks in the environment variable **MORE** to pre-set any flags desired. For example, if you prefer to view files using the **-c** mode of operation, the **csh** command **setenv more -c** or the **sh** command sequence **more='-c'; export more** causes all invocations of the **more** command including invocations by such programs as **man** to use this mode. Normally, the user places the command sequence which sets up the **MORE** environment variable in the **cshrc** or **.profile** file.

If the **more** command is reading from a file, rather than a pipe, a percentage is displayed along with the **--More** prompt. This gives the fraction of the file (in characters, not lines) that has been read so far.

Flags

-n Determines the size (in lines) of the window, where **-n** is an integer.

-c Draws each page by beginning at the top of the screen by erasing each line just before the **more** command draws on it. This avoids scrolling the screen and makes it easier to read while the command is writing. This option is ignored if the terminal does not have the ability to clear to the end of a line.

-d Prints messages for novice users.

-f Causes **more** to count logical lines rather than screen lines. That is, long lines are not folded. This option is useful if **nroff** output is being piped through **ul** since **ul** may generate escape sequences. These escape sequences contain characters which would ordinarily occupy screen positions but do not print. Thus, **more** may think that lines are longer than they actually are and fold line erroneously.

-l Ignores a form feed (***ctr*l-L**); that is, **more** does not pause after any line that contains a **Ctrl-L** and does not clear the screen if a file begins with a **Ctrl-L**.

-p Clear the screen before displaying a new screen.

-s Removes multiple blank lines from the output, producing only one blank line. This options is useful when viewing **nroff** output, maximizing the amount of useful information that can be viewed on the screen.

-u Normally, **more** handles underlining produced by **nroff** in a manner appropriate to the particular terminal. If the terminal can perform underlining or has a stand-out mode, **more** outputs the appropriate escape sequences to enable underlining or stand-out mode for underlined information in the source file. The **-u** option suppresses this processing.

+linenumber Starts up at **linenumber**.

+/pattern Starts up two lines before the line containing the regular expression **pattern**.

Other sequences which may be typed when **More** pauses, and their effects, follow (**i** is an optional integer argument, defaulting to **1** (the numeral 1)):

i<space> Displays **i** more lines (or another screen if no argument is given). If **stty** speed is set to less than 1200 baud, you must follow **i<space>** with a **<CR>** (carriage return).

Ctrl-B Scrolls backward one screen.

b Scrolls backward one screen. Same as **Ctrl-B**.

Ctrl-D Displays 11 more lines (a "scroll"). If **i** is given, the scroll size is set to **i**.

d Same as **Ctrl-D**.

i z Same as typing a space except that **i**, if present, becomes the new window size.

i s Skips **i** lines and prints a screenful of lines.

i f Skips **i** screenfuls and prints a screenful of lines.

q or **Q** Exits from **more**.

= Displays the current line number.

v	Starts up the editor **vi** at the current line.
h	Help commands; gives a description of all the **more** commands.
i/expr	Searches for the **i**-th occurrence of the regular expression **expr**. If there are less than **i** occurrences of **expr** and the output is a file (rather than a pipe), the position in the file remains unchanged. Otherwise, a screenful is displayed, starting two lines before the place where the expression is found. The user's erase and kill characters may be used to edit the regular expression. Erasing back past the first column cancels the search command.
in	Searches for the **i**-th occurrence of the last regular expression entered.
Ctrl-L	Redraws the screen.
'	(Single quote.) Goes to the point from which the last search started. If no search has been performed in the current file, this command goes back to the beginning of the file.
!command	Invokes a shell with **command**. The characters **%** and **!** in **command** are replaced with the current file name and the previous shell command, respectively. If there is no current file name, **%** is not expanded. The sequences **\%** and **\!** are replaced by **%** and **!**, respectively.
i:n	Skips to the **i**-th next file given in the command line (skips to the last file as **n** does not make sense).
i:p	Skips to the **i**-th previous file given in the command line. If this command is given in the middle of printing out a file, **more** goes back to the beginning of the file. If **i** doesn't make sense, **more** skips back to the first file. If **more** is not reading from a file, the bell rings and nothing else happens.
:f	Displays the current file name and line number.
:q or **:Q**	Exits from **more** (same as q or Q).
.	(dot) Repeats the previous command.

The commands take effect immediately. For example, it is not necessary to type a carriage return. Up to the time when the command character itself is given, the user may enter the line kill character to cancel the numerical argument being formed. In addition, the user may enter the erase character to redisplay the **--More--(xx%)** message.

At any time when output is being sent to the terminal, the user can press the quit key (normally **Ctrl-**). The **more** command stops sending output and displays the usual **--More--** prompt. The user may then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, because any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

The terminal is set to **noecho** mode by this program so that the output can be continuous. What you type, therefore, does not show on the terminal, except for the **/** and **!** commands.

If the standard output is not a teletype, **more** acts just like **cat**, except that a header is printed before each file (if there is more than one).

Example

A sample usage of **more** in previewing **nroff** output is:

```
  nroff -ms -s+2 doc.n|more
```

Files

/usr/lib/terminfo/* Terminal information files
/usr/lib/more.help Help file.

Related Information
See the following commands: "csh" in topic 1.1.100, "mail, Mail" in topic 1.1.253, and "sh, Rsh" in topic 1.1.420.

1.1.278 mount

Purpose
Makes a file system available for use.

Syntax

```
         +--------+
mount ---¦ +----+ +---¦
         +-¦ -L +-+
           ¦ -v ¦
          ¦+----+¦
          +------+

                        +---------------+    +------- directory --------+
                      +-¦ +-----------+ +---¦                          +-+
                      ¦ +-¦ -r        +-+    +--- device --- directory -+ ¦
         +--------+   ¦   ¦ -         ¦¦       ¦          +----+          ¦
mount ---¦ +----+ +---¦   ¦¦ -o options ¦¦     +- host:NFSdir -+          +--
         +-¦ -v +-+   ¦   ¦+------------+¦                                ¦
           ¦ -f ¦     ¦    +-------------+                                ¦
          ¦+----+¦    ¦ +---------+                                       ¦
          +------+    ¦ ¦ -a      ¦                                       ¦
                      +-¦ all     +---------------------------------------+
                        ¦ -t type ¦
                        +---------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **mount** command instructs the operating system to make a file system available for use. In addition, you can use **mount** to build other file trees made up of directory and file mounts. The **mount** command mounts the specified **device** on the specified **directory** and records its availability in **/etc/mtab**. After **mount** has finished, **directory** becomes the root of the newly mounted file system. You can also use **host:NFSdir** to mount a **NFSdir** from the indicated host using the network fileserver.

Members of the system group can do any mount described in the **/etc/filesystems** file (**mount directory**). Users operating as superuser can issue any **mount** command.

If you enter the **mount** command without arguments, it writes to standard output all the mounted file systems (except those mounted with the **-s** flag), and whether they are read-only.

The environment variables **LANG** and **LC_TIME** control the appearance of the modification date and time when the **-v** option is specified with the first form of the command. If the Transparent Computing Facility is installed, the use of the **mount** command lists the file systems mounted on any site in the cluster. Use the **-L** flag to see only the file systems mounted on the local site.

If you specify only a **directory** name, **mount** takes it to be the name of the directory on which a file system is usually mounted (as defined in the **/etc/filesystems** file). **mount** looks up the associated device and mounts it. This is the most convenient way of using the **mount** command, as it does not require you to remember what is normally mounted on a directory.

The **/etc/filesystems** file should include a **stanza** for each mountable file system. This stanza should specify at least the name of the file system and the device on which it resides. If the stanza includes a **mount attribute**, the **mount** command uses the associated values. It recognizes three values in the mount attribute: **true**, **false**, and **read/only** (see the **filesystems** file in *AIX Operating System Technical Reference* for a description of these mount attributes). The command **mount all** causes all file systems with the attribute **mount=true** to be mounted in their normal places. This command is typically used during system initialization.

If you are operating with superuser authority, you can mount a file system arbitrarily by naming both a **device** and a **directory** on the command line. **mount** takes **device** to be the name of the block device special file and **directory** to be the directory on which it should mount the file system.

The **mount** and **umount** programs maintain the mount table in **/etc/mtab** as accurately as possible. However, some events can invalidate this mount table. (The system itself has an internal mount table that it maintains independently.) Several programs and library routines use **/etc/mtab** to determine the fully qualified name of the current directory. If **/etc/mtab** does not properly reflect the state of all mounted file systems, these programs may stop working.

Using the **-s** flag, deleting **/etc/mtab**, or truncating it after file systems have been mounted will almost surely invalidate its contents.

If TCF is installed, only devices on the local site may be mounted. Once mounted, a file system appears to be mounted to all processes on each site within the cluster. Similarly, a file system mounted from a network file server (NFS), appears to be mounted through the cluster.

If a file system has been designated as a replicated file system, multiple copies of the file system may be mounted at the same time. Each mounted copy must be mounted from a different site within the cluster and must be mounted over the same directory. As long as one copy of a file system is mounted, files in that file system may be accessed. If the file system copy which has been designated as the primary is one of the mounted file system copies, files within the file system may be modified.

Flags

-r Mounts a file system as a read-only file system, regardless of the specification in **/etc/filesystems**. This is similar to:

```
mount -o ro directory
```

or

```
mount -o ro device directory
```

-s Does not record the availability of the new file system in **/etc/mtab**. This allows a file systems to be mounted on a read-only root where **/etc/mtab** would not be writable. As several programs and library routines depend on **/etc/mtab**, use this flag with caution.

-t type Mounts all stanzas in **/etc/filesystems** that contain **type = type** and are not mounted. (**type** is a string value, such as **remote.**)

-a	Mounts all file systems in **/etc/filesystems**. Has the same effect as **mount all**.
-f	Add all entries to **<LOCAL>/mtab** and do no actual mounts.
-L	Displays information for the local site only.
-v	Verbose; **mount** displays a message indicating the file system being mounted. May be used when listing mount table to include the date the file system was mounted. In addition, the **-v** flag forces large fields to be displayed in their entirety. Without the **-v** flag, fields that would not otherwise fit in the columns of the tabular output are shortened to fit.
-o options	Used to specify options, a list of comma-separated words from the list below. Some options are valid for all file system types, while others apply to a specific type only.

Options valid on all file systems (the default is **rw**):

rw	Read-write
ro	Read-only

Options specific to AIX file systems (the default is **noquota**):

quota	Usage limits enforced
noquota	Usage limits not enforced

The **options** specific to **nfs** (NFS) file systems (the defaults are:

fg,retry=1,timeo=7,retrans=3,port=NFS_PORT,hard

with defaults for **rsize** and **wsize** set by the kernel):

bg	if the first mount attempt fails, retry in the background.
fg	retry in foreground.
retry=n	set number times to retry mount to **n**.
rsize=n	set read buffer size to **n** bytes.
wsize=n	set write buffer size to **n bytes**.
timeo=n	set NFS timeout to **n** tenths of a second.
retrans=n	set number of NFS retransmissions to **n**.
port=n	set server IP port number to **n**.
soft	return error if server doesn't respond.
hard	retry request until server responds.
intr	allow keyboard interrupts on hard mounts.

The **bg** option causes **mount** to run in the background if the server's **mountd**(8) does not respond. **mount** attempts each request **retry=n** times before giving up. Once the filesystem is mounted, each NFS request made in the kernel waits **timeo=n** tenths of a second for a response. If no response arrives, the time-out is multiplied by 2 and the request is retransmitted. When **retrans=n** retransmissions have been sent with no reply a **soft** mounted filesystem returns an error on the request and a **hard** mounted filesystem prints a message and retries the request. Filesystems that are mounted read-write should use the **hard** option. The **intr** option allows keyboard interrupts to kill a process that is hung waiting for a response on a hard mounted filesystem. The number of bytes in a read or write request can be set with the **rsize** and **wsize** options.

Examples

1. To list the file systems that are mounted:

```
mount

sitename  mounted         mounted over    options  Gfs#  pck
tuna      /dev/chd0035    /               rw          1  1 PRI
tuna      /dev/chd0034    /tuna           rw          3  1
tuna      /dev/chd00033   /ul             rw          2  1
tuna      snap:/source    /source         intr,bg   100  1 NFS
cod       /dev/fhd00035   /               rw          1  3 BAC
cod       /dev/fhd00034   /cod            rw          4  1
pogy      /dev/hd3        /               rw          1  4 SEC
pogy      /dev/hd2        /pogy           rw          7  1
pogy      /dev/hd1        /pogy/tmp       rw          6  1
```

   For each file system, **mount** lists the site name, device name, directory which is mounted over, options, global file system number, and file system pack number. For replicated file systems, the last column indicates PRI (primary copy), BAC (backbone copy), or SEC (secondary copy). The last column also indicates NFS file systems.

2. To list the file systems that are mounted including the date mounted:

```
mount -v

sitename  mounted         mounted over   options    date          Gfs#  pck
tuna      /dev/chd0035    /              rw         Jan 20 16:53     1  1 PI
tuna      /dev/chd0034    /tuna          rw         Jan 20 16:53     3  1
tuna      /dev/chd00033   /ul            rw         Jan 20 16:55     2  1
tuna      snap:/source    /source        intr,bg    Jan 20 16:58   100  1 NI
cod       /dev/fhd00035   /              rw         Jan 20 19:27     1  3 BA
cod       /dev/fhd00034   /cod           rw         Jan 20 19:27     4  1
pogy      /dev/hd3        /              rw         Jan 22 11:15     1  4 SI
pogy      /dev/hd2        /pogy          rw         Jan 22 11:15     7  1
pogy      /dev/hd1        /pogy/tmp      rw         Jan 22 11:15     6  1
```

3. To list the file systems mounted on the local site:

```
mount -L

mounted         mounted over   options  Gfs#  pck
/dev/chd0035    /              rw          1  1 PRI
/dev/chd0034    /tuna          rw          3  1
/dev/chd00033   /ul            rw          2  1
snap:/source    /source        intr,bg   100  1 NFS
```

4. To list the file systems mounted on the local site including date mounted:

```
mount -Lv

mounted         mounted over  options    date          Gfs#  pck
/dev/chd0035    /             rw         Jan 20 16:53     1  1 PRI
/dev/chd0034    /tuna         rw         Jan 20 16:53     3  1
```

```
    /dev/chd00033  /ul          rw        Jan 20 16:55    2  1
    snap:/source   /source      intr,bg   Jan 20 16:58  100  1 NFS
```

5. To mount a diskette:

   ```
   mount  /dev/fd0  /diskette0
   ```

   This mounts a diskette (**/dev/fd0**) onto the directory **/diskette0**. A file system must already exist on the diskette, and the directory **/diskette0** must already exist. To access a file on the diskette, use a path name that begins with **/diskette0**. For example, to access **prog.c** use **/diskette0/prog.c**.

   Warning: Be sure that the diskette is unmounted before you remove it from the drive, or you may lose some of your data.

6. To mount a write-protected diskette:

   ```
   mount  -r  /dev/fd0  /diskette0
   ```

   This mounts the diskette on **/diskette0** as a read-only, file system. This tells the operating system not to update file access times, which would cause errors with a write-protected diskette.

7. To mount a default file system:

   ```
   mount  /diskette0
   ```

   This mounts the device that is usually mounted on **/diskette0**, which is determined by information in the file **/etc/filesystems**.

8. To mount all default file systems:

   ```
   mount  all
   ```

   This mounts all standard file systems in **/etc/filesystems** marked **mount = true**.

9. To mount a file or directory from the **/etc/filesystems** file with a specific type:

   ```
   mount -t temp
   ```

   This mounts all files or directories in the **/etc/filesystems** file that have a stanza that contains the attribute **type = temp**.

Files

/etc/filesystems	Descriptions of mountable file systems.
/etc/mtab	Record of mounted file systems. (Symbolic link to **<LOCAL>/mtab**.)

Related Information

See the following command: "umount, unmount" in topic 1.1.491.

See the **mount** and **umount** system calls and the **filesystems** and **mtab** files in *AIX Operating System Technical Reference*.

See the "Introduction to International Character Support" in *Managing the*

AIX Operating System.

For NFS **mount** details, see "Managing the IBM AIX Network File System" in *Managing the AIX Operating System*.

1.1.279 msgchk

Purpose
Checks for messages.

Syntax

```
          +------------+
msgchk ---¦            +---¦
          +--- user ---+
                     ¦
            +--------+

msgchk --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **msgchk** command is used to check mail drops for messages waiting to be received. This command **msgchk** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **msgchk** command checks all mail drops belonging to the specified user IDs and reports which mail drops contain messages that have not been received. **msgchk** also indicates whether it appears you have already seen these messages. If you do not specify a **user** argument, **msgchk** checks the current user's mail drops.

Flag

-help	Displays help information for the command.

Files

$HOME/.mh_profile	The MH user profile.
/usr/lib/mh/mtstailor	The MH tailor file.
$HOME/$USER	The location of the mail drop.

Related Information

See the MH command "inc" in topic 1.1.206.

See the **mh-mail** and and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.280 msgs

Purpose
Reads system messages.

Syntax

```
        +----------------+
msgs ---¦ +------------+ +---¦
        +-¦ -f         +-+
          ¦ -h         ¦¦
         ¦¦ -l         ¦¦
         ¦¦ -p         ¦¦
         ¦¦ -q         ¦¦
         ¦¦ num        ¦¦
         ¦¦ -number    ¦¦
         ¦+------------+¦
         +--------------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **msgs** command is used to read system messages. These messages are sent by mailing to the login **msgs** and should be short pieces of information which are suitable to be read once by most users of the system.

The **msgs** command is normally invoked each time you log in, by placing the command in the file **.profile** if you use **/bin/sh** (**.login** if you use **/bin/csh**). It will then prompt you with the source and subject of each new message. If there is no subject line, the first few non-blank lines of the message will be displayed. If there is more to the message, you will be told how long it is and asked whether you wish to see the rest of the message. The possible responses are:

y	Type the rest of the message.
RETURN	Synonym for y.
n	Skip this message and go on to the next message.
-	Redisplay the last message.
q	Drops you out of **msgs**; the next time you run the program it will pick up where you left off.
s	Append the current message to the file **Messages** in the current directory; **s-** will save the previously displayed message. An **s** or **s-** followed by a space and a file name specifies the file to receive the message (instead of the default "Messages").
m	Or **m-** causes a copy of the specified message to be placed in a temporary mailbox and **mail** to be invoked on that mailbox. Both **m** and **s** accept a numeric argument in place of the '-'.

The **msgs** command keeps track of the next message you will see by a number in the file **.msgsrc** in your home directory. In the directory **/usr/msgs** it keeps a set of files whose names are the (sequential) numbers of the messages they represent. The file **/usr/msgs/bounds** shows the low and high number of the messages in the directory so that **msgs** can quickly determine if there are no messages for you. If the contents of **bounds** is incorrect it can be fixed by removing it; **msgs** will make a new **bounds** file the next time it is run.

The **-s** option is used for setting up the posting of messages. The line

```
  msgs: "¦ /usr/ucb/msgs -s"
```

should be included in **/usr/adm/sendmail/aliases** by the system administrator and the command **/usr/lib/newaliases** should be run.

The **-c** option is used for performing cleanup on **/usr/msgs**. An entry with the **-c** option should be placed in **/usr/spool/cron/crontabs** to run every night. This will remove all messages over 21 days old. A different expiration may be specified on the command line to override the default.

Flags

Options when reading messages include:

-f Causes it not to say "No new messages.." This is useful in your **.login** file since this is often the case here.

-q Queries whether there are messages, printing "There are new messages." if there are. The command "msgs -q" is often used in login scripts.

-h Causes **msgs** to print the first part of messages only.

-l Causes only locally originated messages to be reported.

num Causes **msgs** to start at the specified message **num**, rather than at the next message indicated by your **.msgsrc** file. Thus

```
      msgs -h 1
```

prints the first part of all messages.

-number
Causes **msgs** to start **number** messages back from the one indicated by your **.msgsrc** file, useful for reviews of recent messages.

-p Causes long messages to be piped through **more**.

Within the **msgs** command you can also go to any specific message by typing its number when **msgs** requests input as to what to do.

Files

/usr/msgs/*	Data base.
~/.msgsrc	Number of next message to be presented.

Related Information

See the following commands: "crontab" in topic 1.1.98, "mail, Mail" in topic 1.1.253, and "more, page" in topic 1.1.277.

1.1.281 msh

Purpose
Creates a **msh** shell.

Syntax

```
      +-- prompt (msh) --+   +-- -noscan --+   +-- -notopcur --+   +--------
msh ---¦                  +---¦   one of    +---¦    one of     +---¦
      +- -prompt string -+   ¦ +---------+ ¦   ¦ +-----------+ ¦   +- file --
                             +-¦ -scan   +-+   +-¦ -topcur   +-+
                               ¦ -noscan ¦       ¦ -notopcur ¦
                               +---------+       +-----------+

msh --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **msh** command allows you to run many of the Message Handling (MH) commands on a single packed format file, rather than the usual MH folder containing separate message files. **msh** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **file** parameter must contain MH-style packed format messages. If you don't specify **file**, **msh** will attempt to read **(~)/msgbox**, which also must consist of packed format messages. The MH **packf** command creates packed format files.

The **msh** command performs a modified subset of MH commands on messages stored in packed format. **msh** prompts you to enter one of the following MH commands, and continues to prompt you for commands until you press **END OF FILE** or enter **quit**:

ali	**burst**	**comp**	**dist**
folder	**forw**	**inc**	**mark**
mhmail	**msgchk**	**next**	**pack**
pick	**prev**	**refile**	**repl**
rmm	**scan**	**send**	**show**
sortm	**whatnow**	**whom**	

You can also enter **help** to display a brief overview.

Flags

-help Displays help information for the command.

-noscan Does not scan unseen items.

-notopcur Makes the current message track the center line of the **vmh** scan window when **msh** is invoked from **vmh**. This flag is the default.

-prompt *string* Prompts for **msh** commands with the specified string.

-scan Scans unseen items.

-topcur Makes the current message track the top line of the **vmh** scan window when **msh** is invoked from **vmh**.

Profile Entries

fileproc:	Specifies the program used to refile messages.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your **user_mh_directory**.
showproc:	Specifies the program used to show messages.

Files

$HOME/..mh_profile	The MH user profile.
/usr/lib/mh/mtstailor	The MH tailor file.

Related Information

See other MH commands: "ali" in topic 1.1.17, "burst" in topic 1.1.46, "comp" in topic 1.1.85, "dist" in topic 1.1.131, "forw" in topic 1.1.174, "inc" in topic 1.1.206, "mark" in topic 1.1.259, "mhmail" in topic 1.1.264, "msgchk" in topic 1.1.279, "next" in topic 1.1.293, "packf" in topic 1.1.310, "pick" in topic 1.1.316, "prev" in topic 1.1.323, "refile" in topic 1.1.366, "repl" in topic 1.1.369, "rmm" in topic 1.1.380, "scan" in topic 1.1.409, "send" in topic 1.1.416, "show" in topic 1.1.423, "sortm" in topic 1.1.434, "whatnow" in topic 1.1.533 and "whom" in topic 1.1.539.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.282 mv, move

Purpose
Moves files.

Syntax

```
 one of                    +--- file --- directory ------+
+------+   +--------+   ¦           ¦                   ¦
¦ mv   +---¦ +----+ +---¦ +--------+                   +---¦
¦ move ¦   +-¦ -f +-+   ¦    one of                     ¦
+------+     ¦ -i ¦¦    ¦ +-----------+                 ¦
            ¦¦ -- ¦¦    +-¦ file      +--- newname ---+
            ¦+----+¦      ¦ directory ¦
            +------+      +-----------+
```

Description

Warning: The **mv** command may overwrite existing files unless **-i** is specified to prompt the user.

The **mv** (**move**) command moves files or directories from one directory to another, or it renames a file or directory. If you move a **file** to a new **directory**, it retains the base file name. When you move a file, all links to other files remain intact, except when you move it to a different file system.

When you use **mv** to rename a **file**, then **newname** can specify either a new file name or a new directory path name. If moving the file would overwrite an existing write-protected file and if standard input is a work station, **mv** displays the permission code of the file to be overwritten and reads one line from standard input. If the line begins with **y**, the move takes place and the file is overwritten. If not, **mv** does nothing with the **file**.

If **file** is a symbolic link, the symbolic link is unlinked and a new symbolic link is created. If **newname** is a symbolic link and it points to a directory, **file** is moved to the directory. If **newname** is a symbolic link and it points to a file, the symbolic link is removed and **file** is renamed to **newname**. If **newname** is a symbolic link that points to a non-existent file, and standard input is a work station, **mv** displays the permission code of the symbolic link and reads one line from standard input. If the line begins with **y**, the symbolic link is removed and **file** is renamed to **newname**; otherwise **mv** does nothing.

The **file** may reference a hidden directory, but the @ notation must be used. Moving individual components of hidden directories must be done by identifying them explicitly.

Note: If the **file** is on different file system than **directory**, **mv** must copy the **file** to the new file system and delete the original. In this case, the owner name becomes that of the user, and all links to other files are lost.

Flags

-f Does not prompt before removing a write-protected file. This also overrides the **-i** flag.

-i Whenever an existing file is to be overwritten, the user is prompted by the name of the file followed by a question mark (?). If a line starting with **y** is entered, the move takes place. Any other reply prevents the move from occurring.

-- Interprets all the following arguments to **mv** as file names. This allows file names starting with minus signs.

Examples

1. To rename a file:

   ```
   mv  appendix  apndx.a
   ```

   This renames **appendix** to **apndx.a**. If a file named **apndx.a** already exists, its old contents are replaced with those of **appendix**.

2. To rename a directory:

   ```
   mv  book  manual
   ```

   This renames **book** to **manual**. If a directory named **manual** exists, the directory book becomes a subdirectory of manual.

3. To move a file to another directory and give it a new name:

   ```
   mv  intro  manual/chap1
   ```

   This moves **intro** to **manual/chap1**. The name **intro** is removed from the current directory, and the same file appears as **chap1** in the directory **manual**.

4. To move a file to another directory, keeping the same name:

   ```
   mv  chap3  manual
   ```

   This moves **chap3** to **manual/chap3**.

Note the difference: Examples 1 and 3 name two files, Example 2 names two directories, and Example 4 names a file and a directory.

5. To move several files into another directory:

   ```
   mv  chap4  tom/chap5  /u/manual
   ```

   This moves **chap4** to **/u/manual/chap4** and **tom/chap5** to **/u/manual/chap5**.

6. To use **mv** with pattern-matching characters:

   ```
   mv  manual/*  .
   ```

   This moves all files in the directory **manual** into the current directory (.), giving them the same names they had in **manual**. This also empties **manual**. You must type a space between the star and the period.

7. To use **mv** to rename hidden directories:

   ```
   mv /bin/who@ /bin/OLDwho@
   ```

This renames the whole hidden directory. Programs that need to access its components must use **/bin/OLDwho** as a command.

8. To use **mv** to move components within a hidden directory:

```
mv /bin/who@/v1i386 /bin/who@/v2i386
```

   This moves the **v1i386** of the hidden directory to **v2i386**.

Related Information

See the following commands: "chmod" in topic 1.1.67, "ln" in topic 1.1.234, and "rm, delete" in topic 1.1.375.

See the **rename** system call in *AIX Operating System Technical Reference.*

1.1.283 mvdir

Purpose
Moves a directory.

Syntax

/etc/mvdir --- **directory1 --- directory 2** ---¦

Description
The **mvdir** command renames directories within a file system. To use **mvdir**, you must have write permission to **directory1** and **directory2** and to the parent directories of **directory1** and **directory2**. The **directory1** parameter must name an existing directory. If **directory2** does not exist, **directory1** is moved to **directory2**. If **directory2** exists, **directory1** becomes a subdirectory of **directory2**. Neither directory can be a subset of the other.

Note: **Directory1** and **directory2** may be the names of files. If **directory2** is a file name, it is replaced with **directory1**.

Example

To rename or move a directory to another location:

```
  /etc/mvdir  appendixes  manual
```

If **manual** does not exist, this renames the directory **appendixes** to **manual**. You can also rename a directory with the **mv** command.

If a directory named **manual** already exists, this command moves **appendixes** and its contents to **manual/appendixes**. In other words, **appendixes** becomes a subdirectory of **manual**.

Related Information

See the following commands: "mkdir" in topic 1.1.268 and "mv, move" in topic 1.1.282.

See the **rename** system call in *AIX Operating System Technical Reference*.

1.1.284 m4

Purpose
Preprocesses files, expanding macro definitions.

Syntax

```
      +--------+   +- -B4096 -+   +- -H199 -+   +- -S100 -+   +- -T512 -+
m4 ---¦ +----+ +---¦          +---¦         +---¦         +---¦         +---
      +-¦ -e +-+   +- -Bnum --+   +- -Hnum -+   +- -Snum -+   +- -Tnum -+
        ¦ -s ¦
       ¦+----+¦
       +------+
   +-------------------------+   +----------+   +--------+
 ---¦           +-- ="" ---+ +---¦          +---¦        +---¦
   +- -D name -¦          +-+   +- -Uname -+ ¦ +- file -+
  ¦             +- =value -+                 ¦         ¦
  +------------------------------------------+  +------+
```

Note: This command does not have MBCS support.

Description
The **m4** command is a macro processor used as a preprocessor for C and other languages. You can use it to process built-in macros or user-defined macros. Each **file** is processed in order. If you do not specify a **file** or if you give a minus (**-**) as a file name, **m4** reads standard input. It writes the processed macros to standard output. Macro calls follow the form:

```
  macroname(argument...)
```

The left parenthesis must immediately follow **macroname**. If the left parenthesis does not follow the name of a defined macro, **m4** reads it as a macro call with no arguments. Macro names consist of ASCII alphabetic letters, digits, and the underscore character (_). Extended characters are not allowed in macro names. The first character cannot be a digit.

While collecting arguments, **m4** ignores unquoted leading blanks, tabs, and new-line characters. Use single quotation marks to quote strings. The value of a quoted string is the string with the quotation marks stripped off.

When **m4** recognizes a macro, it collects arguments by searching for a matching right parenthesis. If you supply fewer arguments than appear in the macro definition, **m4** considers the trailing arguments in the definition to be null. Macro evaluation proceeds normally during the collection of the arguments. All commas or right parentheses within the value of a nested call are translated literally; they do not need an escape character or quotation marks. After collecting arguments, **m4** pushes the value of the macro back onto the input stream and scans again.

Subtopics

1.1.284.1 Built-in Macros

The **m4** command makes available the following built-in macros. You may redefine them, but you will lose the original meaning. The values of these macros are null unless otherwise stated:

define(name,new_name)	Replaces the macro **name** with the value of **new_name**. The **new_name** string can take the form **$n**... (where **n** is a digit). In this case, each occurrence of **n** in the replacement text is replaced by the **n**-th argument of **name**. **$0** is the name of the macro. The null string replaces missing arguments. The number of arguments replaces **$#**. A comma-separated list of all arguments replaces **$***. **$@** acts like **$***, but each argument is quoted with the current quotation character (see **changequote**).
undefine(name)	Removes the definition of **name**.
defn(name...)	Returns the quoted definition of **name**.
pushdef(name,new_name)	Redefines **name** with **new_name**, as in **define**, but saves any previous definition.
popdef(name...)	Removes the current definition of **name** and returns to the previous definition, if one existed.
ifdef(name,true,[false])	Returns the value of **true** only if **name** is defined; otherwise returns **false**. If you do not supply **false**, its value is null. **Note:** The word **unix** is predefined.
shift(argument...)	Returns all but the first argument. The other arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.
changequote(L,R)	Changes quote symbols to **L** and **R**. The symbols can be up to five bytes long. **changequote** without arguments restores the original values (' ').
changecom(Lcom,Rcom)	Changes left and right comment markers from the default # and new-line character to **Lcom** and **Rcom**. With no arguments, the comment mechanism is disabled. With one argument, the left marker becomes the parameter and the right marker becomes a new-line character. With two arguments, both markers are affected. Comment markers can be up to five bytes long.
divert(num)	Changes the current output stream to stream **num**. There are 10 output streams, numbered 0-9. The final output is the concatenation of the streams in numerical order. Initially, stream 0 is the current stream. **m4** discards output diverted to

	a stream other than 0-9.
undivert(num...)	Causes immediate output of text from the specified diversions (or all diversions if there is no argument). Text may be undiverted into another diversion. Undiverting discards the diverted text.
divnum	Returns the value of the current output stream.
dnl	Reads and discards characters up to and including the next new-line character.
ifelse([string1,string2,true,[false]]...)	If **string1** and **string2** are the same, the value is **true**. If they are not and if there are more than four arguments, **m4** repeats the process with the additional arguments (4, 5, 6, and 7). Otherwise, the value is either **false** or null if you provide no value for **false**.
incr(num)	Returns the value of its argument incremented by 1.
decr(num)	Returns the value of its argument decreased by 1.
eval(expr[,num1[,num2]])	Evaluates its first argument as an arithmetic expression, using 32-bit arithmetic. The operators you can use include +, -, *****, /,%, ^ (exponentiation), bitwise **&**, |, ~, and ^ relationals, and parentheses. Octal and hex numbers can be specified as in C. **num1** specifies the radix for the result of the expression. The default radix is 10. The optional **num2** specifies the minimum number of digits in the result.
len(string)	Returns the number of bytes in **string**.
dlen(string)	Returns the number of displayable characters in **string**; that is, two-byte extended characters are counted as one displayable character.
index(s1,s2)	Returns the position in the string **s1** where the string **s2** begins (zero origin), or -1 if the second parameter does not occur.
substr(string,position,num)	Returns a substring of **string**. The beginning of the substring is selected with **position**, and **num** indicates the length of the substring. Without **num**, the substring includes everything to the end of the first string.
translit(string,from,to)	Transliterates the characters in **string** from the set given by **from** to the set given by **to**. No abbreviations are permitted. Two-byte extended characters are correctly mapped into the corresponding replacement characters.

include(file)	Returns the contents of **file** or displays an error message if it cannot access the file.
sinclude(file)	Returns the contents of **file**, but gives no error message if **file** is inaccessible.
syscmd(command)	Runs the AIX **command**. No value is returned.
sysval	Returns the return code from the last call to **syscmd**.
maketemp(...XXXXX...)	Replaces **XXXXX** in its argument with the current process ID number.
m4exit(value)	Exits from **m4** immediately, returning the specified exit **value** (the default is 0).
m4wrap(lastmacro)	Runs **lastmacro** after reading the end-of-file character. For example: **m4wrap(`cleanup()')** runs the **cleanup** macro at the end of **m4**.
errprint(message)	Includes **message** on the diagnostic output file.
dumpdef([name...])	Writes to standard output the current names and definitions for the named items or for all if no arguments are provided.
traceon(macro)	Turns on tracing for **macro**. If none is named, tracing is turned on for all macros.
traceoff(macro...)	Turns off trace globally and for any **macro** specified. Macros specifically traced by **traceon** can be untraced only by specific calls to **traceoff**.

Flags

-Bnum	Makes **num** the size of the push-back and parameter collection buffers. If the **-Bnum** flag is not used, the size is 4096.
-e	Operates interactively. Interrupts are ignored and the output is not buffered.
-Hnum	Makes **num** the size of the symbol table hash array. The size must be a prime number. If the **-Hnum** flag is not used, the size is 199.
-s	Enables the line sync output for the C preprocessor (#line...).
-Snum	Makes **num** the size of the call stack. Macros take three slots, and nonmacro arguments take one. If the **-Snum** flag is not used, the size is 800 slots.
-Tnum	Makes **num** the size of the token buffer. If the **-Tnum** flag is not used, the size is 512 bytes.

The preceding flags must appear before any file names and before any **-D** or

-U flags.

-Dname[=val]	Define **name** as **val**. If **val** is not specified, **name** becomes null.
-Uname	Undefines a **name** previously defined with the **-D** flag.

Example

To preprocess a C language program with **m4** and compile it:

```
  m4  prog.m4  >prog.c
  cc  prog.c
```

Related Information

See the following commands: "cc" in topic 1.1.52 and "cpp" in topic 1.1.94.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.285 nccheck

Purpose
Checks cluster **/etc/netparams** configuration.

Syntax

```
         +------------------+  +------+
nccheck --¦      one of      +--¦      +-¦
         +------- -L --------¦  +- -A -+
         ¦                   ¦
         +-- -s --- site # --+
                            ¦
                 +---------+
```

Description

The **nccheck** command is used to rapidly confirm that the cluster **/etc/netparams** file matches the standard values (as shipped). Differences are reported on a per-item basis. The **nccheck** commmand also compares each running kernel in the cluster to the **/etc/netparams** file to confirm the use of **/etc/params**. The **nccheck** provides the user with the ability to check network configuration parameters against the established norms. Furthermore, **nccheck** can be used to report on various system parameters which may affect network configuration.

The **nccheck** command runs on any site in a cluster, executing sub-programs on each desired site in the cluster to gather data. Once the data has been gathered, a report is prepared according to the user options and is written to standard output. Network parameter check reports are formatted as follows:

```
  site_number timeout retries window sp_wind checksum
      ... (for each site) ...
```

Parameters which differ from the norm are highlighted by displaying the norm in parenthesis following the observed value.

Sites appearing in **/etc/sites** from which data was not able to be collected are flagged as in the following report:

```
  Data was not collected from the following sites:
  site_number reason_data_was_not_collected
      ... (for each site) ...
```

Extended reports contain parameters and data related to the network configuration of a site. The extended reports are formatted as in the following example:

```
  ---------------------------------------------------------------
   Site               Memory   Buffers      Swap
   Number                                   Space
  ---------------------------------------------------------------
   fafnir   (1)        12160       482     40000
   werth    (2)         8068       284     40000
   atlas    (3)         6016       200     24000
   zaniah   (4)         6016       200     40000
```

The memory and swap space sizes are reported in 1024-byte granules.

AIX sites running on System/370 virtual machines have the following data reported:

```
  ---------------------------------------------------------------
   Site            370paging  370hdclk
   Number               Rate
  ---------------------------------------------------------------
   fafnir   (1)            0         0
   werth    (2)            4         0
   snap     (5)            4         0
   carmen   (14)           0         0
```

The System/370 paging rate is the paging rate returned by a **cpcmd indicate** command. The System/370 hard clock value is a count of the number of times AIX has determined that VM is providing an insufficient amount of CPU to AIX.

Network status is reported as in the following example:

```
  ---------------------------------------------------------------
   Site               # mbuf     # bad     uptime
   Number             denied    routes    of site
  ---------------------------------------------------------------
   fafnir   (1)            0         0   0w  1d  2h 44m 20s
   werth    (2)            0         0   0w  1d 19h 22m 16s
   atlas    (3)           11         0   0w  2d 23h  9m 45s
   zaniah   (4)            0         6   1w  2d 14h 24m 29s
   snap     (5)            0      6225   0w  0d  9h 43m 37s
```

The **# mbuf denied** and **# bad routes** values are the error counters also returned by the **netstat** command.

The status of all interfaces on selected sites is reported in the same format as **ifconfig** command output, as in the following example:

```
  The Net Interface Status and Address for site fafnir   (1):
  --------------------------------------------
    il0: flags=22<BROADCAST,NOTRAILERS> metric 0
       inet 0.0.0.0 netmask 0 broadcast 0.0.0.0
    ceti0: flags=463<UP,BROADCAST,NOTRAILERS,RUNNING,ETHERNET> metric 4
       inet 130.200.6.1 netmask ffff0000 broadcast 130.200.255.255
    vctc0: flags=10<POINTOPOINT> metric 0
       inet 0.0.0.0 netmask 0 broadcast 0.0.0.0
```

Flags

-A	Prepares an extended report.
-L	Gathers data from the local site only.
-s site #	Gathers data from the specified sites. The default action is to prepare network parameter reports for all sites in the cluster.

Files

/usr/bin/ncbe	Data collector program.

Related Information

See the following commands: "netparams" in topic 1.1.287 and "lnetstat" in topic 1.1.235.

In the *AIX TCP/IP User's Guide*, see the following commands: **ifconfig** and **netstat**.

1.1.286 ncheck

Purpose
Generates path names from i-numbers.

Syntax

```
          +---------------------------------------+   +--------------+
ncheck ---¦ +--- -s --------------------------+ +---¦              +---¦
          +-¦ +------+   +-------------------+ +-+   +- filesystem -+
            +-¦      +---¦                   +-+
              +- -a -+   +- -i --- inumber ---+
                                             ¦
                               +-----------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **ncheck** command without any flags writes to standard output the path name and i-number list for all files in **filesystem**. If you do not specify **filesystem**, **ncheck** operates on the file systems described in **/etc/filesystems**.

If you specify an invalid file system, the **??** in the name stands for the parent of a file that does not have a parent. Path names beginning with ... (dot dot dot) indicate a loop.

Hidden directories and their components are indicated in the output by the presence of @ after the name of the hidden directory, before the **/** that separates the directory name from the component name.

Filesystem may name one of the following:

block or character special file indicating a local device.

block special file indicating a remote device on the same type of machine (AIX PS/2 or AIX/370).

the mount point on which a local file system is normally mounted (as defined in **/etc/filesystems**).

If **filesystem** names a copy of a replicated file system, only files actually stored in this copy of the file system are listed. In particular, if a directory entry for a file is present, but the file is not--the file will be ignored by **ncheck**.

Flags

-a	Lists includes the file names **.** (dot) and .. (dot dot).
-i inumber...	Lists only the files specified by **inumber**.
-s	Lists only special files and files with set-user-ID mode.

Examples

1. To list the i-number and path name of each file in the default file systems:

```
  ncheck
```

2. To list all the files in a specified file system:

   ```
     ncheck  -a  /
   ```

   This lists the i-number and path name of each file in the root file system (/), including the . (dot) and .. (dot-dot) entries in each directory (**-a**).

3. To list the name of a file when you know its i-number:

   ```
     ncheck  -i  690  357  280  /diskette0
   ```

   This lists the i-number and path name for every file in the file system **/diskette0** with i-numbers of **690**, **357**, or **280**. If a file has more than one link, all of its path names are listed.

4. To list special and set-user-ID files:

   ```
     ncheck  -s  /
   ```

   This lists the i-number and path name for every file in the root file system that is a special file (also called a **device file**) or that has set-user-ID mode enabled.

Files

/etc/filesystem

Related Information

See the following commands: "fsck, dfsck" in topic 1.1.177 and "sort" in topic 1.1.432.

See **filesystems** in the *AIX Technical Reference.*

1.1.287 netparams

Purpose
Sets cluster networking parameters.

Syntax

/usr/bin/netparams ---¦

Description

The **netparams** commands reads the file **/etc/netparams** and sets the cluster networking parameters to match those indicated in the file. The **netparams** command is generally run upon start-up, but may be re-run at any time.

Diagnostics

The **netparams** command generally runs silently, but may issue complaints about malformed lines if **/etc/netparams** is corrupted. If any of the values in **/etc/netparams** are out of range, a message will be issued indicating which value was in error. Permission denied may also be generated if the user is not the superuser.

Files
/etc/netparams

Related Information

See **netctrl** in the *AIX Operating System Technical Reference.*

1.1.288 newform

Purpose
Changes the format of a text file.

Syntax

```
           +------------------------------------------+
newform ---¦               one of                     +---
           ¦ +------+   +-----------+   +-----------+ ¦
         ¦ +-¦      +---¦ -a  -anum +---¦           +-+
         ¦   +- -s -+   ¦ -p  -pnum ¦   +- -c char -+
         ¦              +-----------+
         +-----(A)

    +---------------+   +-------------------------+
 ---¦    one of     +---¦ one of                  +---
    ¦ +-----------+ ¦   ¦ +----+   +--- -8 ----+ ¦
    +-¦ -b  -bnum +-+   +-¦ -i +---¦¦           +-+
      ¦ -e  -enum ¦       ¦ -o ¦   +- tabspec -+¦
      +-----------+      ¦+----+                ¦
                         +-----------------------+

    +----- -180 ------+   +------+   +------------+
 ---¦       +- 72 --+ +---¦      +---¦            +---¦
    +- -l --¦¦      +-+   +- -f -+   +--- file ---+ ¦
            +- num -+                           ¦   ¦
                                      +---------+   ¦
                                                    ¦
                                 (A) ---------------
```

```
-----------------
¦ Do not put a blank between these items.
```

Note: This command does not have MBCS support.

Description

The **newform** command takes lines from **file** (standard input by default) and writes the formatted lines to standard output. Lines are reformatted in accordance with command line flags in effect.

Except for **-s**, command line flags can appear in any order, can be repeated, and can be mixed with the **file** parameter. Command line flags are processed in the order specified. In other words, flag sequences like **-e15 -l60** yield results different from **-l60 -e15**. Flags are applied to all **files** on the command line.

An exit value of 0 indicates normal execution; a 1 indicates an error.

Notes:

1. The **newform** command normally keeps track of only physical characters; however, for the **-i** and **-o** flags, **newform** keeps track of backspaces in order to line up tabs in the appropriate logical columns.

2. The **newform** command does not prompt you if a **tabspec** is to be read from the standard input (by use of **-i--** or **-o --**).

3. If the **-f** flag is used and the last **-o** flag specified was **-o--** and was preceded by either a **-o--** or a **-i--**, the tab specification format line will be incorrect.

Flags

-a[num]	Adds **num** characters to the end of the line when the line length is less than the effective line length (see the **-c** and **-p** flags in this section). The default line length is 80 characters. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the 80-character limit may be reduced by as much as 50%, depending on the character code set being used.
-b[num]	Truncates **num** characters from the beginning of the line when the line length is greater than the effective line length (see **-lnum**). The default action truncates the number of characters necessary to obtain the effective line length. If you specify **-b** with no **num**, the default takes effect. This flag can be used to delete the sequence numbers from a COBOL program as follows: `newform  -l1 -b7  file-name` The **-l1** must be used to set the effective line length shorter than any existing line in the file so that the **-b** flag is activated.
-c[char]	Changes the prefix/add character to **char**. Default character for **char** is a space.
-e[num]	Same as **-bnum** except that characters are truncated from the end of the line.
-f	Writes the tab specification format line to standard output before any other lines are written. The tab specification format line displayed corresponds to the format specified in the **last -o** flag. If no **-o** flag is specified, the line displayed contains the default specification of **-8**.
-i[tabspec]	Replaces all tabs in the input with the number of spaces specified by **tabspec**. **tabspec** recognizes all tab specification forms described in "tabs" in topic 1.1.459. If you specify a **--** (minus minus) for the value of **tabspec**, **newform** assumes that the tab specification can be found in the first line read from standard input (see **fspec** in *AIX Operating System Technical Reference*). The default **tabspec** is **-8**. A **tabspec** of **-0** expects no tabs; if any are found, they are treated as **-1**.
-l[num]	Sets the effective line length to **num** characters. If **num** is not entered, **-l** defaults to 72. The default line length without the **-l** flag is 80 characters. Tabs and backspaces are considered to be one character (use **-i** to expand tabs to spaces).
-o[tabspec]	Replaces spaces in the input with a tab in the output, according to the tab specifications given. The default **tabspec** is -8. A **tabspec** of **-0** means that no spaces are

converted to tabs on output.

-p[num] Prefixes **num** characters (see **-cchar**) to the beginning of a line when the line length is less than the effective line length. The default action is to prefix the number of characters that are necessary to obtain the effective line length.

-s Removes leading characters on each line up to the first tab and places up to eight of the removed characters at the end of the line. If more than eight characters (not counting the first tab) are removed, the eighth character is replaced by an ***** (asterisk) and any characters to the right of it are discarded. The first tab is always discarded.

The removed characters are saved internally until all other flags specified are applied to that line. The characters are then added at the end of the processed line.

For example, to convert a file with leading digits, one or more tabs, and text on each line, to a file beginning with the text, all tabs after the first expanded to spaces, padded with spaces out to column 72 (or truncated to column 72), and the leading digits placed starting at column 73, the command would be as follows:

```
newform  -s  -i  -l  -a  -e  file-name
```

The **newform** command displays an error message and stops if this flag is used on a file without a tab on each line.

Related Information

See the following commands: "tabs" in topic 1.1.459 and "csplit" in topic 1.1.101.

See the **fspec** file in *AIX Operating System Technical Reference*.

1.1.289 newgfs

Purpose

Finds an available GFS number and reserves its place and number in **/etc/fsmap**.

Syntax

```
          +------+   +------------+   +----------+   +---------+
newgfs ---¦      +---¦    one of +---¦          +---¦         +---
          +- -v -+   ¦      +---+ ¦   +- -d dev -+   +- -f fs -+
                     +- -r -¦ p +-+
                            ¦ r ¦
                            ¦ n ¦
                            +---+

    +-----------+
 ---¦           +--- mountpoint ---¦
    +- -s site -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **newgfs** command provides a unique free Global File System (GFS) number for a file system to be mounted on **mountpoint** and reserves a dummy stanza for that GFS in **/etc/fsmap** file. The dummy stanza should be cleaned out of the **/etc/fsmap** file and replaced by the real stanza out of the **/local/filesystems** file by running **syncfsmap**. Unless a file with the dummy file system stanza for mountpoint is supplied by specifying **-f** flag, **newgfs** assumes that the file system info exists in **/local/filesystems** (or another site's file systems file if **-s** flag is supplied). In general, command line flags over-ride the information provided in the dummy stanza.

Flags

The options are (in order of evaluation):

-v verbose mode. Allows **newgfs** to report inconsistencies and errors in **/etc/fsmap**.

-s site site on which this file system will be residing. If this flag is present, LOCAL is set to that site's **/local** and device and **/local/filesystems** references are assumed to be for that site (this implies that if the site does not yet exist, it nevertheless has to be in **/etc/site** and either **-f** and/or **-r** flags must be used to provide info about the file system).

-f fs file where a dummy stanza of a file system can be found. The following is the necessary information:

```
/xyz:

dev = /dev/hd3
vol = "/xyz"
site = electra
ftype = repl, primary
```

The file **fs** provides default parameters some of which can be overriden from the command line. The **-r** option overrides **ftype**; **-d** overrides **dev**; and **-s** overrides **sites**.

Special care must be taken with the format of the stanza. Mount point must start in column 0 and be terminated by a colon. **vol** should be the same string as the mount point string. Acceptable **ftype** (file system type, for example, replication) values are **repl**, **nonrepl** and **repl,primary**. If **ftype** is **primary**, the above format (quotes and spacing) MUST be adhered to! Invalid **ftype** will be defaulted to **nonrepl**.

Site specification **site** will not over-write **-s** flag's specification.

Ideally, the dummy stanza file should be generated by a program (like **minidisks** or **newsite**) rather than by a user.

r opt Define the file system's replication. Available mutually exclusive options are:

p primary, implies replication
r replicated, implies back-bone
n non-replicated

-d dev Device where the file system is located (this is useful for re-assigning GFS# when a site is being moved into another cluster and the file system already exists); device name should be a complete local path (for example, **/dev/diskxzy**).

This flag will override the info provided in the dummy stanza.

If replication has not yet been set (**-r** flag was not passed) the replication will be looked up in the superblock.

The **newgfs** command will print the reserved GFS# to the standard output. If the file system is replicated and another copy of the file system already exists, its GFS# is returned (gfspack is not set). Special care is taken to lock **/etc/fsmap** file while **newgfs** is processing it. A dummy stanza reserving the GFS#, the site and the mount point is added to the **/etc/fsmap**.

If successful, **newgfs** exits with status 0. On failure, **newgfs** prints out a diagnostic message to standard error and exits with status **-1**.

Diagnostics

Invoked without any parameters, **newgfs** will produce a usage message.

Most of diagnostic messages are self-explanatory. Complaints about file system already existing stem from already having a replicated or or non-replicated file system with the same mount-point for the same site, or having a non-replicated file system with the same mount-point on another site in the **/etc/fsmap**.

Related Information

See the following commands: "syncfsmap" in topic 1.1.454, "mount" in topic 1.1.278, "fsck, dfsck" in topic 1.1.177, and "minidisks" in topic 1.1.266. See in the Technical Reference Vol II, "file systems" and "format".

1.1.290 newgrp

Purpose
Changes your primary group identification.

Syntax

```
          +-----+   +---------+
newgrp ---¦     +---¦         +---¦
          +- - -+   +- group -+
```

Description
The **newgrp** command changes your **primary group** identification to **group**. This command recognizes only group names, not group ID numbers. Group passwords and group names can contain only ASCII characters.

Without an argument, **newgrp** changes your primary group to the one specified in the **/etc/passwd** file. If the group has a password and you do not or if the group has a password and you are not listed in the **/etc/group** file as a member, **newgrp** asks you for the group password.

Note: Your present shell is terminated and replaced with a new shell by the **newgrp** command.

Flag

-	Changes the environment to the login environment of the new group.

Examples

1. To change the primary group ID of the current shell session to **admin**:

   ```
   newgrp  admin
   ```

2. To change the primary group ID back to your original login group:

   ```
   newgrp
   ```

Files

/etc/group	Group file; contains group IDs.
/etc/passwd	Password file; contains user IDs.

Related Information

See the following commands: "login" in topic 1.1.241 and "adduser, users" in topic 1.1.15.

1.1.291 newkernel

Purpose
Configures, builds and installs new AIX kernels.

Syntax

```
                     +-------+  +--------------+  +---------------+
/usr/sys/newkernel --¦       +--¦              +--¦               +--¦
                     +- -d --+  +-- -install --+  +- -systemfile -+
```

Description

The **newkernel** command takes data from the **/etc/master** and the **/etc/system** description files and builds a kernel as specified by those files. The command runs **config** with appropriate arguments to build a configuration summary. It then uses the configuration summary to build a stubout file for features not configured in, a linker alias file to bring in device drivers, and an **ldminit** patchdeck to set the configuration constants to appropriate values. The **newkernel** command uses these files and the files in **/usr/sys/cpu_type** (where **cpu_type** represents the type of machine for which your kernel is built) as input to **ld** to build a kernel.

A copy of the new kernel is installed in **/local/unix.std** if **-install** was specified; the previous version of **/local/unix.std** is saved as **/local/unix.last** and copies of kernel from previous build is saved as **unix.VERS**, where **VERS** is a serial number indicating the number of the build. If **newkernel** is not invoked with the **-install** option, the new kernel is not installed after it is built; the kernel is stored in **/tmp/sysgen.cpu_type**.

The **-systemfile** argument is used to specify a configuration file other than the default configuration (**/etc/system**).

Flags

-d	Echoes each command as it is executed (for use as a debugging tool).
-install	Installs the new kernel into **/local/unix.std** and moves the previous **unix.std** to **unix.last.** Previous kernel builds and patchdecks are saved in **/local/unix.std.VERS** and **/local/patchdeck.VERS**, respectively.

Files

/usr/sys/cf/GENLD.awk	System configuration file.
/usr/sys/cf/stubout.file	System configuration file.
/usr/sys/cf/ifile	System configuration file.
/etc/master	Default master configuration file.
/etc/system	Default system configuration file.

Related Information

See the following commands: "config" in topic 1.1.88 and Appendix A, "AIX Device Table."

See the topic "Generating a New Kernel" in *Managing the AIX Operating System*.

1.1.292 news

Purpose
Writes system news items to standard output.

Syntax

```
        +--------+   +--------+
news ---¦ one of +---¦        +---¦
        ¦ +----+ ¦   +- item -+
        +-¦ -a +-+          ¦
          ¦ -n ¦      +------+
          ¦ -s ¦
          +----+
```

Description
The **news** command keeps you informed of news concerning the system. Each news **item** is contained in a separate file in the directory **/usr/news**. Anyone having read/write permission to this directory can create a news file.

If you run the **news** command without any flags, it displays every current file in **/usr/news**, showing the most recent first. Or you can specify the **item**s you want displayed.

Each file is preceded by an appropriate header. To avoid reporting old news, **news** stores a currency time. The **news** command considers your currency time to be the modification time of the file named **$HOME/.news_time**. Each time you read the news, the modification time of this file changes to that of the reading. Only news item files posted after this time are considered current.

Pressing the INTERRUPT (**Ctrl-C**) key during the display of a news item stops the display of that item and starts the next. Pressing this key again ends the **news** command.

To run **news** each time you log in, include this line in your **$HOME/.profile** file or in the system's **/etc/profile**:

```
  news  -n
```

Flags

-a Displays all news items, regardless of the currency time. The currency time does not change.

-n Reports the names of current news items without displaying their contents. The currency time does not change.

-s Reports the number of current news items without displaying their names or contents. The currency time does not change.

Examples

1. To display the items that have been posted since you last read the news:

```
   news
```

2. To display all the news items:

   ```
   news  -a  |  pg
   ```

   This command displays all the news items a page at a time (**| pg**) whether you have read them yet.

3. To list the names of the news items that you have not read yet:

   ```
   news  -n
   ```

   Each name is a file in the directory **/usr/news**.

4. To display specific news items:

   ```
   news  newusers  services
   ```

   This command displays news about **newusers** and **services**, which are names listed by **news -n**.

5. To display the number of news items that you have not read yet:

   ```
   news  -s
   ```

6. To post news for everyone to read:

   ```
   cp  schedule  /usr/news
   ```

   This command copies the file **schedule** into the system news directory, **/usr/news**, to create the file **/usr/news/schedule**. To do this you must have write permission for **/usr/news**.

Files

/etc/profile	System-wide profile.
/usr/news/*	User news file.
$HOME/.news_time	Contains user currency time.

Related Information

See the following command: "pg" in topic 1.1.315.

See the **profile** file and **environ** special facility in *AIX Operating System Technical Reference*.

1.1.293 next

Purpose
Shows the next message.

Syntax

```
        +------------+   +--- -header ---+   +-------------------------+
next ---¦            +---¦    one of     +---¦         one of          +---¦
        +- + folder -+   ¦ +-----------+ ¦   ¦ +---------------------+ ¦
                         +-¦ -header   +-+   +-¦ -showproc cmdstring +-+
                           ¦ -noheader ¦       ¦ -noshowproc         ¦
                           +-----------+       +---------------------+


next -- help --¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **next** command is used to display the next message in a folder. This command is equivalent to the **show** command with **next** specified as the message. The **next** command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

Note: If you link to **next** and call that link something other than **next**, your link will function like the **show** command, rather than like the **next** command.

The **next** command links to the **show** program and passes this program its flags and attributes. The **show** command invokes a program to perform the listing. The system default is **/bin/pg**. You can define your own default with the **showproc:** entry in **$HOME/.mh_profile**. If you set **showproc:** entry to **mhl**, the **show** command calls an internal **mhl** routine instead of the **mhl** command. You can also specify the program to perform a listing in the **cmdstring** of the **-showproc** flag.

The **show** command passes any flags that it does not recognize to the program performing the listing. Thus, you can specify flags for the listing program, as well as the flags described in this command section.

Flags

+ *folder*	Specifies the folder that contains the message you want to show.
-header	Displays a one-line description of the message being shown. The description includes the folder name and the message number.
-help	Displays help information for the command.
-noheader	Does not display a one-line description of each message being shown.
-noshowproc	Uses **/bin/cat** to perform the listing.

-showproc *cmdstring*	Uses the specified command string to perform the listing.

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.
showproc:	Specifies the program used to show messages.

Files

$HOME/.mh_profile	The MH user profile.

Related Information

See other MH commands: "prev" in topic 1.1.323 and "show" in topic 1.1.423.

See the **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.294 nfsd

Purpose
Starts the daemons that handle client network file system requests.

Syntax

/etc/nfsd --- **nservers** ---¦

Note: This command does not have MBCS support.

Description
The **nfsd** command starts the daemons that handle client file system requests.

The **nservers** parameter specifies the number of file system request daemons to start. Assign the number based on the load expected on a server. Four is a typical value for **nservers**.

Files

/etc/biod

1.1.295 nfsstat

Purpose

Displays Network File System (NFS) statistics.

Syntax

```
           +--------+   +------------+   +------------+
nfsstat ---¦ +----+ +---¦            +---¦            +---¦
           +-¦ -c +-+   +- namelist -+   +- corefile -+
             ¦ -s ¦¦
            ¦¦ -n ¦¦
            ¦¦ -r ¦¦
            ¦¦ -z ¦¦
            ¦+----+¦
            +------+
```

Note: This command does not have MBCS support.

Description

The **nfsstat** command displays statistical information about the NFS and Remote Procedure Call (RPC) interfaces to the kernel. The command also can be used to reinitialize NFS and RPC information. If no options are given, the default is **nfsstat -csnr**.

Flags

-c	Displays client information. Only the client-side NFS and RPC information are printed. This flag can be combined with the **-n** and **-r** flags to print only client NFS or only client RPC information.
-n	Displays NFS information. NFS information for both the client and server side are printed. This flag can be combined with the **-c** and **-s** flags to print only client or only server NFS information.
-r	Displays RPC information. This flag can be combined with the **-c** and **-s** flags to print only client or only server RPC information.
-s	Displays server information. This flag can be combined with the **-n** and **-r** flags to print only server NFS or only server RPC information.
-z	Reinitializes statistics. This flag is for use by the superuser only, and can be combined with any of the other flags to reinitialize particular sets of statistics after printing them.
corefile	Specifies a kernel core dump. The default is **/dev/kmem**.
namelist	Specifies the AIX kernel. The default is **/unix**.

Files

/unix	System namelist.
/dev/kmem	Kernel memory.

1.1.296 nice

Purpose
Runs a command at a different priority.

Syntax

```
        +-- -10 ---+
nice ---¦          ¦+-- cmdstring --¦
        +- number -+


-----------------
¦ Maximum increment is 19.
```

Description
The **nice** command lets you run the specified **command** at a lower priority. The value of **-number** can range from 1 to 19, with 19 being the lowest priority. The default value of **-number** is 10.

If you have superuser authority, you can run commands at a higher priority by specifying **-number** as a negative number, such as **--10**.

Examples

1. To run a command at low priority:

   ```
   nice  cc  -c  *.c
   ```

   This command runs **cc -c *.c** at low priority. This does not run the command in the background. Your work station is not available for doing other things.

2. To run a low priority command in the background:

   ```
   nice  cc  -c  *.c  &
   ```

   This command runs **cc -c *.c** at low priority in the background. Your work station is free so that you can run other commands while **cc** is running. See page 1.1.420.1 for details about starting background processes with **&**.

3. To specify a very low priority:

   ```
   nice  -15  cc  -c  *.c  &
   ```

   This command runs **cc** in the background at a priority that is even lower than the default priority set by **nice**.

4. To specify a very high priority:

   ```
   nice  --10  wall  <<end
   System shutdown in 2 minutes!
   end
   ```

   This command runs **wall** at a higher priority than all user processes. Doing this slows down everything else running on the system. If you do not have superuser authority when you run this command, the **wall** command runs at the normal priority.

The **<<end** and **end** define a "Here Document," which uses the text entered before the **end** line as standard input for the command. For more details, see "Inline Input Documents" in topic 1.1.420.19.

Related Information

See the following commands: "csh" in topic 1.1.100 and "nohup" in topic 1.1.300.

Note: The **csh** command contains a built-in subcommand named **nice**. The command and subcommand do not necessarily work the same way. For information on the subcommand, see the **csh** command.

See the **getpriority** and **setpriority** system calls in *AIX Operating System Technical Reference*.

1.1.297 nl

Purpose
Numbers lines in a file.

Syntax

```
                                                        +-------------+
      +-------------- -bt -hn -fn ---------------+   +---¦ -v1  -w6    +---+
nl ---¦ +----+                    +-- -l2 --+   +---¦   ¦ -nrn -d"\:" ¦   +---
      ¦ ¦ -b ¦   +----------------¦         +-+ ¦   ¦   ¦ -i1         ¦   ¦
      +-¦ -h +---¦    one of      +- -lnum -+ +-+   ¦   +-------------+   ¦
        ¦ -f ¦   ¦ +----------+               ¦¦    ¦ +-----------------+ ¦
       ¦+----+   +-¦ a        +---------------+¦    +-¦ -vnum      -p   +-+
       ¦           ¦ t        ¦                ¦      ¦ -inum      -dxx ¦¦
       ¦           ¦ n        ¦                ¦     ¦¦ -wnum           ¦¦
       ¦           ¦ ppattern ¦                ¦     ¦¦ -nformat        ¦¦
       ¦           +----------+                ¦     ¦+-----------------+¦
       +---------------------------------------+     +-------------------+

    +--- -stab ---+   +--------+
 ---¦ +--- -s --+ +---¦        +---¦
    +-¦         +-+   +- file -+
      +- -ssep -+
```

Note: This command does not have MBCS support.

Description
The **nl** command reads **file**s (standard input, by default), numbers the lines in the input, and writes the numbered lines to standard output. In the output, **nl** numbers the lines on the left according to the flags you specify on the command line.

The input text must be written in logical pages. Each logical page has a header, a body, and a footer section (you can have empty sections). Unless you use the **-p** flag, **nl** resets the line numbers at the start of each logical page. You can set line numbering flags independently for the header, body, and footer sections (for example, no numbering of header and footer lines while numbering text lines only in the body).

You can signal the start of logical page sections with lines in **file** that contain nothing but the following delimiter characters:

Line contents	**Start of**
\:\:\:	Header
\:\:	Body
\:	Footer

You can name only one file on the command line. You can list the flags and the file name in any order.

Flags
All the parameters are set by default. (With the exception that the **nl** command does not set header and footer lines by default.) Use the following flags to change these default settings. Except for the **-s** flag, enter a flag without a parameter to see its default value.

-btype Chooses which body section lines to number. The recognized **types** are:

a	Numbers all lines.
t	Does not number blank lines (default).
n	Does not number any lines.
ppattern	Numbers only those lines containing the specified **pattern**.

-dxx Uses **xx** as the delimiters for the start of a logical page section. The default characters are \: (backslash followed by a colon). You can specify two 1-byte ASCII characters or one 2-byte character. If you enter only one 1-byte character after **-d**, the second character remains the default (colon). If you want to use a backslash as a delimiter, enter two backslashes (\\).

-ftype Chooses which logical page footer lines to number. The **types** recognized are the same as in **-btype**. The default **type** is **n** (no lines numbered).

-htype Chooses which logical page header lines to number. The **types** recognized are the same as in **-btype**. The default **type** is **n** (no lines numbered).

-inum Increments logical page line numbers by **num**. The default value of **num** is 1.

-lnum Uses **num** as the number of blank lines to count as one. For example, **-l3** will only number the third adjacent blank. The default value of **num** is **2**. This flag can only be used in documents where the **-ba** flag is used.

-n format Uses **format** as the line numbering format. Recognized formats are:

ln	Left justified, leading zeroes suppressed.
rn	Right justified, leading zeroes suppressed (default).
rz	Right justified, leading zeroes kept.

-p Does not restart numbering at logical page delimiters.

-s [sep] Separates the text from its line number by the **sep** characters. The default value of **sep** is a tab character. If you enter **-s** without a parameter, there is no separation between the line number and its text.

-vnum Sets the initial logical page line number to **num** (1 by default).

-wnum Uses **num** as the number of characters in the line number. The default value of **num** is 6.

Examples

1. To number only the nonblank lines:

```
nl  chap1
```

This command displays a numbered listing of **chap1**, numbering only the nonblank lines in the body sections. If **chap1** contains no **\:\:\+ :**, **\:\+ :**, or **\:** delimiters, the entire file is considered the body.

2. To number all lines:

```
nl  -ba  chap1
```

This command numbers all the lines in the body sections, including blank lines. This form of the **nl** command is adequate for most uses.

3. To specify a different line number format:

```
nl  -i10  -nrz  -s::  -v10  -w4  chap1
```

This command numbers the lines of **chap1** starting with line 10 (**-v10**) and counting by tens (**-i10**). It displays four digits for each number (**-w4**), including leading zeroes (**-nrz**). The line numbers are separated from the text by two colons (**-s::**).

For example, if **chap1** contains the text:

```
A not-so-important
note to remember:

You can't kill time
without injuring eternity.
```

then the numbered listing is:

```
0010::A not-so-important
0020::note to remember:

0030::You can't kill time
0040::without injuring eternity.
```

The blank line is not numbered. To number all lines including blank ones, use the **-ba** flag as shown in Example 2.

Related Information

See the following command: "pr" in topic 1.1.322.

See "Overview of International Character Support" in *Managing the AIX Operating System*.

1.1.298 nm

Purpose
Displays the symbol table of an object file.

Syntax

```
      +--------+   +--------+   +--------+
nm ---¦ +----+ +---¦ one of +---¦ one of +--- file ---¦
      +-¦ -e +-+   ¦ +----+ ¦   ¦ +----+ ¦          ¦
        ¦ -f ¦     +-¦ -n +-+   +-¦ -d +-+ +--------+
       ¦¦ -h ¦¦      ¦ -v ¦       ¦ -o ¦
       ¦¦ -r ¦¦      +----+       ¦ -x ¦
       ¦¦ -T ¦¦                   +----+
       ¦¦ -u ¦¦
       ¦¦ -O ¦¦
       ¦+----+¦
       +------+
```

Description
The **nm** command writes the symbol table of each specified object **file** to standard output. **file** can be a single relocatable or absolute common object file or an archive library of relocatable or absolute common object files. **nm** displays the following information for each symbol:

Name	The name of the symbol.
Value	Its value expressed as an offset or an address depending on its storage class.
Class	Its storage class.
Type	Its type and derived type. If the symbol refers to a structure or a union, the structure or union tag follows the type declaration. If the symbol is an array, the array dimensions follow the type.
Size	Its size in bytes, if available.
Line	The source line number at which it is defined, if available.
Section	For static and external storage classes, this is the object file section containing the symbol.

Note: The debug information (type, size and line) is not currently put in the symbol table of object files generated by the AIX Compilers (C, VS, FORTRAN, and VS Pascal).

Flags

-e Displays only static and external symbols.

-f Displays the symbols (.text, .data, .lib and .bss) that are normally suppressed.

-h Does not display output header data.

-d Displays a symbol's value and size as a decimal number.

-n Sorts external symbols by name before displaying them.

-o Displays a symbol's value and size as an octal rather than a decimal number.

-O Prepends file/archive name to each symbol.

-r Sorts symbols in reverse order.

-T Truncates every name that would otherwise overflow its column, making the last character displayed in the name an asterisk. By default, **nm** displays the entire name of the symbols listed, and a name that is longer than the width of the column set aside for it causes every column after the name to be misaligned.

-u Displays only undefined symbols.

-v Sorts external symbols by value before displaying them.

-x Displays a symbol's value and size as a hexadecimal rather than a decimal number.

Files

a.out	Default input file.

Related Information

See the following commands: "ar" in topic 1.1.23, "as" in topic 1.1.25, "backup" in topic 1.1.32, "cc" in topic 1.1.52, "ld" in topic 1.1.226, "nm," "size" in topic 1.1.427 , and "strip" in topic 1.1.445.

See the **a.out** and **ar** files in *AIX Operating System Technical Reference*.

1.1.299 nm - BSD Version

Purpose
Displays the symbol table of an object file.

Syntax

```
                +--------+
/usr/ucb/nm ---¦ +----+ +--- file ---¦
                +-¦ -a +-+          ¦
                  ¦ -g ¦¦  +--------+
                 ¦¦ -n ¦¦
                 ¦¦ -o ¦¦
                 ¦¦ -p ¦¦
                 ¦¦ -r ¦¦
                 ¦¦ -u ¦¦
                 ¦+----+¦
                 +------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **nm** command prints the name list (symbol table) of each object **file** in the argument list. If an argument is an archive, a listing for each object file in the archive will be produced. If no **file** is given, the symbols in **a.out** are listed.

Each symbol name is preceded by its value (blanks if undefined) and one of the letters **U** (undefined), **A** (absolute), **T** (text segment symbol), **D** (data segment symbol), **B** (bss segment symbol), **C** (common symbol), **f** (file name), or **-** for debugger symbol table entries (see **-a** below). If the symbol is local (non-external) the type letter is in lower case. The output is sorted alphabetically.

Flags

-a	Print symbol table entries inserted for use by debuggers.
-g	Print only global (external) symbols.
-n	Sort numerically rather than alphabetically.
-o	Prepend file or archive element name to each output line rather than only once.
-p	Don't sort; print in symbol-table order.
-r	Sort in reverse order.
-u	Print only undefined symbols.

Related Information

See the following commands: "ar" in topic 1.1.23 and "nm" in topic 1.1.298.

1.1.300 nohup

Purpose
Runs a command without hangups and then quits.

Syntax

nohup -- **command** --¦

Description
The **nohup** command runs **command**, ignoring all hangups and **quit** signals. You can use this command to run programs in the background after you log out of the system. To run a **nohup** command in the background, add an **&** to the end of the command.

If **nohup** command output is redirected to a terminal or is not redirected at all, the output goes to the file **nohup.out**. If **nohup.out** is not writable in the current directory, the output is redirected to the file **$HOME/nohup.out**.

The syntax of this command ignores quits and hangups for only one **command**. If you want to apply **nohup** to a pipeline or list of commands, you can put the pipeline or list in a shell script file. Then you can run the **sh** command as the **command** using the format: **nohup sh file**. You can also assign the shell file execute permission and run it as the command in the form: **nohup file**.

Examples

1. To let a command run after you log out:

   ```
   nohup  find  /  -print  &
   ```

   Shortly after you enter this, a message such as the following is displayed:

   ```
   670
   $ Sending output to nohup.out
   ```

   The number displayed the ID of the background process. (See page 1.1.420.1 about starting background processes). The **$** (dollar sign) is your shell prompt. **Sending output...** is a message from **nohup** telling you that it is storing the output from the **.find** command in the file **nohup.out**. You can log out after you see these messages, even if the **find** command has not finished yet.

2. To do the same, but redirecting the standard output to a different file:

   ```
   nohup  find  /  -print  >filenames  &
   ```

   This command runs the **find** command and stores its output in a file named **filenames**. Now only the process ID and your prompt are displayed:

   ```
   677
   $
   ```

   Wait for a second or two before logging out, because the **nohup** command takes a moment to start the command you specify. If you log out too quickly, your command may not run at all. Once your command starts, logging out does not affect it.

3. To run more than one command, use a shell procedure. For example, if you write the shell procedure:

   ```
   neqn math1 | nroff > fmath1
   ```

   and name it **nnmath1**, you can run the **nohup** command for all of the commands in the file **nnmath1** with the command:

   ```
   nohup sh nnmath1
   ```

   If you assign **nnmath1** execute permission, you can obtain the same results by issuing the command:

   ```
   nohup nnmath1
   ```

   To run this command in the background, enter the command:

   ```
   nohup nnmath1 &
   ```

Related Information

See the following commands: "csh" in topic 1.1.100, "nice" in topic 1.1.296 and "sh, Rsh" in topic 1.1.420.

Note: The **csh** command contains a built-in subcommand named **nohup**. The command and subcommand do not necessarily work the same way. For information on the subcommand, see the **csh** command.

See the **signal** system call in *AIX Operating System Technical Reference*.

1.1.301 nroff, troff

1.1.302 nroff, troff

Purpose
Formats text for printing devices.

Syntax

```
         +------------+   +--- -T37 -s1 ---+   +------------+
nroff ---¦   one of   +---¦ +------------+ +---¦            +---¦
         ¦ +--------+ ¦   +-¦ -unum      +-+   +--- file ---+
         +-¦ -olist +-+     ¦ -snum   -q ¦¦              ¦
           ¦ -nnum  ¦      ¦¦ -raN    -z ¦¦      +--------+
           +--------+      ¦¦ -mname  -e ¦¦
                           ¦¦ -Tname  -h ¦¦
                           ¦+------------+¦
                           +--------------+

troff --- -b ---¦

         +--- -n1 ----+   +--------+   +----- -s1 ------+   +------------+
troff ---¦   one of   +---¦ one of +---¦ +------------+ +---¦            +---¦
         ¦ +--------+ ¦   ¦ +----+ ¦   +-¦ -snum   -i +-+   +--- file ---+
         +-¦ -olist +-+   +-¦ -o +-+     ¦ -raN    -a ¦¦               ¦
           ¦ -n     ¦       ¦ -t ¦      ¦¦ -pnum   -z ¦¦     +--------+
           +--------+       +----+      ¦¦ -f      -w ¦¦
                                        ¦¦ -mname  -q ¦¦
                                        ¦+------------+¦
                                        +--------------+
```

Note: This command does not have MBCS support.

Description

Note: A complete list of **nroff** and **troff** requests, escape sequences, and number registers begins on page 1.1.302.5. See the *AIX Operating System Text Formatting Guide* for a complete list of the naming conventions for the non-ASCII special characters and for information on writing text suitable for processing by the **troff** or **nroff** command.

Subtopics

1.1.302.1 nroff

The **nroff** command reads **file**s (standard input, by default), formats the text in its input for printing, and writes to standard output. The **nroff** command formats text for line printers and other printing devices, excluding photo typesetters. An input file name of **-** (minus) indicates standard input.

1.1.302.2 troff

The **troff** command formats text in the input **file**s (or standard input, by default) for a photo typesetter and writes its output to standard output. It is similar to the **nroff** command. An input file name of **-** (minus) indicates standard input.

Flags

1.1.302.3 nroff and troff Flags

-i	Reads standard input after the input files.
-mname	Adds **/usr/lib/tmac/tmac.name** to the beginning of the list of input file names.
-nnum	Numbers the first printed page **num**. Do not use this flag with **-olist**.
-olist	Prints only pages with page numbers appearing in **list** which consist of a comma-separated list of page numbers and ranges. A range of **A-B** means print pages **A** through **B**; an initial **-A** means print from the beginning to page **A**; and a final **A-** means print from page **A** to the end. **Note:** When this flag is used in a pipeline (for example, with the **cw**, **eqn**, or **tbl** command) it may cause a broken pipe diagnostic if the last page in the document is not specified in **list**.
-q	Invokes the simultaneous input/output mode of the **.rd** request. The **nroff** command echoes the **.rd** prompt, but does not echo your input. When you enter two consecutive new-line characters, normal output is resumed.
-raN	Sets register **a** to **N**. What you specify for **a** must be a one-character name. This flag is useful for automatic numbering of sections, paragraphs, lines, and so forth.
-snum	Stops every **num** pages (the default is 1). The **nroff** or **troff** command halts every **num** pages to allow paper loading or changing and resumes upon receipt of a line-feed or new-line character. This flag does not work in pipelines. When the **nroff** command halts between pages, an ASCII BEL character is sent to the printing device.
-z	Suppresses the formatted output. Prints only messages generated by **.tm** (work station message) requests.

1.1.302.4 nroff Flags:

-e	Produces equally spaced words in adjusted lines, using the full resolution of the printing device.
-h	Uses tab characters during horizontal spacing. Tab settings are assumed to be every eight spaces.
-Tname	Prepares the output for the specified printing device. You can specify the following for **name**:

37	TELETYPE Model 37 work station (default)
tn300	GE TermiNet 300 or any work station without half-line capability
300s	DASI 300s
300	DASI 300
450	DASI 450
lp	Any ASCII line printer
382	DCT-382
4000A	Trendata 4000A
832	Anderson Jacobson 832
X	Any EBCDIC printer
2631	Hewlett Packard 2631 line printer. Use a **name** of **2631-c** to get compressed print or **2631-e** to get expanded print.

-unum	Sets the number of character overstrikes for boldface to **num** or to zero if **num** is not specified.

1.1.302.5 troff Flags:

-a	Sends a printable ASCII approximation of the output to standard output.
-b	Reports whether the photo typesetter is busy or available. No text processing is done.
-f	Does not feed out paper and stop the photo typesetter at the end of the run.
-pnum	Prints all characters in the point size specified by **num**. Smaller point sizes may reduce the printing time.
-t	Directs output without modification to standard output instead of the photo typesetter.
-w	Waits until the photo typesetter is available if it is currently busy.

nroff and troff Requests

Request Form	**Function -- Font and Character Size Control**
.ps ± N	Change point size by **N** points. Also, for **troff** only, \s ±**N**.
.ss N	Space-character size set to **N**/36 em (**troff**) only.
.cs F N M	Constant character space (width) mode (font**F**) (**troff** only).
.bd F N	Embolden font **F** by **N** units (**troff** only).
.bd S F N	Embolden Special Font when current font is **F** (**troff** only).
.ft F	Change to font **F**.
.fp N F	Mount font **F** on position **N** (1-4).

Request Form	**Function -- Page Control**
.pl ± N	Change page length by **N**.
.bp ± N	Eject current page, next page number is **N**.
.pn N	Next page number is **N**.
.po ±N	Page offset = **N**.
.ne N	Need **N** vertical space.
.mk R	Mark current vertical place in register **R**.

.rt ±N	Return (upward only) to marked vertical place.

Request Form	**Function -- Text Filling, Adjusting, and Centering**
.br	Break.
.fi	Fill subsequent output lines.
.nf	No filling or adjusting of output lines.
.ad [**c**]	Adjust output lines with mode **c**.
.na	Do not adjust output lines.
.ce N	Center the following **N** lines.

Request Form	**Function -- Vertical Spacing**
.vs N	Set vertical base-line spacing to **N**.
.ls N	Output **N**-1 base-line spaces after each text output line.
.sp N	Space vertical distance **N** in either direction.
.sv N	Save vertical distance **N**.
.os	Output saved vertical space.
.ns	Turn no-space mode on.
.rs	Restore spacing, turn no-space mode off.

Request Form	**Function -- Line Length and Indenting**
.li ±N	Change line length by **N**.
.in ±N	Change indenting by **N**.
.ti ±N	Change indenting on the next line by **N**.

Request Form	**Function -- Macros, Strings, Diversion, and Position Traps**
.de xx yy	Define or redefine macro **xx**; end at call of **yy**.

.am xx yy	Append to a macro.
.ds xx string	Define a string **xx** containing **string**.
.as xx string	Append **string** to string **xx**.
.rm xx	Remove request, macro, or string named **xx**.
.rn xx yy	Rename request, macro, or string **xx** to **yy**.
.di xx	Divert output to macro **xx**.
.da xx	Divert and append to **xx**.
.wh N xx	Set location trap; negative is with respect to the end of the page.
.ch xx N	Change trap location.
.dt N xx	Set a diversion trap.
.it N xx	Set an input line trap.
.em xx	End macro is **xx**.

Request Form	**Function -- Number Registers**
.nr R ±N M	Define and set number register **R**; auto-increment by **M**.
.af R c	Assign format to register **R** (**c**=1, i, I, a, A).
.rr R	Remove register **R**.

Request Form	**Function -- Tabs, Leaders, and Fields**
.ta Nt...	Tab settings; left type, unless **t**=R (right) or C (centered).
.tc c	Tab repetition character.
.lc c	Leader repetition character.
.fc a b	Set field delimiter **a** and pad character **b**.

Request Form	**Function -- Input/Output Conventions and Character Translations**

.ec c	Set escape character.
.eo	Turn off escape character mechanism.
.lg N	Ligature on if **N**>0.
.ul N	Underline in **nroff** or italicize in **troff** the next **N** input lines.
.cu N	Continuous underline in **nroff**. Acts like **.ul** in **troff**.
.uf F	Underline font set to **F** (to be switched to by **.ul**).
.cc c	Set control character to **c**.
.c2 c	Set no-break control character to **c**.
.tr abcd...	Translates **a** to **b**, and so on, on output.

Request Form	**Function -- Hyphenation**
.nh	No hyphenation.
.hy H	Hyphenate; **N** = mode.
.hc c	Hyphenation indicator character **c**.
.wc word...	Exception words.

Request Form	**Function -- Three Part Titles**
.tl 'left'center'right'	Three part title.
.pc c	Page number character.
.lt ±N	Length of title.

Request Form	**Function -- Output Line Numbering**
.nm ±N M S I	Number mode on or off, set parameters.
.nm N	Do not number next **N** lines.

Request Form	**Function -- Conditional Acceptance of Input**
.if c.anything	If condition **c** is true, accept **anything** as input. For multiple lines, use **\{anything\}**.
.if !c anything	If condition **c** is false, accept **anything** as input.
.if N anything	If expression **N**>0, accept **anything** as input.
.if ! N anything	If expression **N**=0, accept **anything** as input.
.if 'string1'string2' anything	If **string1** is identical to **string2**, accept **anything** as input.
.if !'string1'string2.' anything	If **string1** is not identical to **string2**, accept **anything** as input.
.ie c anything	**If** part of **if-else** statement; can take all forms of **if** above.
.el anything	**Else** part of **if-else** statement.

Request Form	**Function -- Environment Switching**
.ev N	Environment switched (push down).

Request Form	**Function -- Insertions from Standard Input**
.rd prompt	Read insertion.
.ex	Exit from **nroff** or **troff**.

Request Form	**Function -- Input/Output File Switching**
.so file	Switch source file (push down).
.nx file	Next file.
.pi program	Pipe output to **program** (**nroff** only).

Request Form	**Function -- Miscellaneous**
.mc c N	Set margin character **c** and separation **N**.
.tm string	Print **string** on standard error output.
.ig yy	Ignore untill call of **yy**.
.pm t	Print macro names and sizes; if **t** is present, print only the total of sizes.
.fl	Flush output buffer.
.ab [**text**]	Prints **text** on standard error output and stops output. **User abort** message is printed if no **text** is included.
! cmd parms	Runs the AIX command **cmd** and interpolates at that point. The standard input for **cmd** is closed.

1.1.302.6 Escape Sequences for Characters, Indicators, and Functions

Escape Sequence	**Meaning**
\\	Prevents or delays interpretation of \.
\e	Printable version of the current escape character.
\`.	Acute accent; equivalent to \(aa.
\`.	Grave accent; equivalent to \(ga.
\-	Minus sign in the current font.
\.	Dot.
\(space)	Unpaddable space-size character.
\0	Digit width space.
\¦	1/6 em narrow space character (zero width in **nroff**).
\^	1/12 em half-narrow space character (zero width in **nroff**).
\&	Non-printing, zero-width character.
\!	Transparent line indicator.
\$N	Interpolate argument 1=**N**=9.
\%	Default optional hyphenation character.
\(xx	Character named **xx**.
\*x, *(xx	Interpolate string **x** or **xx**.
\a	Non-interpreted leader character.
\b'abc...'	Bracket building function.
\c	Interrupt text processing (continue word across input line break).
\d	Forward (down) 1/2 em vertical motion (1/2 line in **nroff**).
\fx, \f(xx, \fN	Change to font **N** named **x** or **xx**, or font position **N**.
\gx, \g(xx	Return the format of register **x** or **xx**. Return nothing if the register has not yet been referenced.
\h'N'	Local horizontal motion; move right **N** (negative left).
\jx, \jxx	Mark in register **x** or **xx** the current horizontal position on the output line.

\kx	Mark horizontal input place in register **x**.
\l'N[c]'	Horizontal line drawing function.
\L'N[c]'	Vertical line drawing function.
\nx, \(xx	Interpolate number register **x** or **xx**.
\o'abc...'	Overstrike characters **a**, **b**, **c**, ...
\p	Break and spread output line.
\r	Reverse 1 em vertical motion (reverse line in **nroff**).
\sN, \s±N	Point-size change function.
\t	Non-interpreted horizontal tab.
\u	Reverse (up) 1/2 em vertical motion (1/2 line in **nroff**).
\v'N'	Local vertical motion ; move down **N** (negative up).
\w'string'	Interpolate width of **string**.
\x'N'	Extra line-space function (negative before, positive after).
\zc	Print **c** with zero width without spacing.
\{	Begin conditional input.
\}	End conditional input.
\(new line)	Concealed new line.
\X	**X**, any character not listed above.

1.1.302.7 Predefined General Number Registers

Register Name	**Description**
%	Current page number.
ct	Character width type (set by width function).
dl	Maximum width of last completed diversion.
dn	Height (vertical size) of last completed diversion.
dw	Current day of the week (1=Sunday ... 7=Saturday).
dy	Current day of the month (1-31).
hp	Current horizontal place on the input line.
ln	Output line number.
mo	Current month (1-12).
nl	Vertical position of last printed text base-line.
sb	Depth of string below base line (generated by width function).
st	Height of string above base line (generated by width function).
yr	Last two digits of current year.

1.1.302.8 Predefined Read-Only Number Registers

Register Name	**Meaning**
$	Number of arguments available at the current macro level.
.A	Set to 1 in **troff** if the **-a** flag is used; always 1 in **nroff**.
.F	The name of the current input file.
.H	Available horizontal resolution in basic units.
.L	Contains the current line spacing parameter.
.P	Contains the value 1 if the current page is being printed; otherwise contains 0.
.R	The number of columns available.
.T	Set to 1 in **nroff**, it the **-T** flag is used; always 0 in **troff**.
.V	Available vertical resolution in basic units.
.a	Post-line extra line-space most recently utilized using **\s'N'**.
.b	Emboldening factor of the current font.
.c	Number of lines read from current input file, including **.so** files.
.d	Current vertical place in current diversion; equal to **nl** if no diversion.
.f	Current font as physical quadrant.
.h	Text base-line high-water mark on current page or diversion.
.i	Current indent.
.j	Current adjustment mode and type.
.k	Contains the horizontal size of the text portion of the current, partially-collected output line, if any, in the current environment.
.l	Current line length.
.n	Length of text portion on previous output line.
.o	Current page offset.
.p	Current page length.
.s	Current point size.

.t	Distance to the next trap.
.u	Equal to 1 in fill mode; equal to 0 in no-fill mode.
.v	Current vertical line spacing.
.w	Width of previous character.
.x	Reserved version-dependent register.
.y	Reserved version-dependent register.
.z	Name of current diversion.

Files

/usr/lib/suftab	Suffix hyphenation tables.
/tmp/ta$#	Temporary file.
/usr/lib/tmac/tmac.*	Standard macro files.
/usr/lib/macros/*	Standard macro files.
/usr/lib/font/*	Font width tables for the **troff** command.
/usr/lib/term/*	Work station driving tables for the **nroff** command.

Related Information

See the following commands: "col" in topic 1.1.77, "cw, checkcw" in topic 1.1.108, "eqn, neqn, checkeq" in topic 1.1.152, "mm, checkmm" in topic 1.1.274, "mmt, mant, mvt" in topic 1.1.275, "tbl" in topic 1.1.463, and "tc" in topic 1.1.464.

See the **mm** miscellaneous facility in *AIX Operating System Technical Reference*.

See the discussion of **nroff** and **troff** in *Text Formatting Guide*.

1.1.303 number

Purpose
Displays the written form of a number.

Syntax

/usr/games/number ---¦

Note: This command does not have MBCS support.

Description
The **number** game displays the written form of a number that it reads from standard input. The largest number it can translate accurately contains 66 digits.

The **number** game does not prompt you for a number. Once loaded, it simply waits for input. To exit the program, press the INTERRUPT (**refer to keyboard definition**) or END OF FILE (**Ctrl-D**) key.

Example
To display the written form of several numbers:

```
You:
    /usr/games/number
    829
System:
      eight hundred twenty nine
      ....
You:
    12345678
System:
      twelve million.
     three hundred forty five thousand.
     six hundred seventy eight.
You:
     Ctrl-D
```

1.1.304 od

Purpose
Writes the contents of storage to standard output.

Syntax

```
      +------- -o -------+
      ¦      one of      ¦
      ¦    +--------+    ¦  +--------------+  +---------------+  +------+
od ---¦ +--¦ -a  -c +--+ +--¦      +- 3 -+ +--¦      +- 32 -+ +--¦      +---
      ¦ ¦  ¦ -b  -C ¦  ¦ ¦  +- -s -¦     +-+  +- -w -¦      +-+  +- -v -+
      ¦ ¦  +--------+  ¦ ¦         +- n -+           +- n --+
      +-¦    one of    +-+
        ¦ +----------+ ¦
        ¦ ¦ -d -F -L ¦ ¦
        ¦ ¦ -D -H -o ¦ ¦
        +-¦    -i    +-+
          ¦ -f -I -x ¦
          ¦ -P -p -l ¦
          +----------+

    +--------+   +-------------- +0 --------------+
 ---¦        +---¦ +-----+¦+- num ---+ +--------+ +---¦
    +- file -+   +-¦     +-+- num. --+-¦ one of +-+
                   +- + -+ +- 0xnum -¦ ¦ ¦+---+ ¦
                           +- xnum --+ +--¦ b +-+
                                          ¦ B ¦
                                          +---+
```

¦ Do not put a blank before these items.

Description

The **od** command reads the specified **file** (standard input by default) and writes to standard output the information stored in the file using the format specified by the first flag. If you do not specify the first flag, **-o** is the default. When the **od** command reads standard input, the **num** argument must be preceded by + (plus sign).

Flags

Format characters are:

-a Interprets bytes as characters and displays them with their ASCII names. If the **-P** flag is given also, bytes with even parity are underlined. The **-p** flag causes bytes with odd parity to be underlined. Otherwise, parity is ignored. The **-p** and **-P** flags only work with the **-a** flag.

-b Displays bytes as unsigned octal values.

-c Displays bytes as ASCII characters. The following nongraphic characters appear as C escapes sequences:

\0	Null
\b	Backspace
\f	Form feed

\n New-line character
\r Return
\t Tab
\s1
\s2
\s3
\s4 Extended character shifts.

Others appear as 3-digit octal numbers. Bytes with the parity bit set are displayed in octal.

-C Displays any extended characters as standard printable ASCII characters using the appropriate character escape string.

-d Displays 16-bit words as unsigned decimal values.

-f Interprets long words as floating point.

-h Interprets (short) words as unsigned hexadecimal.

-i Interprets (short) words as signed decimal.

-l Interprets long words as signed decimal.

-o Displays 16-bit words as octal values.

-p Causes bytes with odd parity to be underlined and modifies the behavior of the **-a** flag.

-P Causes bytes with even parity to be underlined and modifies the behavior of the **-a** flag.

-s[n]
Looks for strings of ASCII graphic characters, terminated with a null byte. The **n** argument specifies the minimum length string to be recognized. By default, the minimum length is 3 characters.

-v Shows all data. By default, display lines that are identical to the last line shown are not output but are indicated with an * (asterisk) in column 1.

-w[n]
Specifies the number of input bytes to be interpreted and displayed on each output line. If **-w** is not specified, 16 bytes are read for each display line. If the **n** argument is not specified, it defaults to 32.

-x Displays 16-bit words as hexadecimal values.

An uppercase format character implies the long or double precision form of the object. Short implies 16-bit words; long implies 32-bit words.

The **offset** parameter specifies the point in the file where the storage output starts. The **offset** parameter is interpreted as octal bytes. If a . (dot) is added to **offset**, it is interpreted in decimal. If **offset** begins with x or 0x, it is interpreted in hexadecimal. If b (B) is appended, the offset is interpreted as a block count, where a block is 512 (1024) bytes.

Label will be interpreted as a pseudo-address for the first byte displayed. It will be shown in parentheses "()" following the file offset. It is intended to be used with core images to indicate the real memory address. The syntax for **label** is identical to that for **offset**.

The storage output continues until the end of the file.

Notes:

1. When you are viewing a file with one of the options which specifies a 16- or 32-bit word, the output appears differently depending on whether the command is run on an AIX/370 or AIX PS/2 machine because the two machines represent binary data in different ways.

2. Looking at directories with the **od** command invokes the UNIX System V directory compatibility code which truncates entries to 14 characters. To see the actual directory contents, use the **fsdb** command. For a description of directory compatibility, see **ulimit** in the *AIX Operating System Technical Reference.*

Examples

1. To display a file in octal a page at a time:

   ```
   od  a.out  |  pg
   ```

   This displays **a.out** in octal (base 8) word format a page at a time.

2. To translate a file into several formats at once:

   ```
   od  -cx  a.out  >a.xcd
   ```

   This writes **a.out** in hexadecimal (base 16) format (**-x**) into the file **a.xcd**, giving also the ASCII character equivalent, if any, of each byte (**-c**).

3. To start in the middle of a file:

   ```
   od  -bcx  a.out  +100.
   ```

   This displays **a.out** in octal-byte, character, and hexadecimal formats, starting from the **100**th byte. The . (dot) after the offset makes it a decimal number. Without the dot, the dump would start from the 64th (100 octal) byte.

Related Information

See the following command: "pg" in topic 1.1.315.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.305 onrpc

Purpose

Execute a command remotely.

Syntax

```
        +--------+   +------+   +-- sitename --+
onrpc ---¦ one of +---¦      +---+- sitenumber -+--- command --- argument ---
        ¦ +----+ ¦   +- -d -+   +-- sitetype --+                          ¦
        +-¦ -i +-+                                            +-----------+
          ¦ -n ¦
          +----+
```

Note: This command does not have MBCS support.

Description

The **onrpc** command program is used to execute commands on another system, in an environment similar to that invoking the program. All environment variables are passed, and the current working directory is preserved. To preserve the working directory, the working file system must be either already mounted on the host or be exported to it. Relative path names will only work if they are within the current file system; absolute path names may cause problems.

Standard input is connected to standard input of the remote command, and standard output and standard error from the remote command are sent to the corresponding files for the **onrpc** command.

Notes:

1. The **onrpc** command cannot be run by the root user.

2. The **onrpc** command is the functional equivalent of the UNIX **on** command for NFS. The AIX **on** command is the TCF version.

Flags

-i Interactive mode: use remote echoing and special character processing. This option is needed for programs that expect to be talking to a terminal. All terminal modes and window size changes are propagated.

-n No input: this option causes the remote program to get end-of-file when it reads from standard input, instead of passing standard input from the standard input of the **onrpc** program. For example, **-n** is necessary when running commands in the background with job control.

-d Debug mode: print out some messages as work is being done.

Related Information

See the following commands: "rpc.rexd" in topic 1.1.387 and "rpcinfo" in topic 1.1.384.

1.1.306 onsite, on

Purpose
Runs a command at a specific site in a TCF cluster.

Syntax

```
      one of
+-----------------+  +------+   +- sitename ---+
¦ /usr/bin/onsite +--¦      +---+- sitenumber -+--- command --- argument ---¦
¦ /usr/bin/on     ¦  +- -v -+   +- sitetype ---+                            ¦
+-----------------+                                          +------------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **onsite** command runs **command** with optional **arguments** on the site specified by **sitename**, **sitenumber**, or **sitetype**. Site names, numbers and types are defined in the site data base, **/etc/site**. The following types are valid:

  i386, i370, xa370

When a **sitetype** is specified, **onsite** selects a machine of the specified type on which to run **command**.

The **onsite** command is essentially a **setlocal** followed by an **rexecve**. (See the *AIX Operating System Technical Reference*.) The arguments to these operations are derived from information contained in the site data base, **/etc/site**. The **setlocal** call indicates that the destination site's **<LOCAL>** is to be used to interpret **command** and as an environment in which **command** will run (see **local**). In particular, if **command** references files in **/tmp**, it references files in its local **/tmp**.

A particularly useful command to run with **onsite** is a shell. In a remote shell, commands are executed on the remote site almost exactly as though the terminal is attached to the remote machine. One noticeable exception is that **Ctrl-T** (which gives machine status, load and uptime) gives information about the machine to which the terminal is really connected and not the remote site.

It is possible to use **onsite** and specify the local site and guarantee that **command** is executed locally or not at all.

Note: Both the **csh** and **sh** have a built in version of **onsite**. This permits the alias mechanisms of both shells to be supported properly. This version does not properly handle commands which are aliases. Furthermore, all wild card expansions are done in the context of the invoking process, not of the new site.

Flag

-v Verbose option; displays the name of the site on which the command is being run.

Restriction
Shell files that do not begin with '#!' cannot be run with **onsite** since

the kernel doesn't recognize these as command files. Again, this problem is eliminated if you use the **onsite** that is built into the shells.

Related Information

See the following commands: "anet" in topic 1.1.18, "csh" in topic 1.1.100, "sh, Rsh" in topic 1.1.420, "fast, fastsite" in topic 1.1.161, and "loads" in topic 1.1.236.

See in the *AIX Technical Reference Vol 1*: "setlocal", "rexecve".

See in the *AIX Technical Reference Vol II*: "site" in the File formats.

1.1.307 open

Purpose
Opens a virtual terminal.

Syntax

```
               +-------+
open -- cmd --¦       +---¦
               +- arg -+
                      ¦
                +-----+
```

Note: This command is for the PS/2 only.

Description
The **open** command opens a virtual terminal and runs the specified **cmd** on that terminal. If **cmd** does not reside in one of the directories specified in the shell **PATH** variable, you must give a full path name. Any arguments that follow **cmd** on the command line are passed to **cmd**. To move from one virtual terminal to another, press the Next Window (**Alt-Action**) key. To close a virtual terminal, press the END OF FILE (**Ctrl-D**) key or end the application.

Notes:

1. Be sure to use the **Alt-Action** key to check for open virtual terminals and the **Ctrl-D** key to close them before you log off.

2. If you run the **actmngr** command before opening any virtual terminals, the activity manager helps you to keep track of virtual terminals so you can close all of them before you log off.

Subtopics

1.1.307.1 Usability Services Commands

These additional commands are available from within the Usability Services Activity Manager (**/usr/bin/actmngr**):

hide	Removes an activity window from the window ring.
activate	Activates an activity window.
cancel	Cancels an activity window.

For details about using these commands, see *Usability Services Reference* or *Usability Services Guide*.

Example

To run another shell on a new virtual terminal:

```
  open sh
```

To move back and forth between this new virtual terminal and any others that you have opened, press the Next Window (**Alt-Action**) key. To close this terminal and log off the new shell, press the END OF FILE (**Ctrl-D**) key.

Related Information

See "Using Display Station Features" in *Using the AIX Operating System*.

1.1.308 osconfig

Purpose
Installs AIX device drivers.

Syntax

```
                   +----------------+   +----------------+   +----------------+
/etc/osconfig ---¦     one of      +---¦     one of      +---¦ +------------+ +-
                   ¦ +------------+ ¦   ¦ +------------+ ¦   +-¦ -c stanza  +-+
                   +-¦ -a stanza  +-+   +-¦ -d stanza  +-+     ¦ -h helpdir ¦
                     ¦ -as stanza ¦       ¦ -ds stanza ¦      ¦¦ -m mfile   ¦¦
                     +------------+       +------------+      ¦¦ -s syfile  ¦¦
                                                               ¦¦ -startup   ¦¦
                                                               ¦+------------+¦
                                                               +--------------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **osconfig** command installs AIX device drivers. Normally, it runs automatically at each system startup. Its exit value is the number of errors it encounters. The **osconfig** command performs its operations through the **/dev/config** driver.

Note: Most users never need to run this command from the command line.

Flags

-a stanza	Adds devices to a running system. The **osconfig** command processes the specified **stanza** in the file **/etc/system** or in the file specified with the **-s** flag.
-as stanza	Same as the **-a** flag except that the **osconfig** command does not invoke the customization helper for the device; it only creates the device special file.
-d stanza	Deletes a device from a running system. The **osconfig** command processes the specified **stanza** in the file **/etc/system** or in the file specified with the **-s** flag. The special file list produced includes commands to remove relevant special files, since the **osconfig** command assumes that the device has been removed from the configuration.
-ds stanza	Same as the **-d** flag except that the **osconfig** command does not invoke the customization helper for the device; it only deletes the device special file.
-c stanza	Changes devices on a running system. The **osconfig** command processes the specified **stanza** in the file **/etc/system** or in the file specified with the **-s** flag and sends the appropriate information to the device driver.
-h helpdir	Specifies the directory that contains the configuration helper programs. The default value is the file **/etc**.
-m mfile	Specifies the input master configuration file. The default value is the file **/etc/master**.

-s syfile	Specifies the input system configuration file. The default value is the file **/etc/system**.
-startup	Causes all defined devices to be configured in at system startup. For devices such as minidisks, which remain configured between system restarts, the **osconfig** command does not perform once-only configuration steps. It does not modify special files that already exist.

Files

/etc/confgstatus	Status file recording current configuration status
/etc/system	Default system file
/etc/master	Default master file
specials	Default special file list
/etc	Default directory containing configuration helper programs
/etc/ddi	Directory containing device specific information files

Related Information
See the following command: "config" in topic 1.1.88.

See the **master** and **system** files in *AIX Operating System Technical Reference.*

1.1.309 pack, unpack, pcat

Purpose
Compresses and decompresses files.

Syntax

```
        +------+
pack ---¦+----++-- file --¦
        +¦ -  ++        ¦
         ¦ -f ¦  +------+
         +----+
  one of
+--------+   +-- file --+
¦ unpack +---¦          +---¦
¦ pcat   ¦   +- file.z -+ ¦
+--------+ +--------------+
```

Description

The **pack** command stores the specified **file** in a compressed form. The **file** is replaced by a packed file with a name derived from the original file name (**file.z**), with the same access modes, access and modification dates, and owner as the original file. Directories cannot be compressed.

If **pack** cannot create a smaller file, it stops processing and reports that it is unable to save space. (A failure to save space generally happens with small files or files with uniform character distribution.) The amount of space saved depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each **.z** file, you generally are not able to save space with files smaller than three blocks. Typically, text files are reduced 25 to 40 percent.

The exit value of the **pack** command is the number of files that it could not pack. Packing is not done under any one of the following conditions:

    The file is already packed
    The file has links
    The file is a directory
    The file cannot be opened
    No storage blocks are saved by packing
    A file called **file.z** already exists.
    The **.z** file cannot be created.
    An I/O error occurs during processing

The **unpack** command expands files created by **pack**. For each file specified, **unpack** searches for a file called **file.z**. If this file is a packed file, **unpack** replaces it by its expanded version. The **unpack** command names the new file name by removing the **.z** suffix from **file**. The new file has the same access modes, access and modification dates, and owner as the original packed file.

The exit value is the number of files the **unpack** command was unable to unpack. A file cannot be unpacked if any one of the following occurs:

    The file cannot be opened
    The file is not a packed file
    A file with the unpacked file name already exists

The unpacked file cannot be created

The **unpack** command writes a warning to standard output if the file it is unpacking has links. The new unpacked file has a different inode than the packed file from which it was created. However, any other files linked to the packed file's original inode still exist and are still packed.

The **pcat** command is equivalent to **unpack** except that after unpacking the specified files it writes them to standard output.

Note: Both **pcat** and **unpack** operate only on files ending in **.z**. As a result, when you specify a file name that does not end in **.z**, **pcat** and **unpack** add the suffix and search the directory for a file name with that suffix.

Flags

The following flags apply only ot **pack**.

- Displays statistics about the input **file**s. The statistics are calculated from a Huffman minimum redundancy code tree built on a byte-by-byte basis. Repeating **-** (minus) on the command line toggles this function.

-f Forces packing to occur on files for which there is no space savings.

Examples

1. To compress files:

   ```
   pack  chap1  chap2
   ```

   This compresses **chap1** and **chap2**, replacing them with files named **chap1.z** and **chap2.z**. **pack** displays the percent decrease in size for each file.

2. To display statistics about the amount of compression done:

   ```
   pack  -  chap1  -  chap2
   ```

   This compresses **chap1** and **chap2** and displays statistics about **chap1**, but not about **chap2**. The first **-** (minus) turns on the statistic display, and the second turns it off.

3. To display compressed files:

   ```
   pcat  chap1.z  chap2  |  pg
   ```

   This displays the compressed files **chap1.z** and **chap2.z** on the screen in expanded form, a page at a time (| **pg**). Note that **pcat** added the **.z** to the end of **chap2**, even though we did not enter it.

4. To use a compressed file without expanding the copy stored on disk:

   ```
   pcat  chap1.z  |  grep  'Greece'
   ```

   This pipes the contents of **chap1.z** in its expanded form to the **grep** command. See page 1.1.420.1 for a discussion of piping.

5. To expand compressed files:

```
  unpack  chap1.z  chap2
```

This expands the compressed files **chap1.z** and **chap2.z**, replacing them with files named **chap1** and **chap2**. You can give **unpack** file names either with or without the **.z** suffix.

Related Information

The following command: "cat" in topic 1.1.49.

1.1.310 packf

Purpose

Compresses the contents of a folder into a file.

Syntax

```
         +-----------+   +-------- all ---------------------------------+
packf ---¦           +---¦ +----------- all --------------------------+ +---
         +- +folder -+   +-¦   +------------ sequence ------------+   +-+
                           +---¦  one of                          +---+
                               ¦ +-------+   +------------------+ ¦ ¦
                             ¦ +-¦ num   +---¦     one of       +-+ ¦
                             ¦   ¦ first ¦   ¦¦+--------------+ ¦   ¦
                             ¦   ¦ prev  ¦   +-¦ :num   -prev +-+   ¦
                             ¦   ¦ cur   ¦     ¦ :+num  -cur  ¦     ¦
                             ¦   ¦ .     ¦     ¦ :-num  -.    ¦     ¦
                             ¦   ¦ next  ¦     ¦ -num   -next ¦     ¦
                             ¦   ¦ last  ¦     ¦ -first -last ¦     ¦
                             ¦   +-------+     +--------------+     ¦
                             +--------------------------------------+

    +- -file ./msgbox -+
 ---¦                  +---¦
    +--- -file file ---+

packf --- -help ---¦


-----------------
¦ Do not put a blank between these items.
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **packf** command is used to move messages from a folder and compress those messages into a single file. You can unpack packed messages by using the **inc** command. **packf** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **packf** command takes the specified messages from the specified folder and appends them to the specified file. If the file does not exist, **packf** displays a prompt that asks if you want to create the file. If you want to create the file, **packf** creates the file and places the messages in the file. **packf** separates each message with four **Ctrl-A** characters and a **Newline** character.

Flags

-file *file*	Places the messages at the end of the specified file. If *file* does not exist, **packf** asks you if you want to create the file. If you want to create the file, **packf** creates *file* The default file is **./msgbox**.
+*foldermsgs*	Specifies the messages that you want to pack. *msgs* can be several messages, a range of messages, or a single message. You can use the following message

references when specifying *foldermsgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default is all of the messages in the current folder. If several messages are specified, the first message packed becomes the current message.

-help	Displays help information for the command.

Profile Entries

Current-Folder:	Sets your default current folder.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your **user_mh_directory**.

Files

$HOME/.mh_profile	The MH user profile.

Related Information

See the MH command "inc" in topic 1.1.206.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.311 pagesize

Purpose
Prints system page size.

Syntax

```
pagesize ----¦
```

Description

The **pagesize** command prints the size of a page of memory in bytes, as returned by the **getpagesize** system call. This command is useful in constructing portable shell scripts.

Related Information

See the **getpagesize** system call in the *AIX Operating System Technical Reference*.

1.1.312 passwd, chfn, chsh

Purpose
Changes your login password.

Syntax

```
          +--------+   +--------+
passwd ---¦ one of +---¦        +---¦
          ¦ +----+ ¦   +- user -+
          +-¦ -f +-+
            ¦ -s ¦
            +----+


        +--------+
chfn ---¦        +---¦
        +- user -+

        +--------+
chsh ---¦        +---¦
        +- user -+
```

Description

The **passwd** command establishes or changes the password associated with your login **user** name. When you enter this command, you get a prompt for the old password if one exists. Then you get two successive prompts for the new password. You must enter the same password twice for it to take effect. There will be no prompt for the old password for root.

New passwords can be made up of alphanumeric and non-alphanumeric characters, not all of which are numeric. The password must contain at least six ASCII characters if it is made up of all uppercase or all lowercase letters. It can be as short as five ASCII characters if it has a mixture of uppercase letters, lowercase letters and digits; and as short as four characters if it contains characters which are not alphanumeric.

The **chfn** command is the same as the **passwd** command with the **-f** option, and allows you to change the full name field of the password file entry for **user**.

The **chsh** command is the same as the **passwd** command with the **-s** option and allows you to change the login shell for **user**.

AIX maintains a hashed password data base which is built from **/etc/passwd** and is used by programs such as **login** to speed up username lookup. The commands **passwd, chfn,** and **chsh** update the hash password data base as well as **/etc/passwd**.

If NIS is used in your system, you may also need to update your passwd in that data base. See "yppasswd" in topic 1.1.552.

Note: If there are two users listed in **/etc/passwd** who have the same numeric user ID, these two users are considered to be the same user. Any of these user names may be specified as the **user** parameter to change the password file information associated with that specific user name. If **user** is not specified, **passwd**, **chfn**, and **chsh** will always select the earliest of these equivalent user names regardless of the user name specified when you logged in.

Files

/etc/passwd	ASCII password data base file, contains userids
/etc/passwd.pag	Data base file
/etc/passwd.dir	Data base index
/etc/shells	Contains list of legal shells for **chsh** (**passwd -s**)

Related Information

See the following commands: "yppasswd" in topic 1.1.552, "mkpasswd" in topic 1.1.272.

1.1.313 paste

Purpose
Merges the lines of several files or subsequent lines in one file.

Syntax

```
         +-- -d"\t" ---+
paste ---¦ +---------+ +-- file1 -- file2 --¦
         +-¦ -s      +-+                  ¦
           ¦ -d lis  ¦¦         +-------+
          ¦+---------+¦
          +-----------+

paste -- -s -- file --¦
                     ¦
             +------+
```

Description

The **paste** command reads input *file* (standard input if you specify a **-** as a file name), concatenates the corresponding lines of the given input files, and writes the resulting lines to standard output. Output lines are restricted to 511 characters. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the 511-character limit may be reduced by as much as 50%, depending on the character code set being used.

Without a flag, or with the **-d** flag, the **paste** command treats each file as a column and joins them horizontally with a tab character by default (parallel merging). You can think of this command as the counterpart of the **cut** command (see page 1.1.49), which concatenates files vertically (one file after another).

With the **-s** flag, the **paste** command combines subsequent lines of an input file (serial merging). These lines are joined with the tab character by default.

Note: The action of the **pr -t -m** command is similar to that of the **paste** command, but creates extra blanks, tabs and lines which result in an attractive page layout.

Flags

-d list Changes the delimiter that separates corresponding lines in the output with one or more characters in **list** (the default is a tab). If more than one character is in **list**, the characters are repeated in order until the end of the output. In parallel merging, the lines from the last file always end with a new-line character instead of a character from **list**.

The following special characters can also be used in **list**:

\n	New-line character
\t	Tab
\	Backslash
\0	Empty string (not a null character).
c	An extended character.

You must quote characters that have special meaning to the shell.

-s Merges subsequent lines from the first file horizontally. With this flag, the **paste** command works through one entire file before starting on the next. When it finishes merging the lines in one file, it forces a new line and then merges the lines in the next input file, continuing in the same way through the remaining input files, one at a time. A tab separates the lines unless you use the **-d** flag. Regardless of the **list**, the last character of the file is forced to be a new-line character.

Examples

1. To join several columns of data together:

   ```
   paste  names  places  dates  > npd
   ```

   This command creates a file named **npd** that contains the data from the files **names**, **places**, and **dates**. If **names**, **places**, and **dates** look like this:

names	**places**	**dates**
rachel **elvis** **mark** **marsha** **scott**	**New York** **Austin** **Chicago** **Boca Raton** **Seattle**	**February 5** **March 13** **June 21** **July 16** **November 4**

   then the file **npd** contains:

   ```
   rachel  New York      February 5
   jerry   Austin   March 13
   mark    Chicago  June 21
   marsha  Boca Raton    July 16
   scott   Seattle November 4
   ```

   A tab character separates the name, place, and date on each line. As in this example, the columns do not always line up because the tab stops are set at every eighth column.

2. To separate the columns with a character other than a tab:

   ```
   paste  -d"!@"  names  places  dates  > npd
   ```

   This command alternates **!** and @ as the column separators. If the data in **names**, **places**, and **dates** is the same as in Example 1, the file **npd** contains:

   ```
   rachel!New York@February 5
   jerry!Austin@March 13
   mark!Chicago@June 21
   marsha!Boca Raton@July 16
   scott!Seattle@November 4
   ```

3. To display the standard input in multiple columns:

   ```
   ls  |  paste  -  -  -  -
   ```

   This command lists the current directory in four columns. Each **-** (minus) causes the **paste** command to create a column containing data read from the standard input. The first line is put in the first column, the second line in the second column,..., the fifth line in the first column, and so on.

   This is equivalent to:

   ```
   ls |  paste -d"\t\t\t\n"  -s  -
   ```

   which fills the columns across the page with subsequent lines from the standard input. The **-d"\t\t\t\n"** defines the character to insert after each column: a tab character (**\t**) after the first three columns and a new-line character (**\n**) after the fourth. Without the **-d** flag, the **paste -s -** command would display all of the input as one line with a tab between each column.

Related Information

See the following commands: "grep, egrep, fgrep" in topic 1.1.193, "cut" in topic 1.1.107, and "pr" in topic 1.1.322.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.314 pdisable, phold

Purpose
Disables or reports the availability of login ports.

Syntax

```
   one of
+----------+  +----- -a -----+
¦ pdisable +--¦      -i      +---¦
¦ phold    ¦  +--- device ---+
+----------+                ¦
                +-----------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **pdisable** and **phold** commands each disable a set of login ports. Disabling a port makes the port unavailable to log in. The system disables a port by updating an entry in the **/etc/inittab** file and then sending a signal to **init**. When **init** receives the signal and reads the updated status entry, it takes the appropriate action.

Use the **device** parameter to specify the ports to be disabled. Permitted values for **device** include:

A full device name, such as **/dev/tty1**.
A simple device name, such as **tty1**.
A general class of devices in the form **attribute=value**, which is equivalent to naming each port with a stanza in **/etc/ports** that includes the specified attribute.

If you do not specify a **device** to disable, each command reports the names of currently disabled ports in its set. You must have superuser privileges to execute these commands.

Subtopics

1.1.314.1 pdisable

The **pdisable** command disables the specified port, even if a user is logged on, and makes the port unavailable for log in. To disable the port, the system ends **logger**. If you do not specify any arguments, **pdisable** reports the names of all disabled ports.

1.1.314.2 phold

The **phold** command allows logged-on users to continue, but does not allow any more users to log on. If you do not specify any arguments, **phold** reports the names of all ports on hold.

Flags

-a With **pdisable**, disables all ports that are currently enabled in the **/etc/inittab** file. With **phold**, holds all ports that are currently enabled in the **/etc/inittab** file.

-i Reinitializes **/etc/inittab** to system default values prior to disabling or holding any login ports. You typically use this flag in the **/etc/rc** command file to re-establish default port enabling before starting up the system with multiple users.

Examples

1. To display the names of all ports currently disabled:

   ```
   pdisable
   ```

2. To disable all ports that are enabled in **/etc/inittab**, even if users are logged on:

   ```
   pdisable  -a
   ```

3. To disable the work station attached to the **/dev/tty8** port:

   ```
   pdisable  tty8
   ```

4. To list the ports that are currently on hold:

   ```
   phold
   ```

5. To put all **9600** baud ports on hold:

   ```
   phold  speed=9600
   ```

Files

/etc/ports	Contains descriptions of known normal, shared, and delayed login ports.
/etc/inittab	Contains current status of each known login port.

Related Information

See the following commands: "init, telinit" in topic 1.1.208 and "pstart, penable, pshare, pdelay" in topic 1.1.338.

See the **ports** and **inittab** files in *AIX Operating System Technical Reference*.

1.1.315 pg

Purpose
Formats files to the work station.

Syntax

```
      +---------------+   +-----------------+   +--------+
pg ---¦ +-----------+ +---¦     one of      +---¦        +---¦
      +-¦ -num      +-+   ¦ +-------------+ ¦   +- file -+
        ¦ -c        ¦¦    +-¦ +linenum    +-+          ¦
       ¦¦ -e        ¦¦      ¦ +/pattern/  ¦      +------+
       ¦¦ -f        ¦¦      +-------------+
       ¦¦ -n        ¦¦
       ¦¦ -p string ¦¦
       ¦¦ -s        ¦¦
       ¦+-----------+¦
       +-------------+
```

Description
The **pg** command reads **file**s and writes them to standard output one screen at a time. If you specify **file** as **-** (minus) or run **pg** without arguments, the command reads standard input. Each screen is followed by a prompt. If you press the **Enter** key, the next page is displayed. The **pg** command lets you back up to review something that has already been displayed.

To determine work station attributes, the **pg** command scans the file **terminfo** for the work station type specified by the environment variable **TERM**. The default type is **dumb**. See *AIX Operating System Technical Reference* for information on **terminfo**.

Subcommands

When **pg** pauses and issues its prompt, you can issue a subcommand. Some of these subcommands change the display to a particular place in the file, some search for specific patterns in the text, and others change the environment in which **pg** works.

The following subcommands display a selected place in the file:

+num	Displays the page **num** pages after the current page.
-num	Displays the page **num** pages before the current page.
l	Scrolls the display one line forward.
num l	Displays a screen with the specified line **number** at the top.
+num l	Scrolls the display **lines forward**
-num l	Scrolls the display **num** lines backward.
d	Scrolls half a screen forward. Pressing **Ctrl-D** also does this.
-d	Scrolls half a screen backward. Pressing **-Ctrl-D** also does this.
Ctrl-L	Displays the current page again. A single period also does this.

The following commands search for text patterns in the text. You can use the patterns described in "ed, red" in topic 1.1.147. They must always end with a new line character, even if the **-n** is not used. In an expression such as **[a-z]**, the minus means "through" according to the current collating sequence. A collating sequence may define **equivalence classes** for use in character ranges. See the "Introduction to International Character Support" in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

[**num**]/**pattern**/ Search for the **num**th occurrence of **pattern**. The search begins immediately after the current page and continues to the end of the current file, without wrap around. The default for **num** is 1.

num^pattern^

num^pattern^ Search backward for the **num**th occurrence of **pattern**. The searching begins immediately before the current page and continues to the beginning of the current file, without wrap around. The ^ (circumflex) is useful for the Adds 100 work station, which cannot handle the **?**. The default for **num** is 1.

After searching, **pg** normally displays the line found at the top of the screen. You can change this by adding :**m** or **b** to the search command to leave the line found in the middle or at the bottom of the window with all succeeding subcommands. Use the suffix **t** to return to displaying the line with the pattern to the top of the screen.

You can change the **pg** environment with the following subcommands:

[**num]n**	Begins examining the **num**th next file in the command line. The default **num** is 1.
[**num]w**	Displays another window of text. If **num** is present, sets the window size to **num**.
s file	Saves the input in **file**. Only the current file being examined is saved. This command must always end with a new line character, even if you specify the **-n** flag.
h	Displays an abbreviated summary of available subcommands.
q or Q	Quits **pg**
!AIX-cmd	Sends the specified AIX command to the shell named in the **SHELL** environment variable. If this is not available, the default shell is used. This command must always end with a new line character, even if the **-n** flag is used.

At any time when output is being sent to the work station, you can press the QUIT WITH DUMP (**Ctrl-V**) or INTERRUPT (**Ctrl-C**) key. The **pg** command stops sending output, and displays the shell prompt. Then you can enter any of the subcommands.

Notes:

1. When you use **pg** in a pipe, an Interrupt is likely to end the other commands in the pipe.

2. If work station tabs are not set every eight positions, unpredictable results can occur.

3. When using **pg** in a pipe with other commands that change work station I/O options, work station settings may not be restored correctly.

Searching for Patterns

The following commands search for text patterns in the text. You can use the patterns described under "ed, red" in topic 1.1.147. The commands must always end with a new-line character, even if the **-n** flag is used. In an expression such as **[a-z]**, the minus means "through" according to the current collating sequence. A collating sequence may define equivalence classes for use in character ranges. See the "Introduction to International Character Support" in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

[num]/pattern/	Searches for the **num**th occurrence of **pattern**. The search begins immediately after the current page and continues to the end of the current file, without wrapping around. The default for **num** is 1.
num?pattern?	
num?pattern?	Searches backward for the **num**th occurrence of **pattern**. The searching begins immediately before the current page and continues to the beginning of the current file, without wrapping around. The ^ (circumflex) is useful for the Adds 100 work station, which cannot handle the **?**. The default for **num** is 1.

After completing its search, the **pg** command displays the line containing the pattern at the top of the screen. You can change this by adding an **m** or **b** to the search command; an **m** puts this line in the middle of the screen, and a **b** puts the line at the bottom of the window with all succeeding subcommands. Use the suffix **t** to again display the line with the pattern at the top of the screen.

Changing the pg Command Environment

You can change the **pg** environment with the following subcommands:

[num]n	Begins examining the **num**th next file in the command line. The default **num** is 1.
[num]p	Begins examining the **num**th previous file on the command line. The default **num** is 1.
[num]w	Displays another window of text. If **num** is present, sets the window size to **num**.
s file	Saves the input in **file**. Only the current file being examined is saved. This command must always end with a new line character, even if you specify the **-n** flag.
h	Displays an abbreviated summary of available subcommands.
q or Q	Quits the **pg** command.

!AIX-cmd	Sends the specified AIX command to the shell named in the **SHELL** environment variable. If this shell is not available, the default shell is used. This command must always end with a new-line character, even if the **-n** flag is used.

Warning: Some output is lost when you press the QUIT WITH DUMP (**Ctrl-V**) or INTERRUPT (**Ctrl-C**) key because any characters waiting in the output queue are purged when the QUIT signal is received.

If standard output is not a work station, the **pg** command acts like the **cat** command, except that a header displays before each file.

While waiting for work station input, **pg** stops running when you press INTERRUPT (**Ctrl-C**). Between prompts, these signals interrupt the current task and place you in the prompt mode.

Flags

-c	Moves the cursor to the home position and clears the screen before displaying each page. This flag is ignored if **clear_screen** is not defined for your work station type in the **terminfo** file.
-e	Does not pause at the end of each file.
-f	Does not split lines. Normally, the **pg** command splits lines longer than the width of the screen.
-n	Stops processing when a **pg** command letter is entered. Normally, commands must end with a new-line character.
-p string	Uses **string** as the prompt. If the **string** contains a **%d**, the **%d** is replaced by the current page number in the prompt. The default prompt is **:** (colon). If **string** contains spaces, you must quote it.
-s	Highlights all messages and prompts.
+linenum	Starts at **linenum**.
-num	Specifies the number of lines in the window. On work stations that contain 24 lines, the default is 23.
+/pattern/	Starts at the first line that contains **pattern**.

Files

/usr/lib/terminfo/*	Terminal information files.
/tmp/pg*	Temporary files for the **pg** command.

Related Information
See the following commands: "ed, red" in topic 1.1.147 and "grep, egrep, fgrep" in topic 1.1.193.

See the **terminfo** file in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.316 pick

Purpose
Selects messages by content and creates and modifies sequences.

Syntax

```
        +---------------------------------------------------------------------
        ¦                              one of
pick ---¦                            +---------+
        ¦ +--------+ +-----------+ +-¦ -after  +----- date ----+ +----------
        +-¦        +-¦           +-¦ ¦ -before ¦               +-¦          -
          +- -not -+ +- -lbrace -+ ¦ +---------                ¦ +- -rbrace --
          ¦                        ¦     one of                ¦
          ¦                        ¦ +--------------+          ¦
          ¦                        +-¦ -cc          +- pattern -+
          ¦                          ¦ -date        ¦
          ¦                          ¦ -from        ¦
          ¦                          ¦ -search      ¦
          ¦                          ¦ -subject     ¦
          ¦                          ¦ -to          ¦
          ¦                          ¦ --component  ¦
          ¦                          +--------------+
          ¦                              one of
          ¦                             +------+
          +-----------------------------¦ -and +------------------------------
                                        ¦ -or  ¦
                                        +------+

    +---------------------------------------------------------+  +----------
 ---¦                      +--- -zero ---+ +----------------+ +---¦   one of
    +--- -sequence name ---¦   one of    +-¦     one of     +-+   ¦ +--------
                         ¦ ¦ +---------+ ¦ ¦ +-----------+ ¦     +-¦ -list   -
      +------------------+ +-¦ -zero   +-+ +-¦ -public   +-+       ¦ -nolist ¦
                             ¦ -nozero ¦     ¦ -nopublic ¦         +---------
                             +---------+     +-----------+

pick --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **pick** command is used to select messages and put them into sequences. The **pick** command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **pick** command allows you to select messages that contain particular character patterns or that have particular dates. You can use the **-and**, **-or**, **-not**, **-lbrace**, and **-rbrace** flags to construct compound conditions for selecting messages.

Flags

-after *date* Selects messages with dates later than the specified date. You can use the following date specifications for *date*:

yesterday	**today**	**tomorrow**
sunday	**monday**	**tuesday**
wednesday	**thursday**	**friday**
saturday	**-dd**	**sysdate**

Only messages whose Date: field value is chronologically after the date specified will be considered.

The **pick** command treats the days of the week as days in the past. For example, **monday** means last Monday, not today or next Monday. You can use the **-dd** argument to specify a number of days in the past. For example, **-31** means 31 days ago. For **sysdate** you can specify any valid format defined for your system. See "date" in topic 1.1.110 for more information about date formats.

-and
Forms a logical AND operation between two message selecting flags (for example, **pick -after Sunday -and -from mark**). **-and** has precedence over **-or**, but **-not** has precedence over **-and**. Use the **-lbrace** and **-rbrace** flags to override this precedence.

-before *date*
Selects messages with dates earlier than the specified date. See the **-after** flag for how to specify *date*.

Only messages whose Date: field value is chronologically before the date specified will be considered.

-cc *pattern*
Selects messages that contain the character string *pattern* in the **Cc:** field.

-date *pattern*
Selects messages that contain the character string *pattern* in the **Date:** field.

-datefield *field*
Specifies which dated field is parsed when the **-after** and **-before** flags are given. By default, **pick** uses the **Date:** field.

+*folder msgs*
Specifies the messages that you want to search. The **msgs** parameter can be several messages, a range of messages, or a single message. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default for *msgs*is **all**. The default folder is the current folder. If you specify a folder, that folder becomes the current folder.

-from *pattern*
Selects messages that contain the character string *pattern* in the **From:** field.

-help
Displays help information for the command.

-lbrace	Groups **-and**, **-or**, and **-not** operations. Operations between the **-lbrace** and **-rbrace** flags are evaluated as one operation. You can nest the **-lbrace** and **-rbrace** flags.
-list	Sends a list of selected message numbers to standard output. This allows you to use the **pick** command to generate message numbers to use as input for other commands. For example, **scan `pick -after tuesday -list`** scans all messages in the current folder that were sent after last Tuesday. **-list** is the default if no sequence is specified.
-nolist	Keeps the **pick** command from generating a list of the selected message numbers (see the **-list** flag). If a sequence is specified, **-nolist** is the default.
-nopublic	Restricts the specified sequence to your usage. **-nopublic** does not restrict the messages in the sequence, only the sequence. This option is the default if the folder is write-protected from other users.
-not	Forms a logical NOT operation on a message selecting flag (for example, **pick -not -from george**). This construction evaluates to all messages not chosen by the message selecting flag. **-not** has precedence over **-and**, and **-and** has precedence over **-or**. Use the **-lbrace** and **-rbrace** flags to override this precedence.
-nozero	Appends the selected messages to the specified sequence (see the **-zero** flag).
-or	Forms a logical OR operation on two message selecting flags (for example, **pick -from amy -or -from mark**). **-not** has precedence over **-and**, and **-and** has precedence over **-or**. Use the **-lbrace** and **-rbrace** flags to override this precedence.
-public	Makes the specified sequence available to other users. **-public** does not make protected messages available, only the sequence. This option is the default if the folder is not write-protected from other users.
-rbrace	Groups **-and**, **-or**, and **-not** operations. Operations between the **-lbrace** and **-rbrace** flags are evaluated as one operation. You can nest the **-lbrace** and **-rbrace** flags.
-search *pattern*	Selects messages that contain the character string *pattern* anywhere in the message.
-subject pattern	Selects messages that contain the character string **pattern** in the subject field.
-sequence *name*	Stores the messages selected by the **pick** command in the sequence *name*.

-search pattern	Selects messages that contain the character string **pattern** in the subject field.
-to *pattern*	Selects messages that contain the character string *pattern* in the **To:** field.
-zero	Clears the specified sequence before placing the selected messages into the sequence. This flag is the default (see the **-nozero** flag).
--*component pattern*	Selects messages that contain the character string *pattern* in the heading field *component*(for example, **pick --reply-to amy**).

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.

Files

$HOME/.mh_profile	The MH user profile.

Related Information

See the MH command "mark" in topic 1.1.259.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.317 piobe

Purpose
Writes a file to standard output in a format suitable for a line printer.

Syntax

```
                  +------------+   +-----------+   +------------+
/usr/lpd/piobe ---¦   one of   +---¦   one of   +---¦   one of   +---
                  ¦ +--------+ ¦   ¦ +-------+ ¦   ¦ +--------+ ¦
                  +-¦ -cdp   +-+   +-¦ -ep   +-+   +-¦ -dwp   +-+
                    ¦ -nocdp ¦       ¦ -noep ¦       ¦ -nodwp ¦
                    +--------+       +-------+       +--------+

    +------------+   +--------------+   +--------------------------+
 ---¦   one of   +---¦    one of    +---¦            +------------+ +---
    ¦ +--------+ ¦   ¦ +-----------+ ¦   +- -fw=num --¦   one of   +-+
    +-¦ -dsp   +-+   +-¦ -pq=value +-+               ¦ +--------+ ¦
      ¦ -nodsp ¦       ¦ -wp       ¦                 +-¦ -wll   +-+
      +--------+       +-----------+                   ¦ -trunc ¦
                                                       +--------+

    +- -profile=/etc/ddi/pprinter -+   +----------------+
 ---¦                              +---¦     one of     +---
    +------- -profile=pname -------+   ¦ +------------+ ¦
                                       +-¦ -pitch=num +-+
                                         ¦ -elite     ¦
                                         +------------+

    +-----------------------------+   +------------+
 ---¦ +-------------------------+ +---¦            +---¦
    +-¦ -device=dname -mhp      +-+   +--- file ---+
      ¦ -display      -plot     ¦¦              ¦
     ¦¦ -dpc=value    -strip    ¦¦      +--------+
     ¦¦ -cs=value     -kpoe     ¦¦
     ¦¦ -lm=num       -fl=num   ¦¦
     ¦¦ -fw=num       -tm=num   ¦¦
     ¦¦ -lpi=num      -bm=num   ¦¦
     ¦¦ -start=num    -ph=num   ¦¦
     ¦¦ -fnt=num      -fid=value ¦¦
     ¦¦ -statusfile   -psm      ¦¦
     ¦¦ -pname=dname  -hopsm    ¦¦
     ¦¦ -cdpg=num     -psd=value ¦¦
     ¦+-------------------------+¦
     +---------------------------+
```

Description
The **piobe** command writes **file** to standard output in a form that is suitable for a line printer. If you do not specify a **file** argument, **piobe** reads standard input. The **piobe** command is normally called by the **qdaemon** command after you have enqueued a file with the **print** command (see "print" in topic 1.1.326). The **qdaemon** directs the output from **piobe** to the appropriate device.

Flags

You can specify the following flags on the **print** command line or in the **/etc/qconfig** file (see *AIX Operating System Technical Reference*).

-bm=num	Sets the bottom margin to **num** lines from the top of the page.
-cdp	
-cdpg=num	Sets code page number for IBM 6202 Quietwriter III. The **num** must be either 437 or 850.
-nocdp	Turns the condensed printing mode on (**-cdp**) or off (**-nocdp**).
-cs=value	Uses PC code set **1** or **2**.
-pname=dname	
-device=dname	Specifies the name of a printer stanza in the printer configuration file (see "Files").
-display	Specifies that the input data stream has KSR code page controls.
-dpc=value	Prints in the specified color. The **value** argument can be red, blue, yellow, or black.
-dsp	
-nodsp	Turns on (**-dsp**) or off (**-nodsp**) the double strike mode.
-dwp	
-nodwp	Turns the double-wide printing mode on (**-dwp**) or off (**-nodwp**).
-elite	Sets the character pitch to 12; the same as specifying **-pitch=12**.
-ep	
-noep	Turns on (**-ep**) or off (**-noep**) the emphasized printing mode.
-fid=value	Specifies the font identifier for an IBM 5202 Quietwriter(¦) III Printer font. For imbedded fonts, the **value** argument can be 11 (Courier 10), 85 (Courier 12), 254 (Courier 17), or 159 (Boldface). Values for fonts in the pluggable cartridges precede the font name on the cartridge label.
-fl=num	Sets the form length to **num**.
-fnt=num	Allows font change. Valid values for the **num** argument are 1 through 8.
-fw=num	Sets the right margin at **num** characters from the left edge of the carriage.
-kpoe	Forgives keying mistakes and ignores invalid flags. If you specify this flag, **piobe** processes the job and

	sends you no message. If you do not specify this flag, **piobe** does not forgive invalid flags, does not print the job, and sends you a message detailing the error.
-lm=num	Sets the left margin at **num** characters from the left edge of the carriage.
-lpi=num	Sets the number of lines per inch to **num**. Valid settings are 6 and 8.
-mhp	Allows the horizontal position on the print line to be maintained for line feed and vertical tab controls.
-ph=num	Allows you to use single-sheet paper in the Quietwriter(¦) printer. The printer stops at the end of each page, beeps three times, and waits for you to push the start button. The **num** argument can have the following values: **0** Manual operation. **1** Sheetfeed operation. **2** Continuous operation.
-pitch=num	Sets the character pitch to **num**.
-plot	Specifies that the input data is to be passed without modification, allowing arbitrary files to be printed on arbitrary printers.
-pq=value	Prints in specified print quality. The **value** argument can be dp, text, or letter.
-profile=pname	Specifies the name of a printer configuration file. The default name is **/etc/ddi/pprinter**.
-psd=value	Specifies a paper source drawer for the optional IBM 5202 Quietwriter(¦) III Printer two-drawer sheetfeeder. Valid values are 1 (top drawer), 2 (bottom drawer), and 3 (envelopes).
-psm	Turns on proportional spacing mode.
-nopsm	Turns off proportional spacing mode.
-start=num	Sets the starting page number to **num**.
-statusfile	Updates the status information in the status file that is open on file descriptor 3. The status information is passed from **qdaemon**.
-strip	Strips all multibyte controls from the data stream. The **strip** flag is useful in filter mode to send data that has imbedded printer controls to a non-printer device.
-tm=num	Sets the top margin to **num** lines.

-trunc	Specifies that lines exceeding the value set by the **-fw** flag should be truncated.
-wll	Specifies that lines exceeding the value set with the **-fw** flag should overflow to the next line; reverse of the **-trunc** flag.
-wp	Selects word processing mode; the same as specifying **-pq=letter**.

Files

/etc/ddi/pprinter	Parallel configuration information.
/etc/ddi/sprinter	Serial configuration information.

Related Information

See the following commands: "print" in topic 1.1.326 and "qdaemon, lp" in topic 1.1.347.

See the **qconfig** file in *AIX Operating System Technical Reference*.

1.1.318 plot

Purpose

Plots a graph.

Syntax

```
        +--------+   +----------+   +-------------+   +------------+
plot ---¦ +----+ +---¦          +---¦ +---------+ +---¦            ¦
        +-¦ -a +-+   +- string -+   +-¦ -d      +-+   +--- file ---+
          ¦ -b ¦                      ¦ -       ¦¦               ¦
         ¦¦ -c ¦¦                    ¦¦ -Fxfile ¦¦     +--------+
         ¦+----+¦                    ¦+---------+¦
         +-------+                   +-----------+
```

Description

The **plot** command plots a graph. The input vectors contain the y values of an x-y graph. Values for the x-axis come from the file specified by **-F**. Axis scales are determined from the first vector plotted.

Flags

-a Suppresses the axes.

-b Plots the graph with bold weight lines (medium is the default weight).

-d Does not connect plotted points (this implies **-m**)

-f Does not build a frame around the plot area.

-Fxfile
Uses the specified file for x values; otherwise the positive integers are used. You can specify this flag more than once, causing a different set of x values to be paired with each input vector. If there are more input vectors than sets of x values, the last set applies to the remaining vectors.

-g Suppresses the background grid.

-m Marks the plotted points.

-ri Puts the graph in GPS region **i**, where **i** is between 1 and 25 inclusive (13 by default).

-xf -yf
Positions the graph in the GPS universe with the x-origin (y-origin) at **f**.

-xa -ya
Does not label the x-axis (y-axis).

-xhf -yhf
Specifies the x-axis (y-axis) high boundary.

-xlf -ylf
Specifies the x-axis (y-axis) low boundary.

-xni -yni
Specifies the approximate number of ticks on the x-axis (y-axis).

-xt -yt
Omits the x-axis (y-axis) title.

Examples

1. To plot against the positive integers:

```
plot  plotdata
```

2. To customize x- and y-axes:

```
plot  -r5,y10,xa,Fxfile  yfile
```

   This plots vector "yfile" against vector "xfile", with y-axis ticks beginning at zero, no x-axis labels being printed, and the plot being placed in region 5 of the GPS universe.

3.

```
plot  -hilo -oy  filea fileb  filea  fileb
```

   This plots vectors "filea" and "fileb" against the positive integers, with y-axis ticks going from the lowest to the highest values in the two vectors.

4.

```
plot -Ffilea,Ffileb  filec  filed  filee
```

   This plots vectors "filec" against "filea"; "filed" and "filee" against "fileb". The y-axis scale is determined from "filec"; the x-axis scale from "filea".

1.1.319 portmap

Purpose

Maps program numbers to port numbers to make service calls.

Syntax

```
portmap ---¦
```

Description

The **portmap** program maps RPC program number to the port numbers on RPC servers in order for NFS clients to make RPC service calls. When an RPC server starts, it contacts the **portmap** program to register the RPC programs it is prepared to serve and the port on which it is listening for calls. When NFS clients call an RPC procedure for a given program number, the clients contact the **portmap** program to determine the port number to which they should send the packets.

The **portmap** program is started from **/etc/rc.tcpip**.

Files

/etc/portmap
/etc/rc.tcpip

Related Information

See the following command: "rpcinfo" in topic 1.1.384.

See the section on NFS in *Managing the AIX Operating System*.

1.1.320 post

Purpose
Delivers a message.

Syntax

```
        +- -alias /usr/lib/mh/MailAliases -+   +--- -format ---+
post ---¦                                  +---¦               +---
        +--- -alias file ------------------+   ¦    one of     ¦
                          ¦                    ¦ +-----------+ ¦
          +---------------+                    +-¦ -format   +-+
                                                 ¦ -noformat ¦
                                                 +-----------+

    +-- -nomsgid --+   +--- -nofilter ----+   +- -width 72 --+
 ---¦              +---¦                  +---¦              +---
    ¦    one of    ¦   ¦      one of      ¦   +- -width num -+
    ¦ +----------+ ¦   ¦ +--------------+ ¦
    +-¦ -msgid   +-+   +-¦ -filter file +-+
      ¦ -nomsgid ¦       ¦ -nofilter    ¦
      +----------+       +--------------+

    +-- -noverbose --+   +-- -nowatch --+
 ---¦                +---¦              +--- file ---¦
    ¦     one of     ¦   ¦    one of    ¦           ¦
    ¦ +------------+ ¦   ¦ +----------+ ¦ +--------+
    +-¦ -verbose   +-+   +-¦ -watch   +-+
      ¦ -noverbose ¦       ¦ -nowatch ¦
      +------------+       +----------+

post --- -help ---¦
```

Description

The **post** command is used to route messages to the proper destinations. This command is designed to be called by other programs rather than being called directly by the user. The **post** command is part of the Message Handling (MH) package.

The **post** command searches all components of a message that specify a recipient's address and parses each address to check for proper format. The command puts proper addresses in the standard format and calls the **sendmail** command. The command also performs header operations such as appending the **Date:** and **From:** components and processing the **Bcc:** component.

The **post** command may report errors when parsing complex addresses (for example, **@A:harold@B.UUCP**). If the **sendmail** program is installed on your system and you use complex addresses, use the **spost** command instead of the **post** command.

Flags

-alias *file*	Searches the specified mail alias file for addresses. You can use this flag repetitively to specify multiple mail alias files. The **post** command automatically searches the file **/usr/lib/mh/MailAliases**.

-filter *file*	Uses the header components in the specified file for copies of the message sent to **Bcc:** recipients.
-format	Puts all recipient addresses in a standard format for the delivery transport system. This flag is the default.
-help	Displays help information for the command.
-msgid	Adds a message-identification component (such as **Message-ID:**) to the message.
-nofilter	Removes the **Bcc:** header from the message for the **To:** and **cc:** recipients. Sends the message with minimal headers to the **Bcc:** recipients. This flag is the default.
-noformat	Prevents alteration of the format of the recipient addresses.
-nomsgid	Does not add a message-identification component to the message. This flag is the default.
-noverbose	Does not display information during the delivery of the message to the **sendmail** command. This flag is the default.
-nowatch	Does not display information during delivery by the **sendmail** command. This flag is the default.
-verbose	Displays information during the delivery of the message to the **sendmail** command. This information allows you to monitor the steps involved.
-watch	Displays information during the delivery of the message by the **sendmail** command. This information allows you to monitor the steps involved.
-width *num*	Sets the width of components that contain addresses. The default is 72 columns.

Files

/usr/lib/mh/MailAliases Default mail alias file.
/usr/lib/mh/mtstailor MH tailor file.

Related Information
See the following commands: "ali" in topic 1.1.17, "mhmail" in topic 1.1.264, "send" in topic 1.1.416, "sendmail, mailq, newaliases" in topic 1.1.417, and "whom" in topic 1.1.539.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.321 pprint

Purpose
Prints text files on IBM 3812 Pageprinter.

Syntax

```
         +----------+  +----------+
pprint --¦          +--¦          +--¦
         +-- flag --+  +-- file --+
                   ¦             ¦
          +--------+    +--------+
```

Note: This command does not have MBCS support.

Description

The **pprint** command filters input through **pr** and sends the resulting output to an IBM 3812 Pageprinter. If no files are specified, **pprint** reads the standard input.

The **pprint** command accepts a number of options which control job parameters, page layout, font selection, and input filtering. Some flags accept optional arguments which must be immediately adjacent to the corresponding flag. Keywords (printed in boldface, below) may be abbreviated to any unique prefix (such as **p** for **portrait**).

Flags

-2	Print two logical pages per physical sheet ("two-up").
-5152	Print file using emulation of PC graphics printer. Flags **2**, **c**, **f**, **l**, **L**, **O**, **R**, **w**, and **W** are ignored.
-c num	Prints **num** copies of each page of the document. Copies are not collated.
-e[where][,level]	Report errors to **where** which may be one of **mail** (default), **message**, or **trailer**. **Level** selects the severity threshold for error reports. **Level** may be one of **none** (default), **error** (errors only), **warning** (warnings and errors), or **all** (warnings, errors and informational messages). (Fatal errors and internal errors are always reported.)
-f font	Print the document in font **font**. The available fonts and code page tables are listed in the *IBM 3812 Pageprinter Programming Reference*. Give the font name and size, and code page table if different from **fcp**. For example: `pprint -fPRESTIGE.9 filea` `pprint -fCourier.B10.acp filea` will print **filea** using the fonts PRESTIGE 9 (with the code page **fcp**) and Courier Bold 10 (with code page **acp**) respectively.
-h header	User **header** as the page header for **pr**.

-in	Indent document by prepending **n** blanks to each line.
-J	Suppress printing of the job header page.
-ln	Adjust interline spacing to print **n** blanks to each line.
-L[orientation]	Set page orientation (default **portrait**). **Orientation** may be one of **portrait**, **inverted**, **left**, **right**, or **landscape;** the default is **right**. Landscape orientations (**left** and **right**) default to a page width of 152 columns.
-mmessage	Print **message** on the job header page.
-n[filter]	Override the default input filter **pr**. The default for **filter** is **cat**.
-O[width]	Outline pages. Optional **width** selects width of border in pels (default 3).
-Pprinter	Print this file on **printer** (default **pp**).
-parg	Pass **arg** to the input filter. This may be a single parameter (for example, **-p-h**) or a string (such as **-p "-h -f -m"**).
-R[.n]	Print page rules every **n** lines (default 2).
-tfilter	Override the default output filter **lpr** The default for **filter** is **cat**.
-v	Send file containing Page Map Primitive (PMP) commands to output filter (**lpr**). Flags **2, c, f, i, l, L, O, R, w,** and **W** are ignored. PMP is described in the *IBM 3812 Pageprinter Programming Reference*.
-wn	Set the line width to **n** columns. A line width of more that 80 columns defaults to right landscape orientation.
-W	Wrap long lines. Multiple occurrences of **-W** toggle between **wrap** and **truncate**.

Files

/usr/lib/font/dev3812/fonts/* .dat	Font descriptions.
/usr/lib/font/dev3812/fonts*.*cp	Font code page descriptions.
/usr/adm/lpd-errs	Error message file (site dependent).

Related Information

See the following commands: "cat" in topic 1.1.49, "pr" in topic 1.1.322, and "lpr" in topic 1.1.248.

See *IBM 3812 Pageprinter Programming Reference*, S544-3268.

1.1.322 pr

Purpose
Writes a file to standard output.

Syntax

```
      +- -l66 -o0 +1 -+   +-- -1 ---------------------------------+   +--------
pr ---¦  +---------+  +---¦ +- -m ------------+   +-----------+ +---¦       or
      +--¦ -d      +--+   +-¦           +------+ +---¦         +-+   ¦ +------
         ¦ -        ¦¦      +- -num --¦      +-+   +- -s char -+    +-¦ -t
        ¦¦ -l num   ¦¦                +- -a -+                         ¦ -h "s
        ¦¦ -o num   ¦¦                                                 +------
        ¦¦ -p       ¦¦
        ¦¦ -r       ¦¦
        ¦¦ -w num   ¦¦
        ¦¦ + num    ¦¦
        ¦+---------+¦
        +-----------+

    +-------------------------------+   +----------------------------------+   +
 ---¦       +--------+   +-- 5 --+ +---¦ one of                           +---¦
    +- -n --¦        +---¦       +-+   ¦ +----+   +--------+   +-- 8 --+ ¦   +
            +- char -+   +- num -+     +-¦ -e +---¦¦       +---¦¦      +-+
                                         ¦ -i ¦   +- char -+   +- num -+¦
                                        ¦+----+                         ¦
                                        +-------------------------------+
```

```
-----------------
¦ Do not put a blank between these items.
```

Description
The **pr** command writes **file** to the standard output. If you do not specify **file** or if **file** is a - (minus), **pr** reads standard input. A heading that contains the page number, date, time, and the name of the file separates the output into pages.

Unless specified, columns are of equal width and separated by at least one space. Lines that are too long for the page width are cut off. If the standard output is a work station, **pr** does not display any error messages until it has ended.

Flags

-a	Displays multi-column output across the page.
-d	Double-spaces the output.
-e[char][num]	Expands tabs to character positions **num**+1, 2***num**+1, 3***num**+1, and so on. The default value of **num** is 8. Tab characters in the input expand to the appropriate number of spaces to line up with the next tab setting. If you specify **char** (any character other than a digit) that character becomes the input tab character. The default value of **char** is the ASCII TAB character.
-f	Uses a form-feed character to advance to a new page.

(Otherwise **pr** issues a sequence of line-feed characters.) Pauses before beginning the first page if the standard output is a work station.

-h "string"	Displays **string** as the page header instead of the file name. The flag and string should be separated by a blank.
-i[char][num]	In the **output**, replaces white space wherever possible by inserting tabs to character positions **num**+1, 2***num**+1, 3***num**+1, and so on. The default value of **num** is 8. If you specify **char** (any character other than a digit), that character becomes the output tab character. (The default value of **char** is the ASCII TAB character.)
-lnum	Sets the length of a page to **num** lines (the default is 66).
-m	Combines and writes all files at the same time, with each file in a separate column. (This overrides the -**num** and -**a** flags).
-n[char][num]	Provides **num**-digit line numbering (the default value of **num** is 5). The number occupies the first **num**+1 character positions of each column of normal output or each line of **-m** output. If you specify **char** (any character other than a digit), that character is added to the line number to separate it from whatever follows (the default value of **char** is an ASCII TAB character).
-onum	Indents each line by **num** character positions (the default is 0). The number of character positions per line is the sum of the width and offset.
-p	Pauses before beginning each page if the output is directed to a work station. (**pr** sounds the alarm at the workstation and waits for you to press the **Enter** key.)
-r	Does not display diagnostic messages if the system cannot open files.
-schar	Separates columns by the single character **char** instead of by the appropriate number of spaces (the default for **char** is an ASCII TAB character).
-t	Does not display the five-line identifying header and the five-line footer. Stops after the last line of each file without spacing to the end of the page.
-wnum	Sets the width of a line to **num** character positions (the default value is 72 for equal-width multi-column output, no limit otherwise).
-num	Produces **num**-column output (the default is 1). The **-e** and **-i** flags are assumed for multi-column output.
+num	Begins the display with page **num** (the default value

is 1).

Examples

1. To print a file with headings and page numbers on the printer:

   ```
   pr  prog.c  |  print
   ```

   This adds page headings to **prog.c** and sends it to the **print** command. The heading consists of the date the file was last modified, the file name, and the page number.

2. To specify a title:

   ```
   pr  -h  "MAIN  PROGRAM"  prog.c  |  print
   ```

   This prints **prog.c** with the title **MAIN PROGRAM** in place of the file name. The modification date and page number are still printed.

3. To print a file in multiple columns:

   ```
   pr  -3  word.lst  |  print
   ```

   This prints the file **word.lst** in three vertical columns.

4. To print several files side-by-side on the paper:

   ```
   pr  -m  -h  "Members  and  Visitors"  member.lst  visitor.lst  |  print
   ```

   This prints **member.lst** and **visitor.lst** side by side with the title **Members and Visitors**.

5. To modify a file for later use:

   ```
   pr  -t  -e  prog.c  > prog.notab.c
   ```

   This replaces tab characters in **prog.c** with blanks and puts the result in **prog.notab.c**. Tab positions are at columns 9, 17, 25, 33, ... The **-e** tells **pr** to replace the tab characters; the **-t** suppresses the page headings.

Related Information

See the following command: "cat" in topic 1.1.49.

1.1.323 prev

Purpose
Displays the previous message.

Syntax

```
        +------------+   +--- -header ---+   +------------------------+
prev ---¦            +---¦    one of     +---¦         one of         +---¦
        +- + folder -+   ¦ +-----------+ ¦   ¦ +--------------------+ ¦
                         +-¦ -header   +-+   +-¦ -showproc cmdstring +-+
                           ¦ -noheader ¦       ¦ -noshowproc         ¦
                           +-----------+       +--------------------+


prev -- -help --¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **prev** command is used to display the previous message in a folder. This command is equivalent to the **show** command with **prev** specified as the message. The **prev** command is part of the Message Handling (MH) package and can be used with other MH commands.

The **prev** command links to the **show** program and passes this program its flags and attributes.

Note: If you link to the **prev** command and call that link something other than **prev**, your link functions like the **show** command rather than the **prev** command.

The **show** command invokes a program to perform the listing. The system default is **/bin/pg**. You can define your own default with the **showproc:** entry in **$HOME/.mh_profile**. If you set **showproc:** entry to **mhl**, the **show** command calls an internal **mhl** routine instead of the **mhl** command. You can also specify the program to perform a listing in the **cmdstring** of the **-showproc** flag.

The **show** command passes any flags that it does not recognize to the program performing the listing. Thus, you can specify flags for the listing program, as well as the flags described in this command section.

Flags

+ *folder*	Specifies the folder that contains the message you want to show.
-header	Displays a one-line description of the message being shown. The description includes the folder name and the message number.
-help	Displays help information for the command.
-noheader	Does not display a one-line description of each message being shown.

-noshowproc	Uses **/bin/cat** to perform the listing.
-showproc *cmdstring*	Uses the specified command string to perform the listing.

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.
showproc:	Specifies the program used to show messages.

Files

.profile	
$HOME/.mh_profile	MH user profile.

Related Information

See the following commands: "next" in topic 1.1.293, "show" in topic 1.1.423.

See the **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.324 prf1

Purpose

IBM 4201 Proprinter/IBM 5152 Graphics Printer **nroff** post-processing filter.

Syntax

```
        +------+
prf1 ---¦      +---¦
        +- -g -+
```

Note: This command does not have MBCS support.

Description

The **prf1** command is a post-processing filter to convert the intermediate output of **nroff** and **colpro** to the IBM 4201 Proprinter control sequences and characters. In particular, the filter does the following:

1. Converts strings of the form <bs><char>_<bs><char> to <start underlining> <char><char><stop underlining>.

2. Converts sequences of the form <Esc>Ax to IBM 4201 Proprinter nonprinting control commands.

3. Converts sequences of the form <Esc>[B,C]? to a set of sequences to download special characters. If the **-g** option is specified, these characters are printed in graphics mode.

4. Convert sequences of the form <Esc>DX to x+130 (**nroff** cannot send codes over 127).

5. Convert sequences of the form <ESC>EX to print the code x using the all-characters font.

6. Convert sequences of the form <Esc>Fx to print the code x+130 using the all-characters font.

7. Convert <Esc>9 to <1/2 space linefeed>.

The **prf1** command is provided primarily for use by **proff**.

Flag

-g	Option uses only those escape sequences understood by IBM 5152 Graphics Printer.

Related Information

See the following commands: "colpro" in topic 1.1.79, "nroff, troff" in topic 1.1.301, and "proff" in topic 1.1.333.

See *IBM 4201 Proprinter Guide to Operations*, PN6328945.

See *IBM Personal Computer Hardware Reference Library Guide to Operations*, 1502490.

1.1.325 primrec

Purpose
A user-level replication reconciliation procedure.

Syntax

```
                  +------+
/etc/primrec ----¦+----++---- gfs --- targetnumber --- hostnumber ---¦
                  +¦ -f ++
                   ¦ -v ¦
                   +----+
```

Description

primrec is the utility that performs the user-level propagation of replicated file systems. **gfs** is the global file system number of the file system to be propagated. Since pack numbers and site numbers are by convention identical, **targetnumber** is is the pack number of the target pack. For the same reason, **hostnumber** identifies the site which will be the source of the updated versions. The host site must have either a primary or backbone copy of the replicated file system or **primrec** will fail. In general this will be the current synchronization site (CSS) of the file system in the current partition. **primrec** uses **/etc/comlist** to determine what files need propagation to bring the local system up to date. Because **primrec** issues **spropin** system calls to cause the actual propagation, **primrec** MUST be executed on the target site.

This is not intended to be used by anyone other than the system administrator.

Flag

-f	Compare all files in file system. Used when the non-primary's low-water mark has questionable accuracy.
-v	Verbose option. The system displays information on the propagation status of each file.

If the **-v** option is not provided, a series of dots are written to **stdout** instead. Each dot corresponds to 20 files being considered, and each line corresponds to 1000 files (50 dots).

With or without the **-v** option, errors that occur while propagating a file (for example, out-of-space or site-down conditions) cause an error message to be written to **stderr**, and no message or dot is written to **stdout**.

Related Information

See the following commands: "recmstr" in topic 1.1.364 and "comlist" in topic 1.1.82.

See the **spropin** and **chlwm** system calls in *AIX Technical Reference*.

1.1.326 print

Purpose
Enqueues a file.

Syntax

```
         +------------------------------+   +---- -bp=2 ----+
print ---¦                +---- 0 ----+ +---¦    one of     +---
         +- queueargname -¦¦          +-+   ¦ +-----------+ ¦
                          +- :device -+     +-¦ -bp -bp=0 +-+
                                              ¦ -nb -bp=1 ¦
                                              ¦     -bp=2 ¦
                                              +-----------+

    +------------------------------------+   +------------+
 ---¦ +--------------------------------+ +---¦         ¦   +---¦
    +-¦ -cp         -pr=num  -rm       +-+   ¦ +- -fi --+ ¦
      ¦ -from=name² -of=file -tl=title ¦¦    +-¦        +-+
     ¦¦ -nc=num     -ot=text -to=name  ¦¦      +- file -+
     ¦¦ -no         -q                 ¦¦     +----------+
     ¦+--------------------------------+¦
     +----------------------------------+

         +------------------------------+   +-------------+
print ---¦                +------------+ +---¦   one of    +--- file ---¦
         +- queueargname -¦           ²+-+   ¦ +---------+ ¦            ¦
                          +- -su=user -+     +-¦ -ca     +-+ +--------+
                                               ¦ -ap=num ¦
                                               +---------+

         +- -dg -+
print ---¦      ²+---¦
         +- -rr -+
                                            one of
         +------------------------------+   +-----+²
print ---¦                +---- 0 ----+ +---¦ -dd +---¦
         +- queueargname -¦¦          +-+   ¦ -dk ¦
                          +- :device -+     ¦ -du ¦
                                            +-----+
```

¦ Do not put a blank between these items.
² Only members of the system group can use these flags.
¦ Only specify **-fi** once.

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **print** command is a general purpose utility for enqueing requests to a shared resource, typically a printer device. With the **print** command you can enqueue print requests, cancel print requests, alter the priority of a print request, display the status of print queues and devices, enqueue backup requests, enqueue restore requests, and display the table of contents of a backup. For information on how to establish default print, backup, and restore queues, see *Managing the AIX Operating System*.

To enqueue, cancel, or reprioritize files on a specific queue, specify its corresponding **queueargname**. Each printer queue on the system has a stanza entry in the **/etc/qconfig** file containing various parameters. One of these is the **argname** parameter. The print command compares the **queueargname** which you specify with a matching **argname** parameter found in the **/etc/qconfig** file in order to determine which queue to send your print request to. Entering the **print** command with the **-q** flag will show the status of the various printer devices and queues, along with each queue's corresponding **argname**. If more than one device services a queue, you can also request a particular device by specifying **queueargname:device**. If you do not specify a **queueargname** at all, the job will be sent to the default printer queue. If you do not specify a file, print will copy standard input into a file and enqueue it for printing. For further information, see **qconfig** in the *AIX Operating System Technical Reference*.

Print requests may have operator messages associated with them. The messages can be used to tell another user who is operating the printer to load a special form into the printer before allowing this job to print. These messages are specified with the **-of** and **-ot** flags. When **qdaemon**, the daemon that processes print requests, is ready to begin a request that has an associated message, the system displays the message on the console of the machine where **qdaemon** is running. The text of the message is accompanied by a prompt that tells the printer operator how to signal the request to continue or cancel the request.

Notes:

1. Before you can print, backup, or restore a file, you must have read access to it. To remove a file, you must also have write access to the directory that contains the file.

2. If you want to continue changing the **file** after you issue the **print** command but before it is printed, you must use the **-cp** flag.

3. When enqueing files on a printer, flags and file names can be interspersed in any order.

4. Blanks between flags and their arguments are not permitted.

5. If the printer must receive data in a specific format (for example, postscript), you must provide a printing backend to process files and send them to the printer.

6. Japanese character output is supported by IBM 5575, 5227, and 5327 printers attached to PS/2 or PS/55 model 5570 computers.

Flags

If you give **print** a list of **file** names, it enqueues them all for printing on the default printer.

-ap=num Changes to **num** the priority of the named file. The file must have been submitted for printing prior to entering the **print** command with this flag. See "-pr" on page 1.1.326 for a description of priorities.

-bp=num

-bp

-nb	Controls the printing of burst pages according to the value of **num** as follows: **0** Does not print headers or trailers. **1** Prints one header page before each **file**. No trailer appears. **2** Prints a header page at the beginning and a trailer page at the end of each **file**. The **header** stanza in the **qconfig** file defines the default treatment of burst pages. Specifying only **-bp** is the same as specifying **-bp=2**. Specifying **-nb** is the same as specifying **-bp=0**.
-ca	Cancels the printing of the named **files**.
-cp	Copies the **file**. Ordinarily, to save disk space, **print** remembers the name of the file, but does not actually copy the file itself. Use the **-cp** flag if you want to continue changing the file while you are waiting for the current copy to be printed.
-fi	Causes **print** to act as a filter. The **print** command automatically reads standard input if you do not specify **file**s as arguments. However, if you do specify **file** arguments, you can use the **-fi** flag to force **print** to read standard input at the appropriate time.
-nc=num	Prints **num** copies of the file. Normally a file is printed only once.
-no	Notifies you when your job is finished. If the **-to** flag is also used, **print** notifies the user for whom the request is intended (see the **-to** flag on page 1.1.326).
-of=file	Submits an operator message with a print request. The specified **file** contains the text of the message.
-ot=text	Submits an operator message with a print request. The specified **text** contains the text of the message.
-pr=num	Sets the priority of the named **file** to **num**. Higher numbers assign higher priority. The default priority is 15. The maximum priority is 20 for most users and 30 for the users with superuser authority and members of the system group (group 0).
-q	Displays the status of the printers and queues, along with each queue's corresponding **argname**. The environment variable **LC_TIME** controls the appearance of the **time** field.
-rm	Removes the file after it has been successfully printed.
-tl=title	Puts **title** on the header page and displays it when the **-q** flag is specified. Normally the job title is the name of the file. If **print** reads from standard input, the job title is STDIN.# where # is the process ID of

the **print** command.

-to=name Labels the output for delivery to **name**. Normally the output is labeled for delivery to the person issuing the print request.

In addition to the previous flags that are available to all users, the **print** command accepts the following flags when they are entered by users that have superuser authority or users that are members of the system group:

-dd Turns off the device associated with **queue**. The **qdaemon** no longer sends jobs to the device, and entering **print -q** shows its status as **OFF**. Any job currently running on the device is allowed to finish.

-dg Kills the **qdaemon** after all currently running jobs are finished. Use of this flag is the only clean way to bring the **qdaemon** down. Use of the **kill** command may cause problems, such as jobs hanging up in the queue.

-dk Acts the same as **-dd**, except current jobs are killed. They remain in the queue, and are run again when the device is turned on.

-du Turns on the device associated with **queue**. The **qdaemon** sends jobs to it again and entering **print -q** shows its status as **READY**.

Note: If more than one device is associated with a queue, you must specify the device as well as the queue when you use the **-dd**, **-dk**, and **-du** flags. Devices are numbered, starting at zero, in the order that they appear in the **qconfig** file. For example, **-lp:0** designates the first device on the **lp** queue. **-lp** designates the same device only if there is no other device on that queue.

-from=name Labels the output as though **name** had submitted it. You can only use this flag with superuser authority.

-rr Forces the **qdaemon** to reread the **qconfig** file after all currently running jobs are finished. With this flag, a user with superuser authority can change the **qconfig** file without having to kill and restart the **qdaemon**.

-su=user Cancels or changes the priority on another **user**'s job when used with the **-ca** or the **-ap** flags. For example, a job **report** submitted by user **ann** can be cancelled as follows:

```
print -su=ann -ca report
```

The **print** command passes flags it does not recognize to the backend that does the printing. Thus, for each queue there are flags not described above that can be included on the **print** command line. See "piobe" in topic 1.1.317 for a list of these flags.

Examples

1. To print a file on the default printer:

   ```
   print  memo
   ```

2. To print a file with page numbers:

   ```
   pr  prog.c  ¦  print
   ```

   The **pr** command puts a heading at the top of each page that includes the date the file was last modified, the name of the file, and the page number. The **print** command then prints the file.

3. To see if a file is still waiting to be printed:

   ```
   print  -q
   ```

   This command displays the status of the queues and printers. If a file has not been printed yet, it appears in the queue status listing. If you piped data to **print**, as in Example 2, it is listed as **PRIMARY.OUTPUT**.

4. To stop printing a file:

   ```
   print  -ca  chapter1
   ```

   This command cancels the request you made earlier to print the file **chapter1**. If the file is currently being printed, the printer stops immediately. If the file has not been printed yet, it is removed from the queue so that it will not be printed. If the file is not in the queue, **print** displays the message:

   ```
   no such request from you -- perhaps it's done?
   ```

5. To disconnect a printer from the queueing system:

   ```
   print  -a:2  -dd
   ```

   This command stops print requests from being sent to the third printer that serves the **-a** queue. If a file is currently being printed, it is allowed to finish. You must be a member of the **system** group (group **0**) to run this command.

   Note: The printers serving a given queue are numbered starting with zero in the order that they appear in the **/etc/qconfig** file.

Files

/etc/qdaemon	Queueing daemon.
/usr/lpd/qdir/*	Queue requests.
/usr/lpd/stat/*	Information on the status of the devices.
/usr/spool/qdaemon/*	Temporary copies of enqueued files.
/etc/qconfig	Queue configuration file.

Related Information

See the following commands: "backup" in topic 1.1.32, "piobe" in topic 1.1.317, "pr" in topic 1.1.322, "qdaemon, lp" in topic 1.1.347, and "restore" in topic 1.1.371.

See the **qconfig** file in *AIX Operating System Technical Reference*.

See the discussion of **print** in *Managing the AIX Operating System*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.327 printlocal

Purpose
Displays the <LOCAL> alias of the current process.

Syntax

```
printlocal ---¦
```

Description

The **printlocal** command displays the current setting of your <LOCAL> directory alias. Your <LOCAL> directory alias controls which file you access when you use a symbolic link beginning with "<LOCAL>" to access the file.

If your AIX system is not a member of a Transparent Computing Facility cluster, your <LOCAL> alias will always refer to the <LOCAL> directory of your system. If your AIX system is part of a TCF cluster, your <LOCAL> alias will usually refer to the <LOCAL> directory of your system; but on occasion it may be set to refer to the <LOCAL> directory of another cluster site.

By convention, the <LOCAL> directory is usually named **/site_name**. To determine the actual <LOCAL> directory of your system, use the **sitelocal** command.

By using the <LOCAL> alias inside symbolic links, AIX allows users on different TCF cluster sites to access different files and directories using the same pathname. For instance, each site requires its own devices (**/site_name/dev**) and temporary file space (**/site_name/tmp**) and AIX provides the symbolic links

```
    /dev  ->  <LOCAL>/dev
```

and

```
    /tmp  ->  <LOCAL>/tmp
```

so that users can use the shorthand names such as **/dev/tty0** and **/tmp/myfile** to refer to **/site_name/dev/tty0** and **/site_name/tmp/myfile**, respectively. Users on other cluster sites must use the full path names to access devices and temporary files on your system, just as you would have to use the full path names to access devices and temporary files on another site in the cluster.

When you execute a command on another cluster site using the **onsite** command, **onsite** changes your <LOCAL> alias to refer to the new site's <LOCAL> directory, just as though you had logged on to that site directly. See the **onsite** command man page for more information on remote command execution in TCF.

When a program migrates, however, from one site in the cluster to another, the <LOCAL> alias is not changed. Because of this, a program which is running can continue to access temporary files and terminal devices after it is migrated using the same path names it used before being migrated. If the **printlocal** command is run from a program which has been migrated, you will see that the <LOCAL> directory alias has been unchanged. See the **migrate** built-in commands on the **sh** and **csh** man pages for more information on process migration in TCF.

Note: If you migrate an interactive **csh** or **sh** shell, the shell will notice that it is being migrated and will set its <LOCAL> alias to be the <LOCAL> directory for the site to which it is migrated. Only non-interactive shell scripts are allowed to continue running with the <LOCAL> alias different from the site's default <LOCAL> directory.

Related Information

See the following commands: "csh" in topic 1.1.100, "onsite, on" in topic 1.1.306, and "sh, Rsh" in topic 1.1.420.

See **getlocal** in the *AIX Operating System Technical Reference*.

1.1.328 printspath

Purpose

Displays the site path of the current process.

Syntax

printspath ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **printspath** command displays your current site path. Your site path controls where in a TCF cluster AIX will be running your commands. You can change your site path using the **setspath** built-in command of your shell.

When you ask AIX to run a command, AIX first selects which command to run and then picks which TCF cluster site should be used to run your program. Both the selection of the command and the selection of the TCF cluster site use your site path.

If the command is installed inside a hidden directory, the site path along with your experimental version prefix (see **setxvers**) controls which element within the hidden directory is selected.

Then, your site path is consulted again to select the machine on which to run your command.

Your site path is made up of the following types of entries:

LOCAL	indicates the local site. This entry is often the first item in your site path so that commands that can run on the local system are run there.
cpu_type	indicates the type of cpu on which you would want your command run. The possible values for **cpu_type** are **i386**, **i370** and **xa370**.
site_name	name of a site in your TCF cluster.

For example, suppose you are logged onto an AIX PS/2 system **pooh** and have the following site path:

```
  LOCAL
  i386
  tigger
  i370
```

where **tigger** is the name of an AIX/370 system in your cluster. If you then run the command **cat**, AIX will run this command on your local system **pooh** because it finds **LOCAL** first on your site path and finds **/bin/cat@/i386**, a program that can run on your local system.

Suppose you next run the command **uvcp**, for which there is only an AIX/370 version of the program (**/usr/bin/uvcp@/i370**). AIX will see that it cannot run this on the **LOCAL** system, and will then look at other elements in our site path. This command cannot be run on any AIX PS/2 system, so the **i386** element is skipped. The command will run on **tigger**, so that is where AIX runs your command.

If the site **tigger** was down when you run the **uvcp** command, however, AIX would have next considered the **i370** element in your site path. This would instruct AIX to select any of the remaining AIX/370 systems. This selection does not necessarily select the least loaded AIX/370 system. To do this, you would need to use the **fast** command (see "fast, fastsite" in topic 1.1.161).

You can override your site path and select a particular site on which to run your command by using the **onsite** command (see the "onsite" command in this book on page 1.1.306).

Related Information

"onsite" on page 1.1.306, "fast, fastsite" on page 1.1.161, "printxvers" on page 1.1.329, and the "setspath" sections in "csh", page 1.1.100.5, and "sh", page 1.1.420.23.

"**getspath, setspath**", and "**getxvers, setxvers**" in the *AIX Operating System Technical Reference*.

1.1.329 printxvers

Purpose

Displays the experimental-version prefix of the current process.

Syntax

printxvers ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **printxvers** command displays the experimental-version prefix, if any, of the current process. This prefix is set by the **setxvers** command which is a built-in subcommand of both **sh** and **csh**.

The experimental-version prefix is used when searching hidden directories for executable files. A hidden directory component whose name begins with the prefix is selected for execution in preference to a component whose name does not begin with this prefix.

The experimental-version mechanism allows for two or more versions of a command (for the same machine type) to be installed in a single hidden directory. Different users will run different versions of the same command depending on how they set their experimental-version prefix. This allows a few users to test out a new version of a command before making the new version official.

For example, if the hidden directory **/bin/who@** contained the two components i386 and newi386, and you have issued the command:

```
  setxvers new
```

then when you run the **who** command, you will be running the new version of the command, rather than the standard, i386, version of the command. In this case, **printxvers** would display **new**.

Related Information

See the following: "sh, Rsh" in topic 1.1.420, "csh" in topic 1.1.100, and "printspath" in topic 1.1.328.

See the discussion of Hidden Directories in Chapter 1 of the *AIX Operating System Technical Reference*.

1.1.330 probe

Purpose

Determines if a site is up.

Syntax

```
         +- site_number -+
probe ---¦               +---¦
         +---site_name --+
```

Description

The **probe** command attempts to send a message to the site specified by **site_name** or **site_number** to determine if that site is operational and functioning in the **TCF** cluster.

Related Information

See the following command: "prober" in topic 1.1.331 and "clusterstart, clusterstop" in topic 1.1.75.

See in the *TCP/IP Users Guide*, **ifconfig** and **ping**.

1.1.331 prober

Purpose
Asynchronous probe daemon.

Syntax

```
          +------------+
prober ---¦            +---¦
          +- interval -+
```

Description

At regular intervals, the **prober** command issues a **probe** to sites believed to be up. The result is to determine whether any of the sites believed to be in the current partition have gone down. Generally normal activity to a site would detect the site failure but under some conditions a site might not be actively involved and therefore artificial traffic is necessary to detect a failure. The **interval** is the optional argument (in seconds) and will default to 60 if not specified.

The **prober** command disconnects itself from the invoking shell by doing an internal **fork** before starting its main activity.

This is not intended to be used by anyone other than the system administrator.

Related Information

See the following command: "probe" in topic 1.1.330.

1.1.332 prof

Purpose
Displays program profile data.

Syntax

```
        +--- -t ----+   +--------+   +-----------+   +- a.out -+   +- -m mon.o
prof ---¦  one of   +---¦ one of +---¦ +-------+ +---¦         +---¦
        ¦ +-------+ ¦   ¦ +----+ ¦   +-¦ -g -s +-+   +- prog --+   +-- -m dat
        +-¦ -a -n +-+   +-¦ -o +-+     ¦ -h -z ¦
          ¦ -c -t ¦       ¦ -x ¦      ¦¦ -S    ¦¦
          ¦ -l    ¦       +----+      ¦¦ -v    ¦¦
          +-------+                   ¦+-------+¦
                                      +---------+
```

Description
The **prof** command interprets profile data collected by the **monitor** subroutine for the object file **prog** (**a.out** by default). It reads the symbol table in the object file **prog** and correlates it with the profile file (**mon.out** by default). **prof** displays, for each external text symbol, the percentage of execution time spent between the address of that symbol and the address of the next, the number of times that function was called, and the average number of milliseconds per call.

For the number of calls to a function to be tallied, you must have compiled the file using the **-p** flag of the **cc** command. This flag also arranges for the object file to include a special profiling startup function that calls the **monitor** subroutine at the beginning and end of execution. It is the call to **monitor** at the end of execution that writes **mon.out**. Thus, only programs that explicitly **exit** or **return** from **main** cause the **mon.out** file to be produced.

Note: No more than 600 functions can have call counters established during program execution. If you exceed this limit, **prof** overwrites other data and damages the **mon.out** file. **prof** automatically reports the number of call counters used whenever the number exceeds 500.

Flags

The mutually exclusive flags **a**, **c**, **n**, and **t** determine how **prof** sorts the output lines:

-a or -l Sorts by increasing symbol address.

-c Sorts by decreasing number of calls.

-n Sorts lexically by symbol name.

-t Sorts by decreasing percentage of total time (default).

The mutually exclusive flags **o** and **x** specify how to display the address of each symbol monitored.

-o Displays each address in octal, along with the symbol name.

-x Displays each address in hexadecimal, along with the symbol name.

Use the following flags in any combination:

-g	Includes non-global symbols (static functions).
-h	Suppresses the heading normally displayed on the report. (This is useful if the report is to be processed further.)
-m mdata	Takes profiling data from **mdata** instead of **mon.out**. The -m flag is optional if prog is specified.
-s	Produces a summary file in mon.sum. This is only useful when more than one profile is specified.
-S	Displays a summary of monitoring parameters and statistics on standard error.
-z	Includes all symbols in the profile range, even if associated with zero calls and zero time.
-v	Suppresses all printing and produces a graphic version of the profile on the standard output for display by the plot filters. When plotting, the numbers low and high by default 0 and 100, may be given to cause a selected percentage of the profile to be plotted with accordingly higher resolution.

Files

mon.out	Default profile.
a.out	Default object file.

Related Information

See the following commands: "cc" in topic 1.1.52 and "nm" in topic 1.1.298.

See the **exit** and **profil** system calls and the **monitor** subroutine in *AIX Operating System Technical Reference*.

1.1.333 proff

Purpose
nroff for the IBM 4201 Proprinter and IBM 5152 Graphics Printer.

Syntax

```
         +-----------------------+
proff ---¦ +--------------------+ +--- files ---¦
         +-¦ -g   -Lparms       +-+
           ¦ -    nroff-options ¦¦
          ¦¦ -Pxx               ¦¦
          ¦+--------------------+¦
          +----------------------+
```

Note: This command does not have MBCS support.

Description

The **proff** command formats text in the named **files** for the IBM 4201 Proprinter (default) and the IBM 5152 Graphics Printer. See also "nroff, troff" in topic 1.1.301.

If no **file** argument is present, the standard input is read. An argument consisting of a single minus (**-**) is taken to be a file name corresponding to standard input.

Flags

-g	Produces output for the IBM 5152 Graphics Printer.
-t	Sends output to standard output.
-P xx	Sends output to printer **xx** which corresponds to an entry in **/etc/printcap**. The default is taken from the **PRINTER** environment variable, if it exists; otherwise **lp** is used.
-L parms	Passes parameter **parms** to **lpr**. This may be a single parameter (for example, **-L-h**) or a string (for example, **-L"-h -r -m"**).

All other options are passed to the **nroff** command.

Files

/usr/lib/term/tabpro	IBM 4201 Proprinter driving tables.
/usr/lib/term/tabgra	IBM 5152 Graphics Printer driving tables.
/usr/lib/ibmlp/ibmpro	IBM 4201 Proprinter lpr filter.
/usr/lib/ibmlp/ibmgra	IBM 5152 Graphics Printer lpr filter.
/usr/adm/lpd-errs	Error message file (site dependent).

Related Information

See the following commands: "colpro" in topic 1.1.79, "eqn, neqn, checkeq" in topic 1.1.152, "lpr" in topic 1.1.248, "nroff, troff" in topic 1.1.301, and "tbl" in topic 1.1.463.

1.1.334 prompter

Purpose
Invokes a prompting editor.

Syntax

```
              +---------------+   +--------------+  +--- -prepend ---+   +-- -no
prompter ---¦                +---¦              +--¦     one of     +---¦    or
              +- -erase char -+   +- -kill char -+  ¦ +------------+ ¦   ¦ +----
                                                    +-¦ -prepend   +-+   +-¦ -ra
                                                      ¦ -noprepend ¦       ¦ -no
                                                      +------------+       +----

prompter --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **prompter** command is used to invoke the prompter editor for message entry. **prompter** is not designed to be run directly by the user; it is designed to be called by other programs. The **prompter** command is part of the Message Handling (MH) package.

The **prompter** command opens the specified file and scans it for empty components such as **To:** and prompts you to fill in those fields. If you press **Enter** without adding text, **prompter** later deletes the component.

After the first blank line or line of dashes in the file, **prompter** accepts text for the body of the message. If the body already contains text and the flag **-noprepend** is specified, **prompter** displays the text followed by the message:

```
  --------Enter additional text
```

prompter appends any text that is entered to the message body. If you specify the **-prepend** flag, **prompter** displays the following message instead:

```
  --------Enter initial text
```

When you press **END OF FILE**, **prompter** ends text entry and returns control to the calling program.

Flags

-erase char	Sets the character to be used as the erase character. You can specify the octal representation of the character in the form \nnn, or you can specify the character itself.
-help	Displays help information for the command.
-kill char	Sets the character to be used as the kill character. You can specify the octal representation of the character in the form \nnn, or you can specify the character itself.

-noprepend	Places additional text below any text already in the message body.
-norapid	Displays any text already in the message body. This is the default.
-prepend	Places additional text before any text already in the message body. This is the default.
-rapid	Does not display text already in the message body.

Profile Entries

Msg-Protect:	Sets the protection level for your new message files.
prompter-next:	Specifies the editor used after exiting **prompter**.

Files

$HOME/.mh_profile	The MH user profile.
/tmp/prompter*	A temporary copy of a message.

Related Information

See other MH commands: "comp" in topic 1.1.85, "dist" in topic 1.1.131, "forw" in topic 1.1.174, "repl" in topic 1.1.369, "whatnow" in topic 1.1.533.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.335 proto

Purpose
Constructs a prototype file for a file system.

Syntax

```
                          +----------+
/etc/proto -- directory --¦          +---¦
                          +- prefix -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **proto** command makes a prototype file for a file system or part of a file system. Use the prototype file as input to the **mkfs** command to construct a file system according to a predefined template. The prototype file consists of a recursive directory listing of every file on the file system, with its owner, group, and protection. It also contains the file from which the prototype file is to be initialized, formatted as described in the **mkfs** command.

Specify the base directory from which the prototype file is made with **directory**. The prototype file includes the complete subtree below **directory** that is contained on the same file system as **directory**.

The **prefix** parameter is added to the names of all the initialization files, forcing the initialization files to be taken from a place other than the prototype. Before the output from **proto** can be used with **mkfs**, **mkfs** needs a startup program, a file system size, and an i-list size. Link information is not preserved with the **proto** command.

The collating sequence is determined by the **ct_collate** array in the **NLctab** subroutine.

Related Information

See the following command: "mkfs" in topic 1.1.269.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.336 prs

Purpose
Displays a Source Code Control System (SCCS) file.

Syntax

```
       +-----------------+  +-------------------------+
prs ---¦ +-------------+ +--¦                 one of  +--- file ---¦
       ¦ ¦ -a          ¦ ¦  ¦ +- -rSID ----+   +----+  ¦          ¦
       +-¦ -d "string" +-+  +-¦            +---¦ -e +--+ +--------+
         +-------------+¦     ¦            ¦   ¦ -l ¦
        +---------------+     +- -ccutoff -+   +----+
```

Description
The **prs** command reads **file**s, and writes to standard output a part or all of a Source Code Control System (SCCS) file. If you specify a directory in place of **file**, **prs** performs the requested actions on all SCCS files (those with a name that has the **s.** prefix). If you specify a **-** (minus) in place of **file**, **prs** reads standard input and interprets each line as the name of an SCCS file. **prs** continues to take input until it reads an END-OF-FILE character (**Ctrl-D**).

1.1.336.1 Data Keywords

Data keywords specify which parts of an SCCS file are to be retrieved and written to standard output. All parts of an SCCS file have an associated data keyword. There is no limit to the number of times a data keyword can appear in a **string**. The information that **prs** displays consists of user-supplied text and appropriate values (extracted from the SCCS file) substituted for the recognized data keywords in the order of appearance in **string**. The format of a data keyword value is either simple, in which the keyword substitution is direct, or multi-line, in which the substitution is followed by a carriage return. Text is any characters other than recognized data keywords. Specify a tab character with **\t** and a carriage return/new-line character with a **\n**. Remember to quote the **\t** and **\n** with an extra **** to prevent shell from interpreting the **** and just passing a **t** or **n** to **prs** as text.

Table 1-5 lists the keywords associated with information in the delta table in the SCCS file (see the **sccsfile** file in *AIX Operating System Technical Reference* for the structure of an SCCS file).

Table 1-5. Delta Table Keywords for SCCS File

Keyword	**Data Represented**	**Value**	**Format**
:R:	Release number	num	Simple
:L:	Level number	num	Simple
:B:	Branch number	num	Simple
:S:	Sequence number	num	Simple
:I:	SCCS ID string (SID)	**:R:**:**L:**:**B:**:**S:**	Simple
:Dy:	Year delta created	YY	Simple
:Dm:	Month delta created	MM	Simple
:Dd:	Day delta created	DD	Simple
:D:	Date delta created	YY/MM/DD	Simple
:Th:	Hour delta created	HH	Simple
:Tm:	Minute delta created	MM	Simple
:Ts:	Second delta created	SS	Simple
:T:	Time delta created	HH:MM:SS	Simple
:DT:	Delta type	D or R	Simple
:P:	User who created the delta	login name	Simple
:DS:	Delta sequence number	num	Simple
:DP:	Previous delta sequence number	num	Simple

:Dt:	Delta information	**:DT:**:**I**:**D**: :**T**::**P**::**DS**::**DP**:	Simple
:Dn:	Sequence numbers of deltas included	**:DS:**...	Simple
:Dx:	Sequence numbers of deltas excluded	**:DS:**...	Simple
:Dg:	Sequence numbers of deltas ignored	**:DS:**...	Simple
:DI:	Sequence numbers of deltas included, excluded, and ignored	**:Dn:**/**:Dx:**/**:Dg:**	Simple
:Li:	Lines inserted by Delta	num	Simple
:Ld:	Lines deleted by Delta	num	Simple
:Lu:	Lines unchanged by Delta	num	Simple
:DL:	Delta line statistics	**:Li:**/**:Ld:**/**:Lu:**	Simple
:MR:	MR numbers for delta	text	Multi-lie
:C:	Comments for delta	text	Multi-lie

Table 1-6 lists the keywords associated with the header flags in the SCCS file. For more information of Header flags, see Table 1-2 in topic 1.1.16.3.

Table 1-6. Header Flag Keywords for SCCS File

Keyword	**Data Represented**	**Value**	**Format**
:Y:	module type	text	simple
:MF:	MR validation flag set	yes or no	Simple
:MP:	MR validation program name	text	Simple
:KF:	Keyword/error warning flag set	yes or no	Simple
:BF:	Branch flag set	yes or no	Simple
:J:	Joint edit flag set	yes or no	Simple
:LK:	Locked releases	**:R:**...	Simple
:Q:	User defined keyword	text	Simple
:M:	Module name	text	Simple
:FB:	Floor boundary	**:R:**	Simple

:CB:	Ceiling boundary	**:R:**	Simple
:Ds:	Default SID	**:I:**	Simple
:ND:	Null Delta flag set	yes or no	Simple
:FL:	Header flag list	text	Multi-li e

Table 1-7 lists the keywords associated with other parts of the SCCS file.

Table 1-7. Other Keywords for SCCS File

Keyword	**Data Represented**	**Value**	**Format**
:UN:	user names	text	Multi-li e
:FD:	descriptive text	text	Multi-li e
:BD:	body of text	text	Multi-li e
:GB:	text in a g-file	text	Multi-li e
:W:	a **what** string	**:Z::M: \tab :I:**	Simple
:A:	a **what** string	**:Z::Y::M::I::Z**	Simple
:Z:	a **what** string delimiter	**@(#)**	Simple
:F:	SCCS file name	text	Simple
:PN:	SCCS file path name	text	Simple

Flags

Each flag or group of flags applies independently to each named file.

-a Writes information for the specified deltas, whether they have been removed (see "rmdel" in topic 1.1.377). If you do not specify the **-a** flag, **prs** supplies information only for the specified deltas that have not been removed.

-ccutoff Specifies a **cutoff** date and time for the **-e** and **-l** flags. Specify **cutoff** in the following form:

```
YY[MM[DD[HH[MM[SS]]]]]
```

All omitted items default to their maximum values, so specifying **-c8402** is the same as specifying **-c840229235959**. You can separate the fields with any non-numeric characters. For example, you can specify **-c84/2/20,9:22:25** or **-c"84/2/20 9:22:25"** or **"-c84/2/20 9:22:25"**.

-d"string" Specifies the data items to be displayed. **string** is a string consisting of optional text and SCCS file data keywords. You must enclose all text and spaces in **string**

in quotation marks.

-e	Requests information for all deltas created **earlier** than and including the delta specified by the **-r** flag.
-l	Requests information for all deltas created **later** than and including the delta specified by the **-r** flag.
-rSID	Specifies the **SID** of a delta for which **prs** will retrieve information. If no SID is specified, **prs** retrieves the information for the SID of the highest numbered delta.

Files

/tmp/pr?????	Temporary **prs** files.

Related Information

See the following commands: "admin" in topic 1.1.16, "delta" in topic 1.1.117, "get" in topic 1.1.186, and "sccshelp" in topic 1.1.411.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.337 ps

Purpose
Reports process status.

Syntax

```
      +-- -t console ---+   +--------+   +- -c/dev/kmem -+   +----- -n/unix --
ps ---¦     one of      +---¦ +----+ +---¦               +---¦
      ¦ +-------------+ ¦   +-¦ -f +-+   +- -c corfile --+   +- -n kernel-imag
      +-¦ -a -g glist +-+     ¦ -k ¦
        ¦ -e -p plist ¦      ¦¦ -l ¦¦
        ¦ -d -t tlist ¦      ¦¦ --  ¦¦
        ¦    -u ulist ¦      ¦+----+¦
        +-------------+      +------+

      +---------------+   +- /unix ------------------------+
ps ---¦ +-----------+ +---¦                +- /dev/knum -+ +---¦
      +-¦ a k s w  ¦+-+   +- kernel-image -¦             +-+
        ¦ c l t x   ¦¦                     +- corefile --+
       ¦¦ e n o U   ¦¦
       ¦¦ g r v pid ¦¦
       ¦+-----------+¦
       +-------------+


-----------------
¦ No spaces between these options.
```

Description
The **ps** command writes certain information about active processes to standard output. Without flags, **ps** displays information about the processes with your effective user ID.

The column headings in a **ps** listing have the following meaning. The letters f and l following the column heads indicate which flags cause the corresponding heading to appear. If **all** follows the column head, that heading always appears. The **-f** and **-l** flags determine only what information is provided about a process; they do not determine which processes are listed.

F (l)
Flags (octal and additive) associated with the process:

00	None apply
01	In core
02	System process
04	Locked in core (for example, for physical I/O);
10	Waiting for a page default, or forking
20	Being traced by another process
40	Another tracing flag
100	Process has shared text.

S (l)
The state of the process:

0	Nonexistent
S	Sleeping
W	Waiting

R Running
I Intermediate
Z Canceled
T Stopped
K Available kernel process
X Growing.

UID (f,l)
The user ID of the process owner; the login name is displayed with the **-f** flag.

PID   (all)
The process ID of the process.

PPID (f,l)
The process ID of the parent process.

C (f,l)
Processor utilization for scheduling. The higher the number, the higher the utilization.

STIME (f)
Starting time of the process. The appearance of this field is locale-specific.

PRI (l)
The priority of the process; higher numbers mean lower priority.

NI (l)
Nice value; used in calculating priority. The default is 20. The higher the number, the lower the priority.

ADDR (l)
The location of the process's kernel stack. If in memory, it is the address of the u area of the process. On 386 machines, a pair of addresses are the page of the kernel stack followed by the page with the user structure. If swapped out, information of the form **bbbb (p)** is output, where **bbbb** is the block number on swap partition number **p**.

SZ (l)
The size in 1K blocks of the core image of the process. This includes the stack, data, and text segments.

WCHAN (l)
The event for which the process is waiting or sleeping; if blank, the process is running. Numbers refer to addresses in the kernel's address space.

TTY (all)
The controlling work station for the process.

TIME (all)
The total execution time for the process. The value is specified in minutes and seconds.

CMD (all)
The command name; the full command name and its parameters are displayed with the **-f** flag. The command name may not be displayed if it is issued from a terminal with a different code page set.

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked **<defunct>**.

With the **-f** flag, **ps** determines what the command name and parameters were when the process was created by examining memory or the paging area. If it cannot find this information, the command name, as it would appear without the **-f** flag, displays in square brackets.

Notes:

1. Things can change while **ps** is running.

2. Some data displayed for defunct processes are irrelevant.

3. To obtain System V-like behavior, precede each option with a hyphen (-). To obtain BSD-like behavior, do not precede options with the hyphen. (Unknown options are ignored by the BSD-compatible parser.)

4. Remote child process selection only applies to processes that are associated with a terminal. Child processes on remote sites are only selected as long as they continue to be associated with the same terminal.

Flags

The **- -** flag shows processes at your local work station.

In case you do not specify a "-" before your options: Specifying **a** causes other users' processes to be candidates to be printed; specifying **x** includes processes without control terminals in the candidate pool.

All output formats include, for each process, the process ID (PID), control terminal of the process (TT), CPU time used by the process (TIME), (this includes both user and system time), the state (STAT) of the process, and an indication of the COMMAND which is running. The state is given by a sequence of four letters; the first letter indicates the reliability of the process: R for runnable processes, T for stopped processes, P for processes in page wait, D for those in disk (or other short term) waits, S for those sleeping for less than about 20 seconds, and I for idle processes (sleeping longer than about 20 seconds. The second letter indicates whether a process is swapped out, showing W if it is or a blank if it is loaded (in-core). A process which has specified a soft limit on memory requirements and which is exceeding that limit shows >; such a process is (necessarily) not swapped. The third letter indicates whether a process is running with altered CPU scheduling priority (nice). If the process priority is reduced, an N is shown. If the process priority has been artificially raised, a < is shown. Processes running without special treatment just have a blank. The fourth letter is reserved for future use.

A second argument is taken to be the file containing the system's namelist. Otherwise, **/unix** is used. A third argument tells **ps** where to look for core if the **k** option is given, instead of **/core**. If a fourth argument is given, it is taken to be the name of a paging file to use instead of the default **/dev/swap**.

Fields which are not common to all output formats:

USER	Name of the owner of the process.

%CPU	CPU utilization of the process. This is a decaying average over up to a minute of previous (real) time. Since the time base over which this is computed varies, (since processes may be very young), it is possible for the sum of all **% CPU** fields to exceed %100.
NICE	(or NI) process scheduling increment (see **setpriority** system call in the *AIX Operating System Technical Reference*).
SIZE	Virtual size of the process (in 1024 byte units).
RSS	Real memory (resident set) size of the process (in 1024 byte units).
LIM	Soft limit on memory used, specified via a call to **setrlimit**. If no limit has been specified, this is shown as **xx**. (See **setrlimit** system call in the *AIX Operating System Technical Reference*.)
TSIZ	Size of text (shared program) image. This parameter gives size in K bytes.
TRS	Size of resident (real memory) set of text.
%MEM	Percentage of real memory used by this process.
RE	Residency time of the process (seconds in core).
SL	Sleep time of the process (seconds blocked).
PAGEIN	Number of disk I/Os resulting from references by the process to pages not loaded in core.
UID	Numerical user-ID of process owner.
PPID	Numerical ID of parent of process.
CP	Short-term CPU utilization factor (used in scheduling).
PRI	Process priority (non-positive when in non-interruptible wait).
ADDR	Swap address of the process.
WCHAN	Event on which the process is waiting (an address in the system). A symbol is chosen that classifies the address, unless numerical output is requested (see **n** flag). In this case, the initial part of the address is trimmed off and is printed hexadecimally. That is, 0x80004000 prints as 4000.
F	Flags associated with the process as defined in **/usr/include/sys/proc.h** for the **p** flag values:

SLOAD	000001	in core
SSYS	000002	swapper or pager process
SLOCK	000004	process being swapped out
SSWAP	000008	save area flag
STRC	000010	process is being traced
SWTED	000020	another tracing flag
SULOCK	000040	user settable lock in core
SPAGE	000080	process in page wait state

SKEEP	000100	another flag to prevent swap out
SDLYU	000200	delayed inlock of pages
SWEXIT	000400	working on exiting
SPHYSIO	000800	doing physical I/O (bio.c)
SVFORK	001000	process resulted from vfork ()
SVFDONE	002000	another vfork flage
SNOVM	004000	no vm, parent in vfork()
SPAGI	008000	init data space on demand from inode
SANOM	010000	system detected anomalous vm behavior
SUANOM	020000	user warned of anomalous vm behavior
STIMO	040000	timing out during sleep
SDETACH	080000	detached inherited by init
SOUSIG	100000	using old signal mechanism

A process that has exited and has a parent that has not yet waited for the process is marked <defunct>; a process which is blocked trying to exit is marked <exiting>; **ps** makes an educated guess as to the file name and arguments given when the process was created by examining memory or the swap area. The method is inherently somewhat unreliable and in any event a process is entitled to destroy this information, so the names cannot be counted on too much.

-a Writes to standard output information about all processes except the process group leaders and processes not associated with a terminal.

-c corefile
Uses **corefile** instead of the default **/dev/mem**. **corefile** is a core image file that has been created by the **Ctrl-**(left)**Alt-Pad7** key sequence.

-d Writes information to standard output about all processes except the process group leaders.

-e Writes information to standard output about all processes except kernel processes.

-f Generates a full listing. The meaning of columns in a full listing is described on page 1.1.337.

-g glist Writes information to standard output only about processes that are in the process groups listed in **glist**. The **glist** is either a comma-separated list of process-group identifiers or a list of process-group identifiers enclosed in double quotation marks (" ") and separated from one another by a comma and/or one or more spaces.

-k Writes information to standard output about kernel processes. Otherwise, it does not list kernel processes.

-l Generates a long listing. The meaning of a long listing is described on page 1.1.337. Size parameter gives stack+data in 4K blocks.

-n kernel-image
Takes **kernel-image** as the name of an alternate **kernel-image** file (**/unix** is the default).

-p plist Displays only information about processes with the process numbers specified in **plist**. **plist** is either a comma-separated list of process-ID numbers or a list of process-ID numbers enclosed in double quotation marks (" ") and separated from one another by a comma and/or one or more spaces.

-r Displays information about children of selected processes that are executing on other cluster sites.

-t tlist Displays only information about processes associated with the work stations listed in **tlist**. **tlist** is either a list of comma-separated work-station identifiers or a list of work-station identifiers enclosed in double quotation marks (" ") and separated from one another by a comma and/or one or more spaces.

-u ulist Displays only information about processes with the user ID numbers or login names specified in **ulist**. **ulist** is either a comma-separated list of user IDs or a list of user IDs enclosed in double quotation marks (" ") and separated from one another by a comma and/or one or more spaces. In the listing, **ps** displays the numerical user ID unless the **-f** flag is used; then it displays the login name.

In case you do not specify a "-" before your options.

a Asks for information about all processes with terminals (ordinarily only one's own processes are displayed).

c Prints the command name, as stored internally in the system for purposes of accounting, rather than the command arguments, which are kept in the process' address space. This is more reliable, if less informative, since the process is free to destroy the latter information.

e Asks for the environment to be printed as well as the arguments to the command.

g Asks for all processes. Without this option, **ps** only prints "interesting" processes. Processes are deemed to be uninteresting if they are process group leaders. This normally eliminates top-level command interpreters and processes waiting for users to login on free terminals.

k Causes the core image file that has been created by **Ctrl-(left) Alt-End** sequence to be used in place of **/dev/kmem**. This is used for postmortem system debugging.

l Asks for a long listing, with fields PPID, CP, PRI, NI, ADDR, SIZE, RSS and WCHAN as described earlier.

n Asks for numerical output. In a long listing, the WHCAN field is printed numerically rather than symbolically, or, in a user listing, the USER field is replaced by a UID filed.

r Display information for remote children of the processes selected on the current site.

s Adds the size SSIZ of the kernel stack of each process (for use by system maintainers) to the basic output format.

t Restricts output to processes whose controlling **tty** is **x** (which should be specified as printed by **ps**; that is **t3** for **tty3**, **tconsole** for console, **tc0** for **ttyc0**, **t?** for processes with no **tty**, **t** for processes at the current **tty**, etc.). This option must be the last one given.

u A user-oriented output is produced. This includes fields USER, %CPU, NICE, SIZE and RSS.

v A version of the output containing virtual memory statistics as output. This includes fields RE, SL, PAGEIN, SIZE, RSS, LIM, TSIZ, %CPU and %MEM.

w Use a wide output format (132 columns rather than 80); if repeated, (that is, ww), use arbitrarily wide output. This information is used to decide how much of long commands to print.

x Asks even about process with no terminal.

U Causes **ps** to update a private data base where it keeps system information. Thus **ps U** should be included in the **/etc/rc** file. Size parameter gives stack+data+text in K bytes.

pid A process number may be given, (indicated here by #), in which case the output is restricted to that process. This option must also be last.

Examples

1. To list the processes that you have started:

   ```
   ps
   ```

   This command displays a summary of information about the processes associated with your work station.

2. To display all process information available:

   ```
   ps  -e  -f  -l  -k
   ```

   This command displays all of the information (**-l -f**) about all processes (**-e** and **-k**).

3. To list processes owned by specific users:

   ```
   ps  -f  -l  -utom,jane
   ```

   This command displays all the information available (**-l -f**) about the processes being run by the users **tom** and **jane**.

4. To list processes associated with specific work stations:

   ```
   ps  -t-,console
   ```

   This command displays information about processes not connected to any work station (**-t-**), and processes associated with the work station **/dev/console**.

5. To list the process that you have started:

   ```
   ps  u
   ```

This command displays BSD-like user information ("u") about process with your UID (default).

6. To display all process information available:

   ```
   ps  lxag
   ```

   This command displays a BSD-like long listing ("l") about all processes ("xag").

7. To display a specific process' information:

   ```
   ps 1
   ```

   This command displays BSD-like summary information for process 1 ("1").

8. To list processes associated with a specific work station>

   ```
   ps vt01
   ```

   This command displays BSD-like virtual information ("v") about processes associated with **tty01** ("to1").

Files

/machine_name/unix	System kernel image.
/dev/mem	Memory.
/etc/passwd	Supplies UID information.
/etc/ps_data	Internal data structures.
/dev	Searched to find work station ("TTY") names.

Related Information

See the following commands: "kill" in topic 1.1.221 and "nice" in topic 1.1.296.

See **setpriority subroutines** and **setrlimit subroutines** in the *AIX Operating System Technical Reference*.

1.1.338 pstart, penable, pshare, pdelay

Purpose
Enables or reports the availability of login ports.

Syntax

```
          +--------+   +--- -a ---+
pstart ---¦ +----+ +---¦          +---¦
          +-¦ -i +-+   +- device -+
            ¦ -w ¦¦           ¦
           ¦+----+¦     +--------+
           +------+


  one of
+---------+   +--------+   +--- -a ---+
¦ penable +---¦        +---¦          +---¦
¦ pshare  ¦   +-- -i --+   +- device -+
¦ pdelay  ¦           ¦             ¦
+---------+    +------+     +--------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **pstart**, **penable**, **pshare**, and **pdelay** commands each enable a set of login ports in the **/etc/ports** file. Enabling a port makes the port available to log in. To enable a port, the stanza for the port in **/etc/ports** must include **enabled=true** and the stanza for the port in **/etc/inittab** must include **action=respawn**. The system enables a port by updating an entry in the **/etc/inittab** file and then sending a signal to **init**. When **init** receives the signal and reads the updated status entry, it takes the appropriate action.

Use the **device** parameter to specify the ports to be enabled. Permitted values for **device** include:

A full device name, such as **/dev/tty1**.
A simple device name, such as **tty1**.
A general class of devices in the form **attribute=value**, which is equivalent to naming each port with a stanza in **/etc/ports** that includes the specified attribute).

If you do not specify a **device** to enable, each command reports the names of currently enabled ports in its set. You must have superuser privileges to execute these commands.

Subtopics

1.1.338.1 pstart

The **pstart** command enables all ports (normal, shared, and delayed) that are enabled in the **/etc/ports** file. If you do not specify a **device** to enable, **pstart** reports the names of all enabled ports and tells whether they are currently enabled as normal, shared, or delayed. Usually the command is run in the form **pstart -a -i** from **/etc/rc** to enable all ports on a multiuser system.

1.1.338.2 penable

The **penable** command enables normal ports that are enabled in the **/etc/ports** file. Normal ports are ports that are asynchronous and only allow users to login to those ports. No outgoing use of the port is allowed while it is enabled. This command is equivalent to the statement **penable enabled=true**. If you do not specify a **device**, **penable** reports the names of the currently enabled normal ports.

1.1.338.3 pshare

The **pshare** command enables shared ports that are enabled in the **/etc/ports** file. Shared ports are bi-directional. This command is equivalent to the statement **pshare enabled=share**. If you do not specify a **device**, **pshare** reports the names of the currently enabled shared ports. To enable shared ports, **getty** attempts to create a lock file in **/etc/locks** which contains the ASCII process ID of the **getty** process. If the port is already in use by some other process, **getty** waits until the port is available and tries again.

1.1.338.4 pdelay

The **pdelay** command enables delayed ports that are enabled in the **/etc/ports** file. Delayed ports are ports that are enabled like shared ports except that the login herald is not displayed until the user types one or more characters (usually carriage-returns). If the port is directly connected to a remote system or connected to an intelligent modem, the port is usually enabled as a delayed port to prevent the **getty** from talking to a **getty** on the remote side or to the modem on a local connection, thereby consuming system resources. This statement is equivalent to **pdelay enabled=delay**. If you do not specify a **device**, **pdelay** reports the names of the currently enabled delayed ports.

Flags

-a With **pstart**, this flag enables all ports enabled in the **/etc/ports** file (normal, shared, and delayed ports). With **penable**, this flag enables all normal ports that are enabled in the **/etc/ports** file. With **pshare**, this flag enables all shared ports that are enabled in the **/etc/ports** file. With **pdelay**, this flag enables all delayed ports that are enabled in the **/etc/ports** file.

-i Reinitializes an existing **/etc/inittab** file instead of updating the existing one. You typically use this flag in the **/etc/rc** command file to re-establish default port enabling before starting up the system with multiple users.

Examples

1. To display the names of all ports (normal, shared, and delayed) currently enabled and how they are enabled:

   ```
   pstart
   ```

2. To enable all normal, shared, and delayed ports that are enabled in **/etc/ports**, and reinitialize existing **/etc/ports**:

   ```
   pstart  -a -i
   ```

3. To enable the work station attached to the **/dev/tty2** port as a shared port:

   ```
   pshare  /dev/tty2
   ```

4. To display the names of the delayed ports that are currently enabled:

   ```
   pdelay
   ```

5. To enable all **9600** baud ports as delayed:

   ```
   pdelay  speed=9600
   ```

Files

/etc/locks	Contains lock files for **pshare** and **pdelay**.
/etc/ports	Contains descriptions of known normal, shared, and delayed ports.
/etc/inittab	Contains current status of each known login port.

Related Information

See the following commands: "init, telinit" in topic 1.1.208 and "pdisable, phold" in topic 1.1.314.

See the **ports** and **inittab** files in *AIX Operating System Technical Reference*.

1.1.339 pstat

Purpose

Prints system facts.

Syntax

```
              +------------- -u pid ---------------+
              ¦     +- -i -+                       ¦
/etc/pstat ---¦   +-¦      +----------+ +------+   +---
              ¦   ¦ +- -I -+          +-¦      +-+ ¦
              ¦ +-¦ +----------+      ¦ +- -a -+ ¦ ¦
              ¦ ¦ +-¦ +- -U -+ +- -p -+          ¦ ¦
              +-¦   +-¦      +-+                 +-+
                ¦     +- -v -+                   ¦¦
               ¦¦ +----+                         ¦¦
               ¦¦ ¦ -f ¦                         ¦¦
               ¦+-¦ -l +-------------------------+¦
               ¦  ¦ -P ¦                          ¦
               ¦  ¦ -q ¦                          ¦
               ¦  ¦ -s ¦                          ¦
               ¦  ¦ -t ¦                          ¦
               ¦  ¦ -T ¦                          ¦
               ¦  +----+                          ¦
               +----------------------------------+
    +----------------------------+
 ---¦             +------------+ +---¦
    +- corefile --¦            +-+
                  +- namelist -+
```

Note: This command does not have MBCS support.

Description

The **pstat** command interprets the contents of certain system tables in kernel memory. If a kernel core dump, **corefile**, is given, the tables are sought there; otherwise, in the **/dev/kmem** file. The **corefile** cannot be the application program core file. If **namelist** (for example, the name of an AIX kernel) is given, that name list is used; otherwise, the **/unix** file is assumed. The **/dev/kmem** file can be specified as **corefile** in order to specify only **namelist**. Certain options such as **-v** below require additional suboptions.

Options

-a Under **-p, -i** or **-I**, describes all process slots or inode slots rather than just active ones.

-f Prints the open file table with these headings:

LOC	Core location of this table entry.
FLG	Miscellaneous state variables encoded thus:

R	Open for reading.
W	Open for writing.
P	Pipe (no longer used).
T	Token is present.

L	File block is locked.
K	Token is wanted by another site.
N	Ready for netfork (token control block has been set up at storage site (SS) if necessary).
S	Open socket.
A	Open for append.

CNT	Number of processes that know this open file.
INO	Location of the inode table entry for this file.
TCB	Index into the TCB table at the SS for this file (see **-q** option on page 1.1.339).
TSITE	Token manager site.
WNTTOK	Number of local processes waiting for token to arrive.
TLCKWNT	Number of processes waiting to lock file block.
OFFS	The file offset (see the **lseek** system call in the *AIX Operating System Technical Reference*).

-i Prints the inode table with these headings:

LOC Core location of this table entry.

FLAGS Miscellaneous state variables encoded thus:

L	Locked.
u	Update time (filsys) must be corrected.
M	File system is mounted here.
W	Wanted by another process (L flag is on).
T	Contains a text file.

CNT	Number of open file table entries for this inode.
DEVICE	Major and minor device number of file system in which this inode resides.
INO	I-number within the device.
MODE	Mode bits (see "chmod" in topic 1.1.67).
NLK	Number of links to this inode.
UID	User ID of owner.
SIZE/DEV	Number of bytes in an ordinary file, or major and minor device of special file.

-I Prints the inode table (including AIX-specific fields) with these headings:

SLT Index into the inode table for this entry.

FLAGS Miscellaneous state variables encoded thus:

L	Locked.
u	Update time (filsys) must be corrected.

D	Storage site has gone down.
M	File system is mounted here.
W	Wanted by another process (L flag is on).
T	Contains a text file.
G	Change time must be corrected.
O	Inode is for an old version.
C	Commit is in progress.
i	Update propagation in is in progress.
o	Update propagation out is in progress.
s	Out of space error has occurred on a write to this file.
w	Read/write error has occurred on this file.

GFS The global file system number. (GFS, INO) uniquely identifies a file.

INO I-number within the device.

AREF Count of all uses of this inode.

OREF Count of local opens of this inode.

URD Count of number of local opens for read.

UMD Count of number of local opens for mod (write).

SIS At a US which is not the SS, the slot of the SS's inode.

DEVICE Major and minor device number of file system in which this inode resides.

DFLAGS AIX disk inode flags, as follows (hex):

01	DIDEL (file has been deleted).
02	DISTORE (copy of file is stored locally).
10	DIALLOC (this inode is allocated).
20	DIHIDDEN (this file is a hidden directory).
40	DILONGDIR (this file is a long directory in BSD format).
80	DILINK (this file is a symbolic link to another file).
100	DIFORCE (DISTORE is forced to retain current value).
200	DIXIPX (this file is used for x386 IPC support).
400	DIMOUNTEDON (this file is mounted on inode #...).
800	This file is a socket.

UID User ID of owner.

PIS During update propagation, in the inode slot of tandem inode.

MODE Mode bits (see "chmod" in topic 1.1.67).

NL Number of links to this inode.

SIZE/DEV Number of bytes in an ordinary file, or the device site, major device and minor device of a special file.

VERSION The version number of this file.

RL The index into the lock table for any record locks applied to this file (see **-1** option on page 1.1.339).

FT	At the SS of a pipe, the index into the token control block table of the file offset token (see **-q** option on page 1.1.339).
IT	At the SS of a regular file or block special file, the index into the token control block table of the file data token (see **-q** option on page 1.1.339).
W	At the using site (US), the number of processes waiting for the file data token.
L	At the US, the number of processes currently using the file data token.
CSSMAP	At the CSS, the bitmap of active storage sites.
SSMAP	At the SS, the bitmap of active using sites.
MODMAP	At the SS, the bitmap of sites having this file open for mod (write).

For pipes (both regular and named), the following is also given:

ROPEN	Read end of pipe currently open.
WOPEN	Write end of pipe currently open.
RLOCAL	Current pipe reader is on local site.
WLOCAL	Current pipe writer is on local site.
RCLOSED	The read end of the pipe has been closed.
WRITING	A write system call on this pipe is in progress.
REMOTE	A remote system call on this pipe is in progress.

At SS, read end and write end are open at the save remote site. OPEN At SS, either now or in the past, had both ROPEN and WOPEN. rptr If RLOCAL or at the SS, the current read pointer. xread IF RLOCAL or at the SS, this file offset records the amount of data requested in a read system call which is in progress. xwrite IF WLOCAL or at the SS, this file offset records the amount of data being written by a write system call which is progress. lastf At the SS, the logical page number of the page most recently freed.

-l Prints the file record lock table, including the following fields:

LOC	Core location of this table entry.
PROC	Process table pointer for the process holding this lock (if local).
PID	Process id for the process.
SITE	Site on which the process is running.
MODE	Type of lock.
LOW	Lower bound of the region which is locked.

HIGH Higher bound of the region.

WAITCT Number of processes waiting for this region to be unlocked.

NEXT Link to another lock if multiple regions are locked. This is also used to link items on the free list.

-p Prints process table for active processes with these headings:

LOC The core location of this table entry.

S Run state encoded thus:

0	No process.
1	Waiting for some event.
3	Runnable.
4	Being created.
5	Being terminated.
6	Stopped under trace.

F Miscellaneous state variables, or-ed together (hexadecimal):

000001	Loaded.
000002	Scheduler process.
000004	Locked for swap out.
00000B	Swapped out.
000010	Traced.
000020	Used in tracing.
000040	Locked in by **lock** (see "lock" in topic 1.1.239).
000080	In page-wait.
000100	Prevented from swapping during **fork** (see the **fork** system call in the *AIX Operating System Technical Reference*).
000200	Gathering pages for raw I/O.
000400	Exiting.
001000	Process resulted from a **vfork** (see the **vfork** system call in the *AIX Operating System Technical Reference*) which is not yet complete.
002000	Another flag for **vfork**.
004000	Process has no virtual memory, as it is a parent in the context of **vfork**.
008000	Process is demand paging data pages from its text inode.
010000	Process has advised of anomalous behavior with the **vadvise** system call (see *AIX Operating System Technical Reference*).
020000	Process has advised of sequential behavior with the **vadvise** system call.
040000	Process is in a sleep which will timeout.
080000	Parent of this process has exited and this process is now considered detached.
100000	Process used some new signal primitives, for example, the **sigset** system call (see *AIX Operating System Technical Reference*). More system calls will restart.
200000	Process is owed a profiling tick.

PRI Scheduling priority (see **nice** in the *AIX Operating System Technical Reference*).

SIG	Signals received (signals 1-32 coded in bits 0-31).
UID	Real user ID.
SLP	Amount of time process has been blocked.
TIM	Time resident in seconds; times over 127 coded as 127.
CPU	Weighted integral of CPU time, for scheduler.
NI	Nice level.
PGRP	Process number of root of process group (the opener of the controlling terminal).
PID	Process ID number.
PPID	Process ID of parent process.
ADDR	If in core, the page frame number of the first page of the 'u-area' of the process. If swapped out, the position in the swap area measured in multiples of 512 bytes.
SRSS	RSS at last swap (0 if never swapped).
WCHAN	Wait channel number of a waiting process.
LINK	Link pointer in list of runnable processes.
CLKT	Countdown for **alarm** (see the **alarm** system call in the *AIX Operating System Technical Reference*) measured in seconds.

-P Prints information about process tracking. This includes the (site, process-id) list of processes originating at this site but no longer on this site and also the (site, process-id) list for each parent of a remote process of its remote children.

-q Prints the token control block and token site request tables. These data structures are used by the token mechanisms that synchronize the concurrent access to files by processes on different sites within the network.

There are two types of tokens: file offset tokens and inode tokens. File offset tokens control access to a file table entry (see **-f** option on page 1.1.339) which is shared between multiple sites. Inode tokens, also known as file data tokens, control access to the content of the file. A list of token site request items is linked to each token control block, one for each site which shares the file. Allocated and unallocated entries for both tables are shown, except that entries which have never been allocated are omitted.

The token control block table is printed with the following headings:

LOC The core location of this table entry.

STATE One of the following:

FREE	Unallocated entry.
IDLE	Token not assigned to any site.

BUSY	Token assigned to first site in request queue.
ALERT	Site with token has been asked to give it up.

R	Indicates that the token control block is used for a file token open for reading.
M	Indicates that the token control block is used for a file token open for writing.

I	Indicates that the token control block is used for an inode token.

GFS	Global file system number for the file.
INUM	Inode number for the file.
I_PTR	Link back to the SS inode. Pipes and inode tokens only.
REQUESTS	Link to head of request queue (address in token site request table).
F_OFFSET	Read/write offset of the file at the last time the token was passed. File offset tokens only.
NEXT	Link to other token control blocks for this file. File offset tokens only.
READMAP	Sites currently holding the inode token for mod. Inode tokens only.
MODSITE	Site currently holding the inode token for mod. Inode tokens only.

The token site request table is printed with the following headings:

LOC	Core location of this table entry.
STATE	One of the following:

UNUSED	Unallocated entry.
TOKEN	This site currently had the token.
REQUEST	This site has requested the token.
ELIGIBLE	This site has not requested the token.
FREQUENT	This site wants to be queued back as a request after it gives up the token.

MOD	File is being modified.
SITE	Using site which is sharing the file.
FDES	Using site's index into its file table.

NEXT	Link to entries for other sites sharing this file.

-s Shows swap space usage.

-t Place the following under the **-t** flag:

DEV	Major device number.
DEL	Delimiter count.
BFLG	Berkeley flags.
BSTA	Berkeley state.
RAW	Number of characters in raw input queue.
MIN	Minor device number.
CAN	Number of characters in canonicalized input queue.
OUT	Number of characters in output queue.
IFLG	Input modes (octal).
OFLG	Output modes (octal).
CFLG	Control modes (octal).
LFLG	Line discipline modes (octal).
STATE	Miscellaneous state variables encoded thus:

T	Timeout.
C	Carrier is on.
B	Busy doing output.
A	Process is awaiting output.
X	Open for exclusive use.
S	Output stopped by SUSP **Ctrl-S**.
I	Wake up when input done.
O	Wake up when output done.
W	Waiting for output to drain.

PGRP	Process group for which this is controlling terminal.
DISC	Line discipline (old or new).

For more information on BFLG and BSTA, see **termio** in the *AIX Operating System Technical Reference*.

-T Displays the number of used and free slots in the several system tables. Useful for checking how full system tables have become if the system is under heavy load. This option overrides the **-f**, **-i**, **-p**, and **-x** flags.

-u Prints detailed information about the user structure for each of the processes in the process table.

-u pid
Displays detailed information about the user structure for the process identified by the **pid** argument. The option functions the

same as the **-U** option except it displays information about a specific process.

-v With the **-p** option, displays detailed process table information vertically. With the **-T** option, displays the number of netmsgs in use.

Files

/dev/kmem	Default *corefile*.
/unix	Default *namelist*.

Related Information

See the following commmand: "crash" in topic 1.1.96.

See the *AIX Operating System Technical Reference*: **stat** and **filsys** file formats.

1.1.340 ptx

Purpose

Generates a permuted index.

Syntax

```
      +------ -g3 ------+   +- -i/usr/lib/eign -+   +-----------------------
ptx ---¦ +------------+ +---¦      one of       +---¦           +-----------+
      +-¦ -b break -f +-+   ¦   +-----------+   ¦   +- infile --¦           -
        ¦ -g num   -r ¦¦    +---¦ -i ignore +---+               +- outfile --
       ¦¦ -w num   -t ¦¦        ¦ -o only   ¦
       ¦+-------------+¦        +-----------+
        +---------------+
```

Note: This command does not have MBCS support.

Description

The **ptx** command reads **infile** (standard input by default), creates a permuted index from its input, and writes to **outfile** (standard output by default).

The **ptx** command searches **infile** for keywords, sorts the lines, and generates the file **outfile**. **outfile** can then be processed with **nroff** or **troff** to produce a permuted index from the file **infile**.

The **ptx** command follows three steps:

1. In the permutation, generates one line for each keyword in an input line, and rotates the keyword to the front.
2. Sorts the permuted file.
3. Rotates the sorted lines so that the keyword comes at the middle of each line.

The resulting lines in output are in the form:

.xx, "**tail**" "**before_keyword**" "**keyword_and_after**" "**head**"

where **.xx** is an **nroff** or **troff** macro provided by the user, or provided by the **mptx** macro package (see the *AIX Operating System Technical Reference* for information on this macro package). The **before_keyword** and **keyword_and_after** fields incorporate as much of the line as will fit around the keyword when it is printed. **tail** or **head**, at least one of which is always the empty string, are wrapped-around pieces small enough to fit in the unused space at the opposite end of the line.

Notes:

1. Line length counts do not account for overstriking or proportional spacing.

2. Lines that contain a tilde (~) do not work because **ptx** uses that character internally.

Flags

-b break Uses the characters in the **break** file to separate words. Tab characters, new-line characters, and spaces are **always** used as break characters.

-f	Does not distinguish between uppercase and lowercase characters while sorting (see "sort" in topic 1.1.432).
-g num	Uses **num** as the number of spaces displayed between the four parts of the line. The default **num** is 3.
-i ignore	Does not use any words in the **ignore** file as keywords. If the **-i** and **-o** flags are not used, **/usr/lib/eign** is the default **ignore** file.
-o only	Uses only the words in the **only** file as keywords.
-r	Takes any leading nonblank characters of each input line to be a reference identifier separate from the text of the line. Attaches that identifier as a fifth field on each output line.
-t	Prepares the output for the photo typesetter.
-w num	Uses **num** as the length of the output line. The default line length is 72 characters for **nroff** and 100 for **troff**.

Files

/bin/sort	Sort program.
/usr/lib/eign	List of words to ignore.
/usr/lib/tmac/tmac.ptx	List of permuted index **nroff, troff** commands.

Related Information

See the following commands: "nroff, troff" in topic 1.1.301.

See the **mm** and **mptx** miscellaneous facilities in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.341 punbkend

Purpose
Punch backend (file transfer).

Syntax

```
                          +------------+
/usr/lpd/punbkend ---¦ +--------+ +---¦
                          +-¦ -uuser +-+
                            ¦ -nnode ¦
                           ¦¦ -ttext ¦¦
                           ¦¦ -fname ¦¦
                           ¦+--------+¦
                           +----------+
```

Note: This command is for the System/370 only.

Description
The file transfer function uses the print command internally to invoke the punch backend. The queue must be one which is defined in the **/etc/qconfig** file as being serviced by **/usr/lpd/punbkend**. The **punbkend** backend accepts the following options to build a tag (nodeid userid text) and a file name for the spool file:

-uuser The userid of the user who is to receive the spooled punch output. If omitted, the output is passed to RSCS and the userid of the tag is left blank.

-nnode The nodeid where the spooled output is sent. The **-nnode** option may be used as follows:

> If omitted but **-uuser** is specified, the user is assumed to belong to the local AIX/370 master-subsystem or the same subsystem as the sender.
>
> If the specified **node** is the nodeid for the local VM system and **-uuser** is also specified, the output is passed to the virtual machine identified by **-uuser**.
>
> For all other conditions, the output is passed to RSCS and the **-nnode** is used to build the tag.

For all these conditions, a blank default value is used.

-ttext A text string. Used with the **-uuser** and **-nnode** to build the tag for the punch output.

-fname Specifies the file name that is to be attached to the spooled punch output. The default is blank.

Files

/etc/qconfig
/usr/lpd/punkbend

Related Information

See the following commands: "print" in topic 1.1.326 and "rdrdaemon" in topic 1.1.361.

See also the **qconfig** file format in the *AIX Operating System Technical Reference*.

1.1.342 puttext

Purpose
Updates an output file that contains message/insert/help descriptions.

Syntax

```
           +------+           +-----------+
puttext ---¦      +-- infile --¦           +---¦
           +- -n -+           +- outfile -+
```

Description
The **puttext** command uses the message/insert/help descriptions in **infile** to change, delete and add message/insert/help text to **outfile** for a component. (For information about the format and contents of **infile**, see *AIX Operating System Programming Tools and Interfaces*.)

The **infile** parameter specifies the name of the file where the message/insert/help descriptions reside. See *AIX Operating System Programming Tools and Interfaces* for a discussion of the **gettext** output file parameters that describes the format and contents of this file.

The **outfile** parameter specifies the name of the output file. If you specify an **outfile** that does not exist, a new component file is created. If you specify an existing **outfile**, a copy of that file is renamed as a backup file. In this case, an old backup file will be deleted.

Note: For the new file to be accessed by the message support run-time services, the output file name must be in the format **xxxccc_EN.m**. If you do not specify **outfile**, the component ID is prefixed to **_EN.m** to form the output file name.

Flag

-n Causes **puttext** to assign available index numbers to the input descriptions. If you specify this flag, all the index number fields of the input file must be underscore characters or blanks.

Related Information

See the following commands: "gettext" in topic 1.1.188.

1.1.343 pwck, grpck

Purpose
Checks the password and group files for inconsistencies.

Syntax

```
        +- /etc/passwd -+
pwck ---¦               +---¦
        +---- file -----+

         +- /etc/group -+
grpck ---¦              +---¦
         +---- file ----+
```

Description
The **pwck** command scans the named **file** or the default file **/etc/passwd** and writes to standard output any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and existence of the login directory and optional program name.

The **grpck** command scans the named **file** or the default file **/etc/group** and writes to standard output any inconsistencies. The checks include validation of the number of fields, group name, group ID, and whether all login names appear in the password file. **grpck** writes to standard output any group entries that do not have login names.

Files

/etc/passwd	Password file; contains user IDs.
/etc/group	Group file; contains group IDs.

Related Information

See the following commands: "groups" in topic 1.1.194, "passwd, chfn, chsh" in topic 1.1.312, and "adduser, users" in topic 1.1.15.

See the discussion of passwords in *Managing the AIX Operating System*.

1.1.344 pwd

Purpose
Displays the path name of the working directory.

Syntax

```
pwd ---¦
```

Description
The **pwd** command writes to standard output the full path name of your current directory (from the root directory). All directories are separated by a / (slash). The root directory is represented by the first /, and the last directory named is your current directory.

Warning: The full path name output by **pwd** is constructed by following hard (non-symbolic) links from the current working directory back to the root directory. This may result in a different path name being output than might have been expected if symbolic links were involved in the original path taken by the user or application program when changing directories.

Related Information

See the following command: "cd" in topic 1.1.53.

See the **statx** system call in *AIX Operating System Technical Reference*.

1.1.345 pwgmap

Purpose
Maps password and group files into a table.

Syntax

pwgmap ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

When two disjoint networks are merged together, the password and group files between the two may need to be merged as well. The **pwgmap** command performs the first step of the transformation. It first prompts for the names of the files to be merged and copies their respective contents to ***passwd.data1***, ***passwd.data2***, ***group.data1,*** and ***group.data2*** in the current directory. While producing the mapping tables, the *uid/gid* values and **login/group** names are used to determine uniqueness. The tables produced, ***passwd.map***, and ***group.map***, have format:

```
password.map
                      old       old       new       new
table     changes     uid      login      uid      login
----------------------------------------------------------

group.map
                      old       old       new       new
table     changes     gid      group      gid      group
----------------------------------------------------------
```

After executing the **pwgmap** command, the system administrator should examine the mapping tables to see if the new values are acceptable. Entries are ordered alphabetically by **old login** or **old group**. The **changes** field allows one to see exactly what was modified and may contain entries **Name** and/or **uid**. If no changes occurred, **none** will be written instead. If while editing the map tables, the administrator discovers that he wants to ignore a particular entry, he may alter the **changes** field to **ignore**. However, it may be the case that two users or groups should actually be the same, but differ in either **uid/gid** or **login/group** name. In this situation, the administrator may simply want to fold one user/group into another by modifying the **changes**, **new uid/new gid**, and **new login/new group** fields (see example below). The **table** field gives a positional indicator where the entry resides. It may contain 1, 2, or both.

The algorithm used by the **pwgmap** command works as follows:

If a **login/uid** pair matches between password files, they are assumed to be identical entries. If two logins are the same, but the uids differ, the case is changed, a letter at a time for one of the names until a difference is found; if two uids are the same but the logins are different, one of the ids is altered by taking the largest unused id value and adding one to it. Groups are treated in the same manner.

The **pwgmap** command is for use in conjunction with the **pwgmerge** command and may only be executed by a system administrator with superuser authority.

Examples

The system administrator may see a table which looks something like:

1.

table	changes	old uid	old login	new uid	new login
1	none	345	bershad	345	bershad
1	none	123	marsh	123	marsh
2	Name	129	marsh	129	Marsh
2	none	567	nathan	567	nathan

If the system administrator determines that the two marsh's were actually the same person, then he should modify the table by altering the **changes** field to fold, the **new uid** field to 123, and the **new login** field to **marsh**. Every occurrence of uid 129 in the file system will be mapped to 123 by the **pwgmerge** program.

If however, it is decided that they were different people, then the administrator may want to simply alter the **new login** field to bmarsh if the user's name is Brian Marsh.

2. If the system administrator wishes, he can make two seemingly unrelated users map to the same person. Thus, by changing the entry for "nathan" to look like:

table	changes	old uid	old login	new uid	new login
2	fold	567	nathan	345	bershad

the user "nathan" will become "bershad" after the merge. This means that if user "nathan" was a member of any group, that name will now be replaced with "bershad" in the new group file. All files owned by uid 567 on the site represented by table 2, will be owned by uid 345.

Similarly, groups themselves may be folded into one another with group membership consisting of the union of all the previous users.

Related Information

See the following command: "pwgmerge" in topic 1.1.346.

See the **passwd** file in *AIX Operating System Technical Reference.*

1.1.346 pwgmerge

Purpose

Merges password and group files from mapping tables.

Syntax

pwgmerge new-password-file new-group-file filesystem-map-table ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **pwgmerge** command reads the map and data tables produced by the **pwgmap** command to produce merged password and group files. These files must reside in the directory in which **pwgmap** was executed and provide the basis for updating the file system's inodes with any new owner or group values. The new file parameters to **pwgmerge** are the path names where the new password and group files will be placed. The file system map parameter is user generated. It contains a list of file systems followed by either a 1 or a 2 to determine whether it is to be associated with the first or second password and group files entered to **pwgmap**. The file might contain:

```
/luke/dev/hd02  1
/luke/dev/hd03  2
/c3p0/dev/hd14  2
/c3p0/dev/hd12  1
/r2d2/dev/hd13  2
```

The disk inodes of each file system listed in the map file will be updated with the new uid/gid values. These file systems should not be mounted to prevent any conflicting updates from occurring. When invoking this program, the user should have superuser authority.

Since identical users on different cluster sites may have different home directories, it is the responsibility of the system administrator to move files appropriately. To assist in this process, **pwgmerge** produces a list of conflicting home directories (pw.conflicts). Additional conflicts may arise between two identical users having different group ids. These conflicts will be printed in the list as well. The actual values output in the new password file will come from the first parameter given in **pwgmap**. No list will be generated should any conflicts arise between passwords or shells. The values used for these will come from the first password file.

Before the merge, identical users on different sites, may not necessarily belong to the same groups in their respective **/etc/group** files. Group membership after the merge will be the union of all previous memberships. If groups and/or users have been folded (the "changes" field is "fold"), then before running this program, it is important that the "new gid", "new uid", "new group", and "new login" fields are modified to contain the login/group values to which they are being mapped. The **pwgmerge** command uses these "new" fields to determine which users and groups are to be considered identical during construction of the merged password and group files.

After creating and installing the new password file, the administrator should run the **mkpasswd** command to produce the **/etc/passwd.pag** and **/etc/passwd.dir** files.

Related Information

See the following command: "pwgmap" in topic 1.1.345 and "mkpasswd" in topic 1.1.272.

See the **passwd** file in *AIX Operating System Technical Reference.*

1.1.347 qdaemon, lp

Purpose
Schedules jobs enqued by the **print** command.

Syntax

```
qdaemon ---¦


AIX/370 options:

                +-------- -fl=55-fw=80 ---------+
/usr/lpd/lp ---¦ +---------------------------+ +--- file ---¦
                +-¦ -fl=value    -tag=text    +-+          ¦
                  ¦ -fw=valu     -ascii       ¦¦  +--------+
                 ¦¦ -skip=value  -fcb=name    ¦¦
                 ¦¦ -statusfile  -dest=dest   ¦¦
                 ¦¦ -class=class -spool=spool ¦¦
                 ¦¦ -form=form                ¦¦
                 ¦¦ -route=vmid               ¦¦
                 ¦¦ -translate=table_file     ¦¦
                 ¦¦ -trtable=table_name       ¦¦
                 ¦¦ -char=name1               ¦¦
                 ¦+---------------------------+¦
                 +-----------------------------+


-trtable = table.name

PS/2 options:

                +----- -fl=55-fw=80 -------+   +------------+
/usr/lpd/lp ---¦ +---------------------+ +---¦   one of   +--- file ---¦
                +-¦ -fl=value     -elite +-+   ¦ +--------+ ¦          ¦
                  ¦ -fw=valu      -plot  ¦¦    +-¦ -ibmpg +-+ +--------+
                 ¦¦ -indent=value -plp   ¦¦      ¦ -oki   ¦
                 ¦¦ -skip=value   -wp    ¦¦      +--------+
                 ¦¦ -statusfile   -wrap  ¦¦
                 ¦+---------------------+¦
                 +-----------------------+
```

Description
The **qdaemon** is a background process (usually started by the **rc** command file) that schedules printing jobs enqueued by **print**.

The **lp** command prints a **file** on its standard output in a form that is suitable for a line printer. The **lp** command is normally invoked by the **qdaemon** which directs the output from **lp** to the appropriate line printer or device.

Flags may be passed to **lp** in the following ways:

Flags specified in the **qconfig** structure are passed each time that **lp** is invoked.

Flags that are not recognized by the **print** command are assumed to be for **lp** and are passed to **lp** with the requested job.

Flags

The following flags are common to AIX/370 and PS/2:

-fl = **value**
Sets the form's length equal to **value**. The default length is 66 lines.

-fw = **value**
Sets the form's width equal to **value**. The default width is 80 columns. Lines that are wider than **value** are truncated. If you set **value** to 0, no truncation is performed.

-skip = **value**
Does not print the first **value** blank lines in the file.

-statusfile
Updates the status information in the status file that is open on file descriptor 3. The status information is passed from **qdaemon**.

The following flags are available with AIX/370 only:

-class = **class**
Sets the output class for the spool file to **class**.

-form = **form**
Specifies which form should be used to print this spool file.

-route = **vmid**
Routes the output for the virtual machine **vmid** instead of the printer.

-tag = **text**
Sets the tag to **text** for the output spool file.

-translate = **table_file**
Downloads an alternate translate table from the file **table_file** into the lp device before printing this file. The filename **table_file** is the full pathname of a translate file such as those generated by the **genxlt** program. These are often kept in the directory **/usr/lib/nls/nlout**. Specifying a translate table will temporarily override the default translate table in the lp device driver and the table specified in the **NLPRIN** command variable. The device driver will revert to the default table at the completion of this print request. The **-translate** flag cannot be used with the **-trtable** flag.

The format of the translate table is described in **genxlt** program in the *AIX Operating System Commands Reference*.

-trtable = **table_name**
Selects one of the device driver's built-in translation tables. The built-in tables are:

Name	**Description**
TN	TN print train (default).
3262	Table for 3262 printer.

ALT3262	Table for 3262 with only uppercase characters.
3820	Table for 3820 printer.

-char = **name1 [name2 [name3 [name4]]]**
Specifies the name of the character arrangement table used when printing a file. The name must be from one to four characters with a maximum of four names given per command.

-fcb = **name**
Controls the vertical spacing of output on a page. The name must be from one to four characters. This parameter applies to output sent to 3800 printers only.

-dest = **dest**
Specifies the destination name for spool files created on this virtual printer. The dest is one to eight character alphanumeric name your VM installation assigns to printers at VM system generation.

-spool = **spool**
Specifies the spool information for this print file. This option allows you to provide several VM CP spool options at once. This option will not check for any syntax errors in the CP SPOOL information.

-ascii Print in ASCII mode on a 3800 using PSF, the file is not translated to EBCDIC for printing.

The following flags are available with PS/2 only:

-elite Prints the text at 12 characters per inch instead of 10 characters per inch. This flag changes the default form width to 96 characters.

-ibmgp Specified an IBM Graphic Printer.

-indent = **value**
Indents the printed output the number of spaces specified with **value**.

-oki Specifies and OKIdata Model 92 or 93.

-plot Passes text directly to the printer without processing. This is useful when using the printer as a plotter. Normally, lines that contain backspaces and carriage return characters return characters are processed so that they print with minimum print head motion. The sequence **Esc-9** maps to half-line feeds.

-plp Sets and resets printer port parameters, if the printer is attached with a parallel interface.

-wp Sets the printer, if possible, to the Word Processing mode.

-wrap Wrap lines longer than the printer line length. The default truncates lines longer than the printer line length.

Related Information

See the following commands: "print" in topic 1.1.326 and "genxlt" in topic 1.1.185.

1.1.348 quiz

Purpose
Tests your knowledge.

Syntax

```
                     +------------+   +--------------------------+
/usr/games/quiz ---¦ +--------+ +---¦                          +---¦
                     +-¦ -ifile +-+   +- category1 --- category2 -+
                       ¦ -      ¦¦
                      ¦+--------+¦
                      +----------+
```

Note: This command does not have MBCS support.

Description

The **quiz** game gives associative knowledge tests on various selectable subjects. It asks about items chosen from **category1** and expects answers from **category2**. If you do not specify the categories, **quiz** gives instructions and lists the available categories.

The **quiz** game gives the correct answer whenever you press the **Enter** key by itself. The game ends when there are no more questions or when you press the interrupt key; **quiz** reports a score and exits.

Flags

-ifile Substitutes the named **file** for the standard index file.

Note: In the following syntax description, brackets are normally used to indicate that an item is optional; a bold-faced bracket or brace, however, should be entered as a literal part of the syntax. A vertical list of items indicates that one and only one must be chosen. The lines in **file** must have the following syntax:

```
line  = category [ :category] . .  .
category = alternate [  |alternate ] . . .
alternate = [primary]
primary = character
          [category]
          option
option = {category}
```

In an index file, the first category of each line must specify the name of an information file (the information file contains the names of files with quiz material). The remaining categories specify the order and contents of the data in each line of the information file. The quiz data in information files follows the same syntax. A \ (backslash) is an escape character which allows you to quote syntactically significant characters or to insert a new-line character (\n) into a line. When either a question or its answer is blank, **quiz** does not ask it. The construct **a|ab** does not work in an information file. Use **a{b}**.

-t Provides a tutorial. Repeats missed questions and introduces new material gradually.

Examples

1. To start a Latin-to-English quiz:

   ```
   /usr/games/quiz latin english
   ```

   The **quiz** command displays Latin words and waits for you to enter what they mean in English.

2. To start an English-to-Latin quiz:

   ```
   /usr/games/quiz english latin
   ```

3. To set up a Latin-English quiz, add the following line to the index file:

   ```
   /usr/games/lib/quiz/latin:latin:english
   ```

   This line specifies that the file "/usr/games/lib/quiz/latin" contains information about the categories "latin" and "english".

   You can add new categories to the standard index file, **/usr/games/lib/quiz/index**, or to an index file or your own. If you create your own index file, run the **quiz** command with the **-i file** flag to give it your list of quiz topics.

4. This is a sample information file:

   ```
   cor:heart
   sacerdos:priest{ess}
   quando:when|since|because
   optat:{{s}he  |it  } [desires|wishes]\
   |desire|wish
   alb[us|a|um]:white
   ```

   This information file contains Latin and English words. The : (colon) separates each Latin word from its English equivalent. Items enclosed in "{   }" (braces) are optional. A | (vertical bar) separates two items when entering either is correct. The [ ] (brackets) group items separated by vertical bars.

The first line accepts only the answer "heart" in response to the Latin word "cor". The second accepts either "priest" or "priestess" in response to "sacerdos". The third line accepts "when", "since", or "because" for "quando".

The \ (backslash) at the end of the fourth line indicates that this entry continues on the next line. In other words, the fourth and fifth lines together form one entry. This entry accepts any of the following in response to "optat":

```
she desires    it desires     desire
she wishes     it wishes      wish
he desires     desires
he wishes      wishes
```

If you start a Latin-to-English quiz, then the last line of this sample information file instructs **quiz** to ask you the meaning of "albus". If you start an English-to-Latin quiz, then **quiz** displays "white" and accepts "albus" "alba", or "album" for the answer.

If any of the characters "{", "}", [,], or | appear in a question item, then **quiz** gives the first alternative of every | group and displays every optional group. Thus, the English-to-Latin question for the fourth definition in this sample is "she desires".

Files

/usr/games/lib/quiz/index
/usr/games/lib/quiz/*

1.1.349 quot

Purpose
Summarizes file system ownership.

Syntax

```
             +--------+   +------------------+
/etc/quot ---¦ +----+ +---¦                  +---¦
             +-¦ -c +-+   +--- filesystem ---+
               ¦ -f ¦                      ¦
              ¦¦ -n ¦¦      +--------------+
              ¦+----+¦
              +------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **quot** command prints the number of 4K blocks in the named **filesystems** currently owned by each user. If no **filesystems** option is specified, all file systems listed in **/etc/filesystems** are assumed.

Flags

The following options are available:

-c	Prints three columns giving file size in blocks, number of files of that size, and cumulative total of blocks in that size or smaller file.
-f	Prints number of files and their cumulative size in blocks by user.
-n	Reads standard input for a list of inodes and reports the owner of the file corresponding to the inode in the named file system.

Examples

1. The following example pipes the output of **ncheck** (after sorting) to **quot -n**.

```
ncheck /dev/rhd01 | sort +0n | quot -n /dev/rhd01
```

Files

/etc/passwd To get user names.

/etc/filesystems Default file systems.

Related Information

See the following commands: "ls, lf, lr" in topic 1.1.252, "du" in topic 1.1.141, and "ncheck" in topic 1.1.286.

1.1.350 quota

Purpose
Displays disk usage and limits.

Syntax

```
                    +--------+
         +--------+ ¦        ¦
quota ---¦ +----+ +--- user ---¦
         +-¦ -q +-+
           ¦ -v ¦
          ¦+----+¦
          +------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **quota** command displays users' disk usage and limits. Only the superuser may use the optional **user** argument to view the limits of users other than himself.

The **quota** command reports only on file systems which have disk quotas. If the **quota** command exits with a non-zero status, one or more file systems are over quota.

Flags

-q	Prints a more terse message, containing only information on file systems where usage is over quota.
-v	Displays user's quotas on all file systems including those where no storage is allocated.

Related Information

See the following commands: "quotaon, quotaoff" in topic 1.1.352.

1.1.351 quotacheck

Purpose
File system quota consistency checker.

Syntax

```
                +--------+
quotacheck ---¦ +----+ +--- filesystem ---¦
                +-¦ -p +-+                ¦
                  ¦ -v ¦   +--------------+
                 ¦+----+¦
                 +------+

                +--------+
quotacheck ---¦ +----+ +--- -a ---¦
                +-¦ -p +-+
                  ¦ -  ¦¦
                 ¦+----+¦
                 +------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **quotacheck** command examines each file system, builds a table of current disk usage, and compares this table against that stored in the disk quota file for the file system. If any inconsistencies are detected, both the quota file and the current system copy of the incorrect quotas are updated (the latter occurs only if an active file system is checked). Only superusers can view quotas on file systems they do not own.

The **quotacheck** command expects each file system to be checked to have a quota file named **quotas** in its root directory. If none is present, **quotacheck** ignores the file system.

In calculating the actual disk usage for each user, the **quotacheck** command accesses the raw device. Therefore, the file systems checked should be quiescent while **quotacheck** is running.

Normally the **quotacheck** command reports only those quotas modified. If the **-v** flag is specified, **quotacheck** indicates the calculated disk usage for each user on a particular file system.

Flags

-a	Reports on all file systems in **/etc/filesystems** that are set to **quotas=on**.
-p	Allows file systems on different drives to be specified, as in the following example:
	`quotacheck -p /dev/hd1 - -p /dev/hd7`
-v	Provides verbose output.

Files

/etc/dfsck	Dual file systems check.
/etc/filesystems	Default file systems.

Related Information

See the following commands: "quota" in topic 1.1.350, "quotaon, quotaoff" in topic 1.1.352, and "fsck, dfsck" in topic 1.1.177.

See "setquota" subroutine in the *AIX Operating System Technical Reference*.

1.1.352 quotaon, quotaoff

Purpose
Turns file system quotas on or off.

Syntax

```
           +------+   +--- filesystem ---+
quotaon ---¦      +---¦                  ¦ +---¦
           +- -v -+   ¦ +--------------+ ¦
                      +------- -a -------+

            +------+   +--- filesystem ---+
quotaoff ---¦      +---¦                  ¦ +---¦
            +- -v -+   ¦ +--------------+ ¦
                       +------- -a -------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **quotaon** command announces to the system that disc quotas should be enabled on one or more file systems. The file systems specified must have entries in **/etc/filesystems** and be mounted at that time. The file system quota files must be present in the root directory of the specified file system and be named **quotas**. The optional argument **-v** causes **quotaon** to print a message for each file system where quotas are turned on. If, instead of a list of file systems, a **-a** argument is give to **quotaon**, all file systems in **/etc/filesystems** marked read-write with quotas will have their quotas turned on. This is normally used at boot time to enable quotas, and is usually preceeded by running of **quotacheck**.

The **quotaoff** command announces to the system that file systems specified should have any disc quotas turned off. As above, the **-v** forces a verbose message for each file system affected; and the **-a** option forces all file systems in /etc/filesystems to have their quotas disabled.

These commands update the status field of devices located in **/etc/mtab** to indicate when quotas are on or off for each file system.

Files

/etc/mtab	Mount table.
/etc/filesystems	File system table (default filesystems).

Related Information

See the following commands: "quota" in topic 1.1.350, "quotacheck" in topic 1.1.351, and "fsck, dfsck" in topic 1.1.177.

See "setquota" subroutine in the *AIX Operating System Technical Reference*.

1.1.353 ranlib

Purpose
Converts archives to random libraries.

Syntax

```
         +------+
ranlib ---¦      +--- archive ---¦
         +- -t -+
```

Description

This command is included for compatibility only. The functionality is in **ar**. The **ranlib** command converts each **archive** to a form which the loader can load more rapidly. Sufficient temporary file space must be available in the file system which contains the current directory.

If given the **-t** option, ranlib only "touches" the archives and does not modify them. This is useful after copying an archive or using the **-t** option of **make** in order to avoid having **ld** complain about an "out of date" symbol table.

The **ranlib** command is a shell script which invokes **ar** with the appropriate flags.

Flags

-t   "Touch" the named archives without modifying them.

Related Information

See the following commands: "ld" in topic 1.1.226, "ar" in topic 1.1.23, "lorder" in topic 1.1.245, and "make" in topic 1.1.254.

1.1.354 rc

Purpose
Initializes normal system startup.

Syntax

```
              ¦
/etc/rc ---¦
```

```
----------------
¦ This command is not usually run from the command line.
```

Description
When the **init** process starts up the system in normal operating mode, it runs the command file **/etc/rc** to perform the necessary system initialization, including the enabling of various loggers.

The contents of **/etc/rc** may be installation specific, but there are a few things that it should do:

Remake the message of the day file
Check the default file systems (run **fsck**).
Mount the default file systems (run **mount**).
Purge temporary files
Set printer defaults

If all of the necessary operations complete successfully, the file exits with a zero return code that allows **init** to start loggers to complete normal initialization and startup.

Notes:

1. The mail facility is started by **rc** indirectly when it runs **/etc/rc.sendmail**.

2. The root file system is implicitly mounted.

Files

/etc/rc.tcpip	Performs functions required to start TCPIP.
/etc/rc.sendmail	Performs functions required by **sendmail**.
/etc/Single2multi	Takes system from single to multi-user mode.

Related Information

See the following commands: "cron" in topic 1.1.97, "fsck, dfsck" in topic 1.1.177, "init, telinit" in topic 1.1.208, "mount" in topic 1.1.278, and "pstart, penable, pshare, pdelay" in topic 1.1.338.

See the discussion of starting up the system in *Managing the AIX Operating System*.

1.1.355 rcvdist

Purpose
Sends a copy of incoming messages to additional recipients.

Syntax

```
           +------------------+
rcvdist ---¦                  +--- user -----¦
           +- -form formfile -+         ¦ ¦
         ¦                      +--------+ ¦
         +------------- -help -------------+
```

Note: This command does not have MBCS support.

Description

The **rcvdist** command is used to forward copies of incoming messages to other users. **rcvdist** is not designed to be run directly by the user; it is designed to be called by **/usr/lib/mh/slocal**. The **rcvdist** command is part of the Message Handling (MH) package.

The **rcvdist** command sends a copy of the incoming message to the specified users. **rcvdist** uses the format string facility described in **mh-format**. You can run **rcvdist** on all incoming messages by specifying the **rcvdist** command in the **.maildelivery** file.

Flags

-help	Displays help information for the command.
-form formfile	Uses the form contained in the **formfile** for the form of the reply. **rcvdist** treats each line in **formfile** as a format string.

Files

$HOME/.maildelivery	The user's local mail delivery instructions.
$HOME/.forward	The user's default message filter.

Related Information

See the following commands: "ali" in topic 1.1.17, "rcvpack" in topic 1.1.356, "rcvstore" in topic 1.1.357, "rcvtty" in topic 1.1.358, "sendmail, mailq, newaliases" in topic 1.1.417, "slocal" in topic 1.1.430, and "whom" in topic 1.1.539.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*. "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.356 rcvpack

Purpose

Saves incoming messages in a packed file.

Syntax

```
           +- file --+
rcvpack ---¦         +---¦
           +- -help -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rcvpack** command is used to place incoming messages in a packed file. **rcvpack** is not designed to be run directly by the user; it is designed to be called by **/usr/lib/mh/slocal**. The **rcvpack** command is part of the Message Handling (MH) package.

The **rcvpack** command appends a copy of the incoming message to the specified file and runs the **packf** command on the file. You can run **rcvpack** on all incoming messages by specifying the **rcvpack** command in the **.maildelivery** file.

Flag

-help	Displays help information for the command.

Files

$HOME/.mail delivery	The user's local maildelivery instructions.
$HOME/.forward	The user's default message filter.

Related Information

See the following commands: "packf" in topic 1.1.310, "inc" in topic 1.1.206, "rcvdist" in topic 1.1.355, "rcvstore" in topic 1.1.357, "rcvtty" in topic 1.1.358, "sendmail, mailq, newaliases" in topic 1.1.417, and "slocal" in topic 1.1.430.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*. "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.357 rcvstore

Purpose
Incorporates new mail from standard input into a folder.

Syntax

```
             +- +inbox --+   +--- -create ---+   +------------------+
rcvstore ---¦            +---¦    one of     +---¦                  +---
             +- +folder -+   ¦ +-----------+ ¦   +- -sequence name -+
                             +-¦ -create   +-+
                               ¦ -nocreate ¦
                               +-----------+

    +---------------+   +-- -nozero --+
 ---¦    one of     +---¦   one of    +---¦
    ¦ +-----------+ ¦   ¦ +---------+ ¦
    +-¦ -public   +-+   +-¦ -zero   +-+
      ¦ -nopublic ¦       ¦ -nozero ¦
      +-----------+       +---------+

rcvstore --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rcvstore** command is used to incorporate incoming messages. **rcvstore** is not designed to be run directly by the user; it is designed to be called by **/usr/lib/mh/slocal**. The **rcvstore** command is part of the Message Handling (MH) package.

The **rcvstore** command accepts messages from standard input and places them in a specified folder. You can run **rcvstore** on all incoming messages by specifying the **rcvstore** command in the **.maildelivery** file.

You can specify **rcvstore** flags in **$HOME/.maildelivery** or as with most MH commands, in **$HOME/.mh_profile**.

Flags

-create	Creates the specified folder in your mail directory if the folder does not exist.
+*folder*	Places the incorporated messages in the specified folder. The default is **+inbox**.
-help	Displays help information for the command.
-nocreate	Does not create the specified folder if the folder does not exist.
-nopublic	Restricts the specified sequence to your usage. **-nopublic** does not restrict the messages in the sequence, only the sequence. This flag is the default if the folder is write-protected from other users.
-nozero	Appends the messages incorporated by **rcvstore** to the

	specified sequence (see the **-zero** flag).
-public	Makes the specified sequence available to other users. **-public** does not make protected messages available, only the sequence itself. This flag is the default if the folder is not write-protected from other users.
-sequence *name*	Adds the incorporated messages to the specified sequence.
-zero	Clears the specified sequence before placing the incorporated messages into the sequence. This flag is the default (see the **-nozero** flag).

Profile Entries

Folder-Protect:	Sets the protection level for your new folder directories.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your **user_mh_directory**.
Unseen-Sequence:	Specifies the sequences used to keep track of your unseen messages.
Rcvstore:	Specifies flags for the **rcvstore** program.

Files

$HOME/.mh_profile	The **mhl** user profile.
$HOME/.maildelivery	The user's local mail delivery instructions.
$HOME/.forward	The user's default message filter.

Related Information
The following commands: "inc" in topic 1.1.206, "rcvdist" in topic 1.1.355, "rcvpack" in topic 1.1.356, "rcvtty" in topic 1.1.358, "sendmail, mailq, newaliases" in topic 1.1.417, and "slocal" in topic 1.1.430.

The **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*. The *"Overview of the Message Handling Package"* in *Managing the AIX Operating System*.

1.1.358 rcvtty

Purpose
Notifies the user of incoming messages.

Syntax

```
          +-- cmd --+
rcvtty ---¦         +---¦
          +- -help -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rcvtty** command is used to send the user a message when incoming mail has arrived. **rcvtty** is not designed to be run directly by the user; it is designed to be called by **/usr/lib/mh/slocal**. The **rcvtty** command is part of the Message Handling (MH) package.

The **rcvtty** command sends a one-line scan listing to your terminal. If you give **rcvtty** a command as an argument, **rcvtty** executes the command with the incoming message as the command's standard input, and sends the output to the terminal. For **rcvtty** to write output to your terminal, your terminal's write permission must be set as all.

You can run **rcvtty** on all incoming messages by specifying the **rcvtty** command in the **.mail delivery** file.

Flag

-help	Displays help information for the command.

Files

$HOME/.maildelivery	The user's local mail delivery instructions.
$HOME/.forward	The user's default message filter.

Related Information

See the following commands: "rcvdist" in topic 1.1.355, "rcvpack" in topic 1.1.356, "rcvstore" in topic 1.1.357, "rcvtty," "sendmail, mailq, newaliases" in topic 1.1.417 , and "slocal" in topic 1.1.430.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*. The *"Overview of the Message Handling Package"* in *Managing the AIX Operating System*.

1.1.359 rdevcvt

Purpose
Changes the TCF site number specification of all of the device special files in a specified file system.

Syntax

```
                      +- old-site-number -+
rdevcvt --- device ---¦                   +--- new-site-number ---¦
                      +-------- - --------+
```

Description

The **rdevcvt** command searches the file system stored on a device for device special files. For each file it finds, **rdevcvt** changes the site specification in that file to refer to the **new-site-number**, provided that the special file had previously referred to the **old-site-number**. If the **old-site-number** is specified as **-**, all special files found are converted except those which referred to site number 0 (using site devices).

The **rdevcvt** command is usually run only by the **chparm** command to modify the device inodes in the **/dev** directory of the **<LOCAL>** file system. Since it is often necessary to run this command on a mounted file system, extreme care must be given to make sure that the system is quiescent while this command is running and that no pipes are in use. It is recommended that this command not be run on a mounted file system except when run by the **chparm** command.

Related Information

See the following commands: "mknod" in topic 1.1.271 and "chparm" in topic 1.1.69.

See the **inode** and **fs** file formats in *AIX Operating System Technical Reference*.

1.1.360 rdf

Purpose
Reports free space information on replicated file systems.

Syntax

```
rdf---------------------¦
       ¦ -c          ¦ ¦
     ¦ ¦ -f          ¦ ¦
     ¦ ¦ -h          ¦ ¦
     ¦ ¦ -n          ¦ ¦
     ¦ ¦ -L          ¦ ¦
     ¦ ¦ device      ¦ ¦¦
     ¦ ¦ filename    ¦ ¦
     ¦ ¦ gfs_number  ¦ ¦
     ¦ +------------+ ¦
     +----------------+
```

```
-----------------
¦ The default action is to provide information for each
  file system which is currently mounted on any site in
  the current TCF cluster.
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rdf** command writes to standard output information about total space and available space on the specified file systems. If the file system is a replicated file system, information is displayed about each mounted copy of the file system and additional replicated file system information is also displayed. The flag arguments restrict which file systems or which copies of replicated file systems are considered. They also adjust what information is displayed about each file system.

A file system may be specified by a device name (**device**), by the name of a directory or regular file (**filename**) inside a mounted file system, or by the global file system number (**gfs_number**) of a file system.

Each copy of a replicated file system maintains a separate allocation of free blocks to be assigned to files. The **rdf** command can be used to monitor the free space in each copy of a replicated file system. Each file system copy may run out of free space at different times, for a number of reasons, so **rdf** should be used to monitor when file systems run out of space and whether files have successfully propagated from the primary copy of the replicated file system to the other copies.

Non-primary copies of replicated file systems may run out of space for the following reasons:

1. The file system copy has a smaller total allocation of file system blocks. This usually only applies to secondary file system copies.

2. A new version of a program was installed in the primary copy of the file system, but a process on another site was running the copy stored there. The disk space for the old version will not be freed until the

process terminates or the system reboots and **fsck** is run.

3. The use of shadow pages by the file system requires that new pages be allocated to a propagated file before the old pages of the file can be freed. AIX requires that the free space in a file system is as large as the largest file in the file system.

4. A **clri** command was run on an inode in the primary copy of a replicated file system. This deletion will not automatically propagate to the other file system copies. If **clri** is not run on the other copies of the file system, the disk space will remain in use until the inode is used for a new file.

When a non-primary copy of a file system is full, the recommended procedure is to delete some files in the file system or use the **chfstore** or **store** command to discontinue the storing of some files in the full copy of the file system.

The fields displayed by **rdf** for each file system are:

GFS	The global file system of the file system. This number is the same for all copies of a replicated file system.
PK	The global file system pack number of this copy of a replicated file system. Each copy of a replicated file system has a different pack number. If the file system is not replicated, the pack number is always 1.
SITE	The cluster site name where the file system is currently mounted.
FL	File system flags. These flags identify the type of file system.

-	Not replicated.
SP	System Replicated, Primary Copy.
SB	System Replicated, Backbone Copy.
S-	System Replicated, Secondary Copy.
UP	User Replicated, Primary Copy.
UB	User Replicated, Backbone Copy.
U-	User Replicated, Secondary Copy.

TOTAL	Total number of 1024-byte blocks in the file system.
USED	Number of 1024-byte blocks in the file system which are currently allocated to a file.
FREE	Number of 1024-byte blocks in the file system which are currently available.
%USED	Percentage of the total number of blocks which are currently allocated to files.
LWM	Low water mark for this copy of a file system. The LWM will always equal the HWM on the primary copy of a file system. On a non-primary copy, the LWM value indicates which file changes (creations, modifications, deletions) have been successfully propagated from the primary copy to this non-primary copy of the file system. Each file system change is assigned a number, and this number defines the order in

which changes are propagated. If the LWM value of one copy equals the HWM value of the primary copy, this copy has successfully propagated all changes made to files in this file system that this site is supposed to store. Non-replicated file systems do not have a LWM value.

HWM High water mark for this copy of a file system. This indicates the highest numbered change that has occurred in this file system. Except for very brief moments, all copies of a replicated file system should have the same value for HWM. If the file system is not replicated, the file system does not have a HWM value.

DEVICE The name of the device for the mounted file system. This device usually resides in the **/dev** directory on the site where this file system copy is mounted.

DIR The directory on which a file system is mounted. All copies of a replicated file system must be mounted on the same directory.

Flags

-c Display information on only the current synchronization site (CSS) for the specified file systems. The CSS for a replicated file system is one of the mounted copies which keeps track of which files are open and what file record locks are held. If the primary copy of a file system is mounted, this site will be the CSS.

-f Fast option. Suppresses the DEVICE and DIR columns. The DEVICE and DIR columns are determined by reading the **/etc/mtab** files on each site in the cluster. The **-f** option can often speed up the command by only providing the other output fields.

-h Suppress header line. This option displays information without a header line and thus makes the output more easily analyzed by programs such as **awk**.

-n Display site number rather than site name.

-L Only display information about file systems or file system copies mounted on the local cluster site.

Examples

To list information about the root file system:

```
  $ rdf /

  GFS PK   SITE  FL  TOTAL  UUSED   FREE %USED    LWM     HWM   DEVICE DIR
    1  1  fafnir SP 115000 115000   4780   96%  654076  654076  diskroot /
    1  3   atlas S-  25000  22820   2180   91%  654076  654076       hd2 /
    1  4  swords S-  25000  22820   2180   91%  654076  654076       hd2 /
    1  7   coins S-  25000  22820   2180   91%  654076  654076       hd2 /
    1 13    hath SB 115000 110220   4780   96%  654076  654076  diskroot /
    1 14  carmen SB 115000 110220   4780   96%  654076  654076  diskroot /
```

Files

/etc/mtab	Lists the currently mounted file systems on each cluster site.

Related Information

See the following commands: "df" in topic 1.1.121, "fsck, dfsck" in topic 1.1.177, "where" in topic 1.1.534, "chfstore" in topic 1.1.64, and "store" in topic 1.1.443.

See the **mtab** and **fs** descriptions in the *AIX Operating System Technical Reference.*

1.1.361 rdrdaemon

Purpose

Handles (file transfer) spool files in the reader queue.

Syntax

/etc/rdrdaemon ---¦

Note: This command is for the System/370 only.

Description

The **rdrdaemon** is a background process usually started via **/etc/inittab** that handles queued files for reader (File Transfer). It checks the file control information; if the file is not in PUN format or if inconsistencies are detected, the file is put in USER HOLD status for disposition by the AIX/370 System Administrator. It also checks that the receiving userid is a valid one for the receiving node (subsystem). If the userid is not valid, the file is returned to the sender or placed in USER HOLD status for disposition by you. In either of these situations a mail message will be sent to the systems administrator to inform him that these files were placed in user hold.

If the file checks out OK, it is placed in the user's **$HOME/netfile** directory using the spool fileid as the file name. If a file with this name already exists in the directory, a new file name is generated by appending the spool fileid with a . and a number (00, 01, 02, and so on) to generate a unique name. If the users home directory is unavailable, the file is placed in a tempory hold state and retried every thirty minutes.

If a file in the CP spool queue is a note sent from a CMS user, the filetype is Note or note. The **rdrdaemon** calls the server **/usr/bin/rscssrvr** to translate the file into mail format and call **sendmail** to deliver the file in the user's incoming mailbox. If the filetype is not NOTE or note, **rdrdaemon** puts the file in the $HOME/netfile and notifies the user of this incoming file via **sendmail.**

The following information from CP Q RDR and from the tag is saved in a header line for each file stored in **$HOME/netfile**:

The nodeid of the originating system

The originating userid (either the userid or the VM-ID depending o the conditions).

The data, time, and file name from CP Q RDR

The file is HELD in the **netfile** directory until the receiving user has initiated action to purge it.

Files

/dev/rdr	Reader.
$HOME/netfile/*	Files to be received.
/etc/init.dir/Singl2multi	Shell script starting the daemon.
/etc/system.netid	

Related Information

See the following command: "rscssrvr" in topic 1.1.397.

1.1.362 rdump

Purpose
File system dump across the network.

Syntax

```
              +---------------------------+
/etc/rdump ---¦       +------------------+ +--- filesystem ---¦
              +- key -¦                  +-+
                      +- machine:device -+
```

Description

The **rdump** command copies to magnetic tape or specified device all files changed after a certain date in the **filesystem**. The command is identical in operation to **dumpbsd** except the "f" should be specified in the **key** and the device supplied should be of the form **machine:device**.

The **rdump** command creates a remote server, **/etc/rmt**, on the client machine to access the tape device.

You must have read permissions for all files to be copied, or be the superuser.

Related Information

See the following commands: "dumpbsd" in topic 1.1.143 and "rmt" in topic 1.1.381.

1.1.363 reboot

Purpose
Restarts the machine.

Syntax

```
          +--------+
reboot ---¦ +----+ +---¦
          +-¦ -l +-+
            ¦ -n ¦
           ¦¦ -q ¦¦
           ¦+----+¦
           +------+
```

Description

The **reboot** command starts by placing the kernel in memory at location zero and transferring to the entry point. Since the system is not re-enterable, it is necessary to read the kernel in from disk or tape each time it is to be loaded.

Rebooting a running system

When the system is running and multiple users are logged in, **shutdown** is normally used to perform a reboot. If there are no users, **reboot** can be used.

The **reboot** command causes the disks to be synced and allows the system to perform other shutdown activities such as resynchronizing the hardware time-of-day clock. A reboot is then started, as described below. By default, the system boots and the disks are automatically checked. If all this succeeds without incident, the system comes up for multiple users.

Power fail and crash recovery

For a 370:

To force a reboot, type "**PA1** CP IPL ### CLEAR" where ### is the virtual address of the system's residence disk. If you are set up for auto-IPL, you can log out the virtual machine, and then log back in.

For a PS/2:

Normally, the system will reboot itself at power-up or after crashes. To force a reboot, type "**Ctrl-Alt-7**" at the PS/2 console. This will reboot the machine without unmounting file systems and without syncing the disks.

Boot Procedure

For a 370:

After the IPL command, the disk is scanned for a bootstrap program. This program loads a more complex boot program in an AIX file system on the disk. This boot program then loads the kernel debugger, and depending on the specified boot parameter, it either boots unix.std from that file system, or prompts the operator for the name of the AIX kernel.

+--+

boot parameter	**kernel**	**initial state**	**initial debugger entry**
0	unix.std	multi-user	no
1	ask name	multi-user	yes
2	unix.std	single-user	yes
3	ask name	single-user	yes

For a PS/2:

Once the PS/2 resets and completes a self-test program, the fixed disks are scanned for a bootable AIX partition. The boot program is loaded, which will begin to boot unix.std, but may be interrupted by pressing any key. After such an interruption, the boot program will provide the devices. The menus may be used to boot alternate kernels and to perform system maintenance. Refer to the Installing and Customizing AIX for further information.

Flags

-l	Do not log the reboot or place a shutdown record in the accounting file. The **-n** and **-q** options imply **-l**.
-n	Do not perform the sync. Use this option in case of catastrophe.
-q	Reboot quickly and ungracefully, without shutting down running processes first.

The **reboot** command normally logs the reboot using **syslog** (8) and places a shutdown record in the login accounting file **/usr/adm/wtmp**. These actions are inhibited if the **-l, -n,** or **-q** options are present.

Files

/unix	System code.
/boot	System bootstap.

Related Information

See the following commands: "crash" in topic 1.1.96, "fsck, dfsck" in topic 1.1.177, "halt" in topic 1.1.195, "init, telinit" in topic 1.1.208, "minidisks" in topic 1.1.266, "rc" in topic 1.1.354, "shutdown" in topic 1.1.425, and "syslogd" in topic 1.1.457.

1.1.364 recmstr

Purpose
Recovery daemon.

Syntax

```
recmstr ---¦
```

Description

The **recmstr** command issues **raccept** system calls waiting to receive notification of a file system in need of user-level reconciliation. When it receives the notification, it causes **primrec** to be invoked with the necessary parameters.

This is not intended to be used by anyone other than the system administrator.

Related Information

See the following command: "primrec" in topic 1.1.325.

1.1.365 refer

Purpose
Finds and inserts literature references in documents.

Syntax

```
        +-----------------------------+
refer ---¦ +-------------+ +--------+  +---¦
        +-¦ -a    -n     +-¦        +--+
          ¦ -     -pbib  ¦¦+- file -+
         ¦¦ -c    -skeys ¦¦
         ¦¦ -e    -Bl.m  ¦¦
         ¦¦ -fn   -P     ¦¦
         ¦¦ -kx   -S     ¦¦
         ¦¦ -lm,n        ¦¦
         ¦+-------------+¦
         +---------------+
```

Note: This command does not have MBCS support.

Description

The **refer** command is a preprocessor for **nroff** or **troff**. that finds and formats references for footnotes or endnotes. It is also the base for a series of programs designed to index, search, sort, and print stand-alone bibliographies, or other data entered in the appropriate form.

Given an incomplete citation with sufficiently precise keywords, **refer** searches a bibliographic data base for references containing these keywords anywhere in the title, author, journal, etc. The input file (or standard input) is copied to standard output, except for lines between .[and .] delimiters, which are assumed to contain keywords, and are replaced by information from the bibliographic data base. The user may also search different data bases, override particular fields, or add new fields. The reference data, from whatever source, are assigned to a set of **troff** strings. Macro packages such as **ms** print the finished reference text from these strings. By default references are flagged by footnote numbers.

Flags

-an Reverse the first **n** author names (Jones, J. A. instead of J. A. Jones). If **n** is omitted, all author names are reversed.

-b Bare mode: do not put any flags in text (neither numbers nor labels).

-ckeys Capitalize the fields whose key-letters are in **keys**.

-e Instead of leaving the references where encountered, accumulate them until a sequence of the form

```
.[
$LIST$
.]
```

is encountered, and then write out all references collected so far. Collapse references to same source.

-fn	Set the footnote number to **n** instead of the default of 1 (one). With labels rather than numbers, this flag is a no-op.
-kx	Instead of numbering references, use labels as specified in a reference data line beginning **%%**; by default, **x** is **L**.
-lm,n	Instead of numbering references, use labels made from the senior author's last name and the year of publication. Only the first **m** letters of the last name and the last **n** digits of the date are used. If either **m** or **n** is omitted the entire name or date respectively is used.
-n	Do not search the default file **/usr/dict/papers/Ind**. If there is a **REFER** environment variable, the specified file will be searched instead of the default file; in this case the **-n** flag has no effect.
-pbib	Take the next argument **bib** as a file of references to be searched. The default file is searched last.
-skeys	Sort references by fields whose key-letters are in the **keys** string; permute reference numbers in text accordingly. Implies **-e**. The key-letters in **keys** may be followed by a number to indicate how many such fields are used, with **+** taken as a very large number. The default is **AD** which sorts on the senior author and then date; to sort, for example, on all authors and then title, use **-sA+T**.
-Bl.m	Bibliography mode. Take a file composed of records separated by blank lines, and turn them into **troff** input. Label **l** will be turned into the macro **m** with **l** defaulting to **%X** and **.m** defaulting to **.AP** (annotation paragraph).
-P	Place punctuation marks .,:;?! after the reference signal, rather than before. (Periods and commas used to be done with strings.)
-S	Produce references in the Natural or Social Science format.

To use your own references, put them in the format described below. They can be searched more rapidly by running **indxbib** on them before using **refer**; failure to index results in a linear search. When **refer** is used with the **eqn**, **neqn**, or **tbl** preprocessors, **refer** should be first, to minimize the volume of data passed through pipes.

The **refer** preprocessor and associated programs expect input from a file of references composed of records separated by blank lines. A record is a set of lines (fields), each containing one kind of information. Fields start on a line beginning with a "%," followed by a key-letter, then a blank, and finally the contents of the field, and continue until the next line starting with "%". The output ordering and formatting of fields is controlled by the macros specified for **nroff/troff** (for footnotes and endnotes) or **roffbib** (for stand-alone bibliographies). For a list of the most common key-letters and their corresponding fields, see **addbib**. An example of a **refer** entry is given below.

Examples

```
 %A    M. E. Lesk
 %T    Some Applications of Inverted Indexes on the \s-1UNIX\s0 System
```

```
  %B    UNIX Programmer's Manual
  %V    2b
  %I    Bell Laboratories
  %C    Murray Hill, NJ
  %D    1978
```

Files

/usr/dict/papers	Directory of default publication lists.
/usr/lib/refer	Directory of companion programs.

Related Information

See the following commands: "addbib" in topic 1.1.14, "sortbib" in topic 1.1.433, "roffbib" in topic 1.1.382, and "lookbib, indxbib" in topic 1.1.244.

1.1.366 refile

Purpose
Files messages in other folders.

Syntax

```
                one of
            +------------+
          +-¦ -draft     +----------------------------------------------------
          ¦ ¦ -file file ¦
refile ---¦ +------------+
          ¦ +----------------+   +-------- cur -------------------------------
          +-¦                +---¦ +----------- all --------------------------
            +- -src +folder -+   +-¦   +------------ sequence ------------+
                                   +---¦  one of                          +---
                                       ¦ +-------+   +-----------------+  ¦  ¦
                                     ¦ +-¦ num   +---¦     one of       +-+  ¦
                                     ¦   ¦ first ¦   ¦¦+--------------+ ¦    ¦
                                     ¦   ¦ prev  ¦   +-¦ :num   -prev +-+    ¦
                                     ¦   ¦ cur   ¦     ¦ :+num  -cur  ¦      ¦
                                     ¦   ¦ .     ¦     ¦ :-num  -.    ¦      ¦
                                     ¦   ¦ next  ¦     ¦ -num   -next ¦      ¦
                                     ¦   ¦ last  ¦     ¦ -first -last ¦      ¦
                                     ¦   +-------+     +--------------+      ¦
                                     +---------------------------------------+

    +-- -nolink --+   +-- -nopreserve --+
 ---¦   one of    +---¦     one of      +--- +folder ---¦
    ¦ +---------+ ¦   ¦ +-------------+ ¦             ¦
    +-¦ -link   +-+   +-¦ -preserve   +-+ +-----------+
      ¦ -nolink ¦       ¦ -nopreserve ¦
      +---------+       +-------------+

refile --- -help ---¦


----------------
¦ Do not put a blank between these items.
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **refile** command is used to copy and move messages to other files. The **refile** command is part of the Message Handling (MH) package and can be used with MH commands.

The **refile** command copies messages or moves messages from one folder to another folder. If a destination folder does not exist, **refile** asks if it should create the folder.

Flags

-draft	Copies the current draft message from your mail directory.
-file *file*	Copies the specified file. The file must be in valid

message format. (Use the **inc** command to incorporate new messages and to format them correctly.)

+*folder*	Copies the messages to the specified folder. Any number of folders can be specified.
-help	Displays help information for the command.
-link	Leaves the messages in the source folder or file after they are copied.
-nolink	Removes the messages from the source folder or file after they are copied. This flag is the default.
-nopreserve	Renumbers the messages that are copied. Renumbering begins with the number that is one higher than the last message in the destination folder. This is the default.
-preserve	Preserves the message numbers of the messages that are copied. If messages with those numbers already exist, **refile** issues an error message and does not alter the contents of the folders.
-src +*folder msgs*	Specifies the messages to be copied. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default message is the current message in the current folder. If a folder is specified, it becomes the current folder. If the **-link** flag and **all** are used, the current message does not change. Otherwise, if a message is specified, that message becomes the current message.

Profile Entries

Current-Folder:	Sets your default current folder.
Folder-Protect:	Sets the protection level for your new folder directories.
Path:	Specifies your **user_mh_directory**.
rmmproc:	Specifies the program used to remove messages from a folder.

Files

$HOME/.mh_profile	The MH user profile.

Related Information

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.367 regcmp

Purpose
Compiles patterns.

Syntax

```
          +-----+
regcmp ---¦     +-- file --¦
          +- - -+          ¦
                  +------+
```

Description
The **regcmp** command compiles the pattern in **file**, placing its output in **file.i**.

In most cases, **regcmp** makes unnecessary the use of the **regcmp** system call in your C programs, saving execution time and program size. The output of **regcmp** is C source code. Make each file entry a C variable name, followed by one or more blanks, followed by a pattern enclosed in double quotation marks (" "). Compiled patterns are initialized **char** declarations. Thus, **file.i** can be **included** in C programs, and **file.c** can be a file parameter to the **cc** command. The C program that uses **regcmp** output should use the **regex** subroutine to apply it to a string. (See **regcmp** and **regex** in *AIX Operating System Technical Reference*.)

Flag

- Places the output in **file.c**

Related Information

See the **regcmp** subroutine in *AIX Operating System Technical Reference*.

1.1.368 renice

Purpose

Alters priority of running processes.

Syntax

```
                         +------+
                       +-¦      +--- pid ----+
                       ¦ +- -p -+          ¦ ¦
                       ¦          +--------+ ¦
                       ¦ +------+            ¦
renice --- priority ---+-¦      +--- pgrp ---+---¦
                       ¦ +- -q -+          ¦ ¦ ¦
                     ¦ ¦          +--------+ ¦ ¦
                     ¦ ¦ +------+            ¦ ¦
                     ¦ +-¦      +--- user ---+ ¦
                     ¦   +- -u -+          ¦   ¦
                     ¦            +--------+   ¦
                     +-------------------------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **renice** command alters the scheduling priority of one or more running processes. The **who** parameters are interpreted as process IDs, process group IDs, or user names. When you **renice** a process group it causes all processes in the process group to have their scheduling priority altered. When you **renice** a user it causes all processes owned by the user to have their scheduling priority altered. By default, the processes to be affected are specified by their process IDs. To force **who** parameters to be interpreted as process group IDs, a **-g** may be specified. To force the **who** parameters to be interpreted as user names, a **-u** may be given. Supplying the **-p** will reset **who** interpretation to be (the default) process IDs. For example,

```
    renice +1 987 -u daemon root -p 32
```

would change the priority of process IDs 987 and 32, and all processes owned by users daemon and root.

Users other than the superuser may only alter the priority of processes they own, and can only monotonically increase their "nice value" within the range 0 to 20. (This prevents overriding administrative fiats.) The superuser may alter the priority of any process and set the priority to any value in the range -20 to 20. Useful priorities are: 20 (the affected processes will run only when nothing else in the system wants to), 0 (the "base" scheduling priority), anything negative (to make things go very fast).

These values are mapped by the command to those actually used by the kernel.

Files

/etc/passwd To map user names to user IDs

Related Information

See the following command: "nice" in topic 1.1.296.

See "getpriority" "setpriority" in nice in *AIX Technical Reference*.

1.1.369 repl

Purpose
Replies to a message.

Syntax

```
        +-----------+   +------- cur -------+   +-- -noannotate --+
repl ---¦           +---¦      one of       +---¦     one of      +---
        +- +folder -+   ¦ +---------------+ ¦   ¦ +-------------+ ¦
                        +-¦ num       cur  +-+  +-¦ -annotate   +-+
                          ¦ sequence .     ¦      ¦ -noannotate ¦
                          ¦ first     next ¦      +-------------+
                          ¦ prev      last ¦
                          +---------------+

    +-------- -cc all --------+   +-- -noquery --+   +--------------------+
 ---¦    one of     one of    +---¦    one of    +---¦                    +---
    ¦    +-------+   +-----+   ¦   ¦ +----------+ ¦   +--- -fcc +folder ---+
    +---¦ -cc    +---¦ all +---+  +-¦ -query   +-+                        ¦
        ¦ -nocc      ¦ to  ¦ ¦      ¦ -noquery ¦        +----------------+
      ¦ +-------+    ¦ cc  ¦ ¦      +----------+
      ¦              ¦ me  ¦ ¦
      ¦              +-----+ ¦
      +---------------------+

    +-----------------------------------------------------------------+
 ---¦ +--- -nodraftfolder -------------------------------------------+ +---
    +-¦                  one of                                      +-+
      ¦ +-----------------------------------+   +------- new -------+ ¦
      +-¦ -draftfolder +folder -draftmessage +---¦      one of       +-+
        ¦ -draftfolder +folder               ¦   ¦ +---------------+ ¦
        ¦ -draftmessage                      ¦   +-¦ num            +-+
        +-----------------------------------+      ¦ sequence next ¦
                                                   ¦ first    last ¦
                                                   ¦ prev     new  ¦
                                                   ¦ cur           ¦
                                                   +---------------+

    +--------------+   +----------------+   +---- -format -----+   +-- -noin
 ---¦              +---¦     one of     +---¦      one of      +---¦     one
    +- -form file -+   ¦ +-------------+ ¦   ¦ +--------------+ ¦   ¦ +------
                       +-¦ -editor cmd +-+   +-¦ -format      +-+   +-¦ -inpl
                         ¦ -noedit     ¦       ¦ -noformat    ¦       ¦ -noin
                         +-------------+       ¦ -filter file ¦       +------
                                               +--------------+

    +- -width 72 --+   +--------------------------+
 ---¦              +---¦          one of          +---¦
    +- -width num -+   ¦ +-----------------------+ ¦
                       +-¦ -whatnowproc cmdstring +-+
                         ¦ -nowwhatnowproc       ¦
                         +-----------------------+

repl --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **repl** command is used to compose a reply to a message. **repl** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

By default, **repl** copies a message form to a new draft message and invokes an editor. You can then fill in the message header fields **To:** and **Subject:**, fill in or delete the other header fields (such as **cc:** and **Bcc:**), and add the body of the message. When you exit the editor, the **repl** command invokes the MH command **whatnow**. You can press **Enter** to see a list of the available **whatnow** subcommands. These subcommands enable you to continue editing the reply, list the reply, direct the disposition of the reply, or end the processing of the **repl** command. "whatnow" in topic 1.1.533 describes the subcommands.

You can specify the message that you want to reply to by using the **+folder msg** flag. If you do not specify a message, **repl** replies to the current message.

You can specify the format of the reply by using the **-form** flag. If you do not specify this flag, **repl** uses your default message format located in the file **user_mh_directory/replcomps**. If this file does not exist, **repl** uses the system default message format located in **/usr/lib/mh/replcomps**.

Note: The line of dashes or a blank line must be left between the header and the body of the message for the message to be identified when it is sent.

Flags

-annotate
Annotates the message being replied to with the lines:

```
Replied: date
Replied: addrs
```

The annotation appears in the original draft message so that you can maintain a complete list of activities associated with the original message.

If you do not actually send the reply using the immediate **repl** command, the **-annotate** flag may fail to provide annotation. The **-inplace** flag forces annotation to be done in place.

-cc *names*
Specifies the users who will be listed in the **cc:** field of the reply. You can specify the following for *names*: **all**, **to**, **cc**, and **me**. The default is **-cc all**.

-draftfolder +*folder*
Places the draft message in the specified folder. If you do not specify this flag, **repl** selects a default draft folder according to the information supplied in the MH profiles. You can define a default draft folder in **$HOME/.mh_profile**. If **-draftfolder +***folder* is followed by *msg*, *msg* represents the **-draftmessage** attribute.

-draftmessage *msg*	Specifies the draft message. You can use one of the following message references as *msg*: *num* *sequence* **fi** **prev** **cur** **.** **next** **last** **nev** The default draft message is **new**.
-editor *cmd*	Specifies that *cmd* is the initial editor for composing the reply. If you do not specify this flag, **repl** selects a default editor or suppresses the initial edit, according to the information supplied in the MH profiles. You can define a default initial editor in **$HOME/.mh_profile**.
-fcc +*folder*	Places a file copy of the reply in *folder*. If you do not specify this flag, **repl** will not produce a file copy.
-filter *file*	Reformats the message being replied to and places the reformatted message in the body of the reply. **-filter** uses the **mhl** command and the specified format file. If you do not specify this flag, **repl** will omit the original message from the reply. The **repl** command does not have a default filter file. Thus, if you specify **-filter**, you must also specify *file*.
+*folder msg*	Replies to the specified message in the specified folder. You can use one of the following message references as *msg*: *num* *sequence* **fi** **prev** **cur** **.** **next** **last** The default message is the current message in the current folder. If you specify a folder, that folder becomes the current folder. The message being replied to becomes the current message.
-form *file*	Uses the form contained in the specified file for the form of the reply. **repl** treats each line in *file* as a format string.
-format	Removes duplicate addresses from the fields **To:**, **cc:**, and **Bcc:** and standardizes these fields. **repl** also uses the the columns specified by the **-width** flag to determine the format of these fields. This flag is the default.
-help	Displays help information for the command.
-inplace	Forces annotation to be done in place in order to preserve links to the annotated message.
-noannotate	Does not annotate the message. This flag is the default.

-nocc *names*	Specifies the users who will not be listed in the **cc:** field of the reply. You can specify the following for *names*: **all**, **to**, **cc**, and **me**.
-nodraftfolder	Places the draft in the file **user_mh_directory/draft**.
-noedit	Suppresses the initial edit.
-noformat	Does not Remove duplicate addresses from the fields **To:**, **cc:**, and **Bcc:** or standardize these fields. **repl** also does not use the the columns specified by the **-width** flag to determine the format of these fields.
-noinplace	Does not perform annotation in place. This flag is the default.
-noquery	Automatically builds the **To:** and **cc:** fields. This flag is the default.
-nowhatnowproc	Does not invoke a program that guides you through the reply tasks. The **-nowhatnowproc** flag also prevents any edit from occurring.
-query	Builds the **To** and **cc:** fields by interactively asking you if you want each address that would normally be placed in these fields actually placed in these fields.
-whatnowproc *cmdstring*	Invokes *cmdstring* as the program to guide you through the reply tasks. See "whatnow" in topic 1.1.533 for information about the default whatnow program and its subcommands. **Note:** If you specify **whatnow** for *cmdstring*, **repl** invokes an internal **whatnow** procedure rather than a program with the file name **whatnow**.
-width *num*	Sets the width of the address fields. The default is 72 columns.

Profile Entries

Alternate-Mailboxes:	Specifies your mailboxes.
Current-Folder:	Sets your default current folder.
Draft-Folder:	Sets your default folder for drafts.
Editor:	Sets your default initial editor.
fileproc:	Specifies the program used to refile messages.
mhlproc:	Specifies the program used to filter the message for which you are creating a reply.
Msg-Protect:	Sets the protection level for your new message files.
Path:	Specifies your **user_mh_directory**.
whatnowproc:	Specifies the program used to prompt "What now?" questions.

Files

/usr/lib/mh/replcomps	The MH default reply template.
user_mh_directory/replcomps	The user's default reply template. (If it exists, it overrides the MH default reply template.)
$HOME/.mh_profile	The MH user profile.
user_mh_directory/draft	The draft file.

Related Information

Other MH commands: "ali" in topic 1.1.17, "anno" in topic 1.1.19, "comp" in topic 1.1.85, "dist" in topic 1.1.131, "forw" in topic 1.1.174, "mhl" in topic 1.1.263, "prompter" in topic 1.1.334, "send" in topic 1.1.416, "whatnow" in topic 1.1.533, "whom" in topic 1.1.539.

The **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

"Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.370 repquota

Purpose
Summarizes quotas for a file system.

Syntax

```
            +--------+
repquota ---¦ +----+ +--- filesystem ---¦
            +-¦ -a +-+
              ¦ -  ¦¦
             ¦+----+¦
             +------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **repquota** command displays a summary of the disk usage and quotas for the specified file systems. For each user, the current number of files and amount of space (in kilobytes) is displayed. In addition, any quotas created with the **edquota** command are displayed. Only superusers can view quotas which are not their own.

Flags

-a	Reports on all file systems in **/etc/filesystems** that are set to **quotas=on**.
-v	Provides verbose output.

Files

quotas	At the root of each file system with quotas.
/etc/filesystems	For file system names and locations.

Related Information

See the following commands: "quota" in topic 1.1.350, "quotaon, quotaoff" in topic 1.1.352, and "edquota" in topic 1.1.150.

1.1.371 restore

Purpose
Copies back files created by the **backup** command.

Syntax

```
                one of
                +----+   +---------------------+
/etc/restore ---¦ -t +---¦   +-------------+   +---¦
                ¦ -T ¦   +---¦ -v        -q +---+
                +----+       ¦ -f devic     ¦ ¦
                           ¦ ¦ -C num       ¦ ¦
                           ¦ +-------------+ ¦
                           +-----------------+
                       +---------------------+   +------------+
/etc/restore --- -x ---¦   +-------------+   +---¦            +---¦
                       +---¦ -d        -h +---+   +--- file ---+
                           ¦ -         -q ¦ ¦              ¦
                         ¦ ¦ -f device -s ¦ ¦       +--------+
                         ¦ ¦ -C num    -I ¦ ¦
                         ¦ +-------------+ ¦
                         +-----------------+
                one of
                +----+   +---------------------+
/etc/restore ---¦ -r +---¦   +-------------+   +--- filesystem ---¦
                ¦ -R ¦   +---¦ -v        -q +---+
                +----+       ¦ -f device -s ¦
                           ¦ ¦ -C num       ¦ ¦
                           ¦ +-------------+ ¦
                           +-----------------+
                       +---------------------+
/etc/restore --- -m ---¦   +-------------+   +--- filesystem ---¦
                       +---¦ -v        -q +---+
                           ¦ -f device -s ¦
                         ¦ ¦ -C num       ¦ ¦
                         ¦ +-------------+ ¦
                         +-----------------+
```

Description
The **restore** command reads files written by the **backup** command to a backup medium and restores them to a file system.

There are four ways to use the **restore** command:

- To display a table of contents for the backup **-T**) or to display label information (**-t**)
- To restore specified files **-x**)
- To restore an entire file system **-r**) or begin at an arbitrary volume number (**-R**)
- To restore an entire minidisk **-m**).

When you do not specify a restore device with the **-f** flag, the **restore** command reads files from a default device. For restore by name, **restore -x**, the system reads from the default device, **/dev/rfd0**. For restore by file system, **restore -r**, or restore by minidisk, **restore -m**, restore checks **/etc/filesystems** for a stanza that matches the file name you specified. If such a stanza exists and contains a **backupdev** entry, the system reads from the device specified by **backupdev**. Otherwise, the system reads from the default device, **/dev/rfd0**.

Notes:

1. If the file system you are restoring is mounted and is not the root filesystem, **restore** unmounts the filesystem before it performs an inode restore and then remounts the filesystem after completion. If an error occurs during restore, the filesystem remains unmounted.

2. Files must be restored using the same method by which they were backed up. For example, if a file system was backed up by minidisk, it must be restored by minidisk.

3. When more than one diskette is required, **restore** reads the one mounted, prompts the user for a new one, and waits for the user's response. After inserting the new diskette, press **Enter** to continue restoring files.

4. For internal tape backup units, use a 200 series minor device (for example, **/dev/rst204**) to restore a tape backed up with a 0 series (best for **backup**) minor device (for example, **/dev/rst0**). Using the 0 series minor device number can, under some circumstances, result in a "READ ERROR on sector ####, assuming zeros" message, but should not affect the data being restored.

Flags

-Cnum Specifies the number of blocks to read in a single input operation. If you do not specify this flag, **restore** selects a default value appropriate for the physical device you have selected. Larger values of **num** result in longer physical transfers from tape devices. **restore** always ignores the value of the **-C** flag when it reads a diskette; the input is always read in clusters that occupy a complete track.

-d Indicates that if **file** is a directory, all files in that directory should be restored. In this case, the name of each restored file is always its name as shown by **restore -T**, whether the backup was by name or by inode. The **file** names supplied need not be directories. Thus, for inode backups:

```
/etc/restore -x a/b/file.c
```

creates a file whose name is its inode number, while:

```
/etc/restore -xd a/b/file.c
```

creates a file named **a/b/file.c**. With this flag, **file** names can include pattern-matching characters, although you must quote these characters to prevent their expansion by the shell.

Use this flag only when you are restoring by individual file name (**-x**).

-fdevice Specifies the input device. Specify **device** as a file name (such as **/dev/rmt0**) to get input from the named device or specify **-** (minus) to get input from the standard output device. The **-** feature enables you to improve performance when restoring from streaming tape by piping the output of a **dd** command to the **restore** command (see example). The **restore** command recognizes a special syntax for the names of input files. If the device parameter is a range of names, for example **/dev/rfd0-3**, **restore** automatically goes from one drive in the range to the next. After using all of the specified drives, it stops and requests that another diskette be inserted.

-h Specifies that the access and modification times of restored files are to be set to the time of restoration. (The default action is to set the access and modification times to the file times on the backup medium.) If a restored file is an archive, the modification times in all the member headers are also set to the time of restoration. You can specify this flag only when you are restoring individually named files.

-I Used only with the **-x** flag. Operates exactly like **-x** except that file names are read from standard input.

-m Restores an entire minidisk as an exact image.

Note: You can use this flag only with minidisks that are at least as large as the original minidisk that was backed up. If the minidisk is larger than the original, the leftover space becomes unusable after restoring the minidisk. You can use **restore -t** to see how large a minidisk you need.

-q Specifies that the removable medium is ready to use. In this case, **restore** proceeds without prompting you to prepare the removable medium.

-r Restores an entire file system. Use this flag with inode backups only (see "backup" in topic 1.1.32). **filesystem** can be a device name (block or character special file) or a directory name that **restore** looks up in **/etc/filesystems**.

If you are restoring a **full** (level 0) **backup**, run the **mkfs** command to create an empty file system before doing the restore. If you are restoring an **incremental backup** at, say, level 2, run **mkfs**, restore the appropriate level 0 backup, then the level 1 backup, and finally the level 2 backup.

Warning: If you do not follow this procedure carefully, you can ruin an entire file system. As an added safety precaution, run **fsck** after you restore each backup level.

-R Restarts an aborted **restore** at a specified point. **restore** prompts you for the starting volume number. This flag is invalid in combination with the **-m** flag.

-s type "Flattens" hidden directory when restoring on a system of the given type. This flag is only used with the **-x** (restore file name) flag. A restore of x@/i370 using the **-s** i370 flag restores x@/i370 as x.

-T Displays the backup file header and the names of the backed up files. If the backup was made by name (**backup -i**), the names displayed are the ones you provided to **backup**. If the backup was made by inode, **restore** displays the i-number of each file along with the file name. The names are relative to the root directory of the file system backed up. The only exception is the root directory itself, whose name is given as a slash (/).

-t Displays only the backup file header.

-v Reports the progress of the restoration as it proceeds.

-x Restores individually named files. The names must be in the same form as the names shown by **restore -T**. With a name backup, **restore** gives the restored file whatever name was supplied when the file was backed up. If the original name was specified relative to the current directory, **restore** creates a file relative to the current directory. **restore** automatically creates any needed directories. With an inode backup, the name of the restored file is the same as its i-number. This flag is invalid with the **-m** flag.

Examples

1. To list the names of files previously backed up:

   ```
   /etc/restore  -T
   ```

   Information is read from the default backup device **/dev/rfd0**. If individual files were backed up, only the file names are displayed. If an entire file system was backed up, the i-number is also shown.

2. To display technical information about a backup:

   ```
   /etc/restore  -t
   ```

   This command displays information including when the backup was made, which file system was saved, and whether it is a backup by name, a backup by minidisk, or a backup by file system or inode.

3. To restore files to the main file system:

   ```
   /etc/restore  -x  -v
   ```

   The **-x** extracts all the files from the backup medium and restores them to their proper places in the file system. The **-v** displays a progress report as each file is restored. If a file system backup is being restored, the files are named with their i-numbers. Otherwise, just the names are displayed.

4. To copy selected files:

   ```
   /etc/restore  -xv  /u/tom/manual/chap1
   ```

   This command extracts the file **/u/tom/manual/chap1** from the backup medium and restores it. To work properly, **/u/tom/manual/chap1** must be a name that can be displayed by **restore -T**.

5. To copy all the files in a directory:

   ```
   /etc/restore  -xdv  manual
   ```

   This command restores the directory **manual** and the files in it. If it does not exist, a directory named **manual** is created in the current directory to hold the files being restored.

6. To restore an entire file system backup:

```
  mkfs  /dev/hd1
  /etc/restore  -rv  /dev/hd1
```

This command restores an entire file system backup onto **/dev/hd1**. It destroys and replaces any file system that was previously stored on **/dev/hd1**. If the backup was made using incremental file system backups, restore the backups in increasing backup-level order (0, 1, 2...).

7. To restore a minidisk:

```
  /etc/restore  -m  /dev/hd1
```

   This restores the exact image of minidisk **/dev/hd1**. You can also identify the minidisk by its stanza name in the **/etc/filesystems** file.

8. To improve performance on streaming tape, pipe the **dd** command to the **restore** command:

```
  dd if=/dev/rmt0 bs=30b |/etc/restore -xf-
```

   This command cannot be used on multi-volume backups.

   The **dd** command copies the files from an input file which is a streaming tape device (**if=/dev/rmt0**) and specifies a file size of thirty 512-byte blocks (**bs=30b**). The output is piped to **restore**. The **restore** command gets the input from the standard input device (**f-**) and restores up by name (**x**).

Files

/etc/filesystems	Descriptions of mountable file systems; consulted for default parameters.
/dev/rfd0	Default restore device.

Related Information

See the following commands: "backup" in topic 1.1.32 and "print" in topic 1.1.326.

See the discussion of **filesystems** and **backup** in *AIX Operating System Technical Reference*.

See "Backing up and Restoring Files" in *Using the AIX Operating System*.

1.1.372 restorebsd

Purpose
Reads tapes dumped with the **dumpbsd** command.

Syntax

```
                   +----------+   +------------------+   +--------+
/etc/restorebsd ---¦  one of  +---¦ +--------------+ +---¦        +---¦
                   ¦ +------+ ¦   +-¦ v            +-+   +- name -+ ¦
                   +-¦ r  + +-+     ¦ F input_file ¦¦  +------------+
                     ¦ R  i ¦      ¦¦ y            ¦¦
                     ¦ x    ¦      ¦¦ m            ¦¦
                     +------+      ¦¦ h            ¦¦
                                   ¦¦ b            ¦¦
                                   ¦¦ c            ¦¦
                                   ¦¦ d            ¦¦
                                   ¦¦ s            ¦¦
                                   ¦+--------------+¦
                                   +----------------+
```

Description

Note: The **restorebsd** command is not used to port information between AIX and BSD systems. The **tar** command is used for compatibility with BSD systems.

The **restorebsd** command reads the tape or specified device dumped with the **dumpbsd** command. Its actions are controlled by the flags described below. Other arguments to the command are file or directory names specifying the files that are to be restored. Unless the **h** flag is specified, the appearance of a directory name refers to the files and (recursively) subdirectories of that directory.

Flags

You can specify one of the following flags to describe the function:

r The tape is read and loaded into the current directory. This should not be done lightly; the **r** key should only be used to restore a complete dump tape onto a clear file system or to restore an incremental dump tape after a full level zero restore. Thus

```
/etc/mkfs /dev/rhd?
/etc/mount /dev/hd? /mnt
cd /mnt
restorebsd r
```

is a typical sequence to restore a complete dump. Another **restorebsd** can be done to get an incremental dump in on top of this. Note that **restorebsd** leaves a file **restoresymtable** in the root directory to pass information between incremental restore passes. This file should be removed when the last incremental tape has been restored.

R The **restorebsd** command requests a particular tape of a multi-volume set on which to restart a full restore (see the **r** key above). This allows **restorebsd** to be interrupted and then restarted.

x The named files are extracted from the tape. If the named file matches a directory whose contents had been written onto the tape, and the **h** key is not specified, the directory is recursively extracted. The owner, modification time, and mode are restored (if possible). If no file argument is given, then the root directory is extracted, which results in the entire content of the tape being extracted, unless the **h** key has been specified.

t The names of the specified files are listed if they occur on the tape. If no file argument is given, then the root directory is listed, which results in the entire content of the tape being listed, unless the **h** key has been specified. The **t** key replaces the function of the old **dumpdir** program.

i This mode allows interactive restoration of files from a dump tape. After reading in the directory information from the tape, **restorebsd** provides a shell like interface that allows the user to move around the directory tree selecting files to be extracted. The available commands are given below; for those commands that require an argument, the default is the current directory.

ls [arg] List the current or specified directory. Entries that are directories are appended with a "/". Entries that have been marked for extraction are prepended with a "*". If the verbose key is set the inode number of each entry is also listed.

cd arg Change the current working directory to the specified argument.

pwd Print the full path name of the current working directory.

add [arg] The current directory or specified argument is added to the list of files to be extracted. If a directory is specified, then it and all its descendents are added to the extraction list (unless the **h** key is specified on the command line). Files that are on the extraction list are prepended with a "*" when they are listed by **ls**.

delete [arg] The current directory or specified argument is deleted from the list of files to be extracted. If a directory is specified, then it and all its descendents are deleted from the extraction list (unless the **h** key is specified on the command line). The most expedient way to extract most of the files from a directory is to add the directory to the extraction list and then delete those files that are not needed.

extract All the files that are on the extraction list are extracted from the dump tape. **restorebsd** will ask which volume the user wishes to mount. The fastest way to extract a few files is to start with the last volume, and work towards the first volume.

setmodes	All the directories that have been added to the extraction list have their owner, modes, and times set; nothing is extracted from the tape. This is useful for cleaning up after a restore has been prematurely aborted.
verbose	The sense of the **v** key is toggled. When set, the verbose key causes the **ls** command to list the inode numbers of all entries. It also causes **restorebsd** to print out information about each file as it is extracted.
help	List a summary of the available commands.
quit	Restore immediately exits, even if the extraction list is not empty.
D	Toggles debugging mode.
x	Immediate quit (same as **quit** command).

The following flags can be used to modify the function of **restorebsd**:

v Normally **restorebsd** does its work silently. The **v** (verbose) key causes it to type the name of each file it treats preceded by its file type.

f The next argument to **restorebsd** is used as the name of the archive device instead of /dev/rmt?. If the name of the file is "-", **restorebsd** reads from standard input. Thus, **dumpbsd** and **restorebsd** can be used in a pipeline to dump and restore a file system with the command

```
/etc/dumpbsd 0f - /usr | (cd /mnt; /etc/restorebsd xf -)
```

F The next argument to **restorebsd** is used as the name of a file from which interactive input is read. Normally, standard input (or the controlling terminal if the **f** key specifies standard input) is read. This flag allows the interactive mode of **restorebsd** to be driven from a command file when the archive file is standard input. The interactive interface, the prompt for next volume number, and the prompt to set the access mode for "." are affected. Error recovery interaction and verifying operator readiness are not affected. For example, if the file **inputfile** contains

```
add
delete foo
add foo/bar
extract
1
yes
quit
```

then the command

```
    /etc/restorebsd iF inputfile
```

will use the interactive mode to automatically mark everything for extraction, unmark the directory foo, mark foo/bar, extract the marked files, specify volume 1, set the access mode for ".", and quit. The easiest way to determine the commands needed is to do the restore by hand once, and write down everything that you type.

y	The **restorebsd** command will not ask whether it should abort the restore if gets a tape error. It will always try to skip over the bad tape block(s) and continue as best it can.
m	The **restorebsd** command will extract by inode numbers rather than by file name. This is useful if only a few files are being extracted, and one wants to avoid regenerating the complete path name to the file.
h	The **restorebsd** command extracts the actual directory, rather than the files that it references. This prevents hierarchical restoration of complete subtrees from the tape.
b	The next argument to **restorebsd** is used as the block size of the tape (in kilobytes). If this key is not present, **restorebsd** tries to determine the block size dynamically.
c	Convert an old style dump tape (pre 4.2BSD file system).
d	Debug mode. The **restorebsd** command will perform many internal consistency checks and print out the debugging information.
s	The next argument to **restorebsd** is used as the number (1 origin) of the dump file to restore. Allows more than one dump file on a tape.

Files

/dev/rmt?	The default tape drive.
/tmp/rstdir*	File containing directories on the tape.
/tmp/rstmode*	Owner, mode, and time stamps for directories.
./restoresymtable	Information passed between incremental restores.

Note: The **restorebsd** command is not used to port information between AIX and BSD systems. The **tar** command is used for compatibility with BSD systems.

Related Information

See the following commands: "backup" in topic 1.1.32, "dumpbsd" in topic 1.1.143, "mount" in topic 1.1.278, "mkfs" in topic 1.1.269, "rrestore" in topic 1.1.395, and "rdump" in topic 1.1.362.

1.1.373 restrict

Purpose
Enforces licensing in a TCF cluster.

Syntax

```
application_name --- args ---¦
```

Note: **/etc/restrict/restrict** is not run directly. It is run based on running another command which is linked to **/etc/restrict/restrict**.

Description
The restrict program enforces licensing in a TCF cluster so that licensed programs can only be executed on sites where the program has been licensed to run. When the user or system administrator installs a program in a TCF cluster and wishes that the program only be runnable on specific sites within the cluster, the user or System administrator moves the program into the /etc/restrict directory, and replaces the program in its standard place in the AIX file system with a link or symbolic link to **/etc/restrict/restrict**.

When run, restrict opens /etc/restrict/restrictlist to determine the execution restrictions for the controlled program. **/etc/restrict/restrictlist** is an attribute file with stanzas for each controlled program. The name of the program being run (argument 0) is used as the name of the stanza in this attribute file.

Each stanza of the file should have two fields:

site	- which site(s) have been licensed to run the program, and
program	- the pathname of the real program to be run.

The **site** field is specified as a comma-separated list of site names or site numbers. Only if the site where the command is running is listed in the list given by the site field is the program indicated by the program field actually run.

If the execution site is not in the list of licensed sites, the following error message is written to **stderr**:

```
  Sorry, <commandname> has not been licensed to run on site <sitename>.
```

Note that if the program being restricted is normally installed with the set-user-id or set-group-id mode bit set, the program should continue to be installed that way when moved into the **/etc/restrict** directory. The link to **/etc/restrict** which the user will be calling directly is NOT to be made set-user-id, even though in so doing users who use **'ls -l'** to look at the program may be misled by the apparent owner and permissions of the program they are running.

Examples
An example /etc/restrict/restrictlist file might be:

```
  ls:
          site=fuji,akagi,9
          program=/etc/restrict/ls
```

and **/bin/ls** would be set up each as a symbolic link to

/etc/restrict/restrict.

When **/bin/ls** is run on site fuji, **restrict** validates that fuji is one of the licensed sites for ls, and therefore runs **/etc/restrict/ls** - the real **ls** program.

Similarly, **ls** could be run on site akagi or the site whose site number is 9.

If **ls** is run on site myoko, the **restrict** command denies the user permission to run the command - giving the user the following error message:

```
  Sorry, ls has not been licensed to run on site myoko.
```

Files

/etc/restrict/restrict	Restrict executable
/etc/restrict/restrictlist	Attribute file of restricted programs

1.1.374 rev

Purpose
Reverses characters in a line of a file.

Syntax

```
       +------------+
rev ---¦            +---¦
       +--- file ---+
                  ¦
         +--------+
```

Note: This command does not have MBCS support.

Description
The **rev** command copies the named **files** to the standard output, reversing the order of characters in every line. If no **file** is specified, the standard input is used.

1.1.375 rm, delete

Purpose
Removes files or directories.

Syntax

```
  one of
+--------+   +------------+
¦ rm     +---¦   +----+   +--- file ---¦
¦ delete ¦   +---¦ -f +---+           ¦
+--------+       ¦ -i ¦ ¦   +--------+
               ¦ ¦ -r ¦ ¦
               ¦ ¦ -- ¦ ¦
               ¦ ¦ -s ¦ ¦
               ¦ +----+ ¦
              +--------+
```

Description
The **rm** (**delete**) command removes the entries for **file**s from a directory. If an entry is the last link to a file, it is destroyed. To remove a file, you must have write permission in its directory. If the directory has been marked with the saved text permission mode (SVTX), only the owner of the file or directory or superuser can remove the file. If a file has no write permission and standard input is a work station, **rm** displays the file permission code and reads a line from standard input. If that line begins with **y**, **rm** deletes the file; otherwise it remains.

Note: When you use the **install** command to install files, a hidden directory is created. To remove these files, use **rm -rf**.

Flags

-f Does not prompt before removing a write-protected file.

-i Prompts you before deleting each file. When you use both **-i** and **-r** together, **rm** also asks if you want to examine directories.

-r Permits recursive removal of directories and their contents (for cases where **file** is a directory).

-s Permits the removal of symbolic links from a directory in a system-replicated file system. This option is ignored if the specified files are not symbolic links, and is unnecessary if the **r** option is used.

-- Indicates that the arguments following this flag are to be interpreted as file names. This null flag allows the specification of file names that start with a minus (-).

Examples

1. To delete a file:

   ```
   rm  myfile
   ```

   If there is another link to this file, the file remains under that name, but the name **myfile** is removed. If **myfile** is the only link, the file itself is deleted.

2. To delete a file silently:

   ```
   rm  -f  core
   ```

   This removes **core** without asking any questions or displaying any error messages. This is normally used in shell procedures. It prevents confusing messages from being displayed when deleting files that may or may not exist.

3. To delete files one by one:

   ```
   rm  -i  mydir/*
   ```

   This interactively asks you if you want to remove each file. After each file name is displayed, enter **y** to delete the file, or press **Enter** alone to keep it.

4. To delete a directory tree:

   ```
   rm  -ir  manual
   ```

   This recursively removes the contents of all subdirectories of **manual**, then removes **manual** itself, asking if you want to remove each file. For example:

   ```
      You: rm  -ir  manual

   System: rm:remove directory manual?

      You: y

   System: rm:remove directory manual/draft1?

      You: y

   System: rm:remove manual/draft1/chapter1?

      You: y

   System: rm:delete manual/draft1/chapter2?

      You: y

   System: rm:remove manual/draft2?

      You: n

   System: rm:  cannot remove directory
   ```

   Here, because **manual** contains directories, **rm** asks for permission to search these directories for files to delete. Then, for each of these directories, **rm** asks for permission to delete the files in that directory (**manual/draft1/chapter1** and **manual/draft1/chapter2**). Next, **rm** asks for permission to delete the directory **manual/draft1**. This is repeated for the other subdirectory (**manual/draft2**), and then **rm** asks for permission to delete **manual**. Because you denied permission to delete **manual/draft2**, **rm** will not delete **manual**.

5. To delete a symbolic link from a directory in a system-replicated file

    system:

```
      rm -s /etc/symlink
```

Related Information

See the following commands: "del" in topic 1.1.116 and "ln" in topic 1.1.234.

See the **unlink** system call in *AIX Operating System Technical Reference*.

1.1.376 rmail

Purpose
Handles remote mail received via UUCP.

Syntax

```
rmail --- user ---¦
                  ¦
       +--------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rmail** command interprets incoming mail received via **uucp**. It collapses **From** lines in the form generated by **bellmail** into a single line of the form:

```
  return-path!sender
```

It passes the processed mail on to **sendmail**.

The **rmail** command works with **uucp** and **sendmail**.

Related Information

See the following commands: "uucp" in topic 1.1.506 and "sendmail, mailq, newaliases" in topic 1.1.417.

1.1.377 rmdel

Purpose
Removes a delta from a Source Code Control System (SCCS) file.

Syntax

```
rmdel -- -r SID -- file --¦
                          ¦
              +------+
```

Description

The **rmdel** command removes the delta specified by **SID** from each named **Source Code Control System** (SCCS) **file**. You can remove only the most recently created delta in a branch, or the latest trunk delta if it has no branches. In addition, the SID you specify must not be a version currently being edited for the purpose of making a delta. To remove a delta, you must either own the SCCS file and the directory, or you must be the user who created the delta you want to remove.

If you specify a directory in place of **file**, **rmdel** performs the requested actions on all SCCS files (those with file names that have the **s.**prefix). If you specify a **-** (minus) in place of **file**, **rmdel** reads standard input, and interprets each line as the name of an SCCS file. **rmdel** continues to take input until it reads an end-of-file character (**Ctrl-D**).

Flag

-rSID	Removes the delta **SID** from the SCCS file. This flag is required.

Related Information

See the following commands: "delta" in topic 1.1.117, "get" in topic 1.1.186, "sccshelp" in topic 1.1.411, and "prs" in topic 1.1.336.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.378 rmdir

Purpose
Removes a directory.

Syntax

```
rmdir -- directory --¦
                     ¦
       +-----------+
```

Description
The **rmdir** command removes a directory from the system. The directory must be empty before you can remove it, and you must have write permission in its parent directory. Use the **li -l** command to see if the directory is empty.

Example

To empty and remove a directory:

```
  rm   mydir/*   mydir/.*
  rmdir  mydir
```

This removes the contents of **mydir**, then removes the empty directory. The **rm** command displays an error message about trying to remove the directories . (dot) and .. (dot dot), and then **rmdir** removes them.

To remove a hidden directory, append @ to the directory name:

```
  rmdir test@
```

Hidden directories are created with the **-h** flag of the **mkdir** command.

Note that **rm mydir/* mydir/.*** first removes files with names that do not begin with a dot, then those with names that do begin with a dot. You may not realize that the directory contains file names that begin with a dot because the **li** or **ls** command does not normally list them.

Related Information

See the following command: "rm, delete" in topic 1.1.375.

See the **unlink** and **rmdir** system calls in *AIX Operating System Technical Reference*.

1.1.379 rmf

Purpose
Removes a folder.

Syntax

```
      +------------+   +--- -interactive ---+
rmf --¦            +---¦       one of       +---¦
      +- + folder -+   ¦ +---------------+ ¦
                       +-¦ -interactive   +-+
                         ¦ -nointeractive ¦
                         +---------------+

rmf -- -help --¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rmf** command is used to remove folders and the messages that they contain. **rmf** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

Warning: The **rmf** command irreversibly deletes messages that do not have other links.

The **rmf** command deletes all of the messages within the specified folder and then deletes the folder. If the folder contains files that are not messages, **rmf** does not delete those files and reports an error.

If you have read-only access to the specified folder, **rmf** does not delete the folder or any of its messages. **rmf** deletes only your private sequences and your current message information from the profile.

The **rmf** command does not delete folders recursively. Thus, you cannot remove subfolders by requesting the removal of a parent folder.

Flags

+*folder*	Specifies the folder to be removed. If you remove a subfolder, the parent of that folder becomes the current folder. If you remove the current folder, **+inbox** becomes current. The default folder is the current folder. If **+***folder* is not specified and **rmf** cannot find the current folder, **rmf** requests confirmation for removing **+inbox**.
-help	Displays help information for the command.
-interactive	Requests confirmation before removing the folder. If **+***folder* is not specified, this is the default.
-nointeractive	Removes the folder and its messages without requesting confirmation. If **+***folder* is specified, this is the default.

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.

Files

$HOME/.mh_profile	The MH user profile.

Related Information

See the MH command "rmm" in topic 1.1.380.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.380 rmm

Purpose
Removes messages.

Syntax

```
       +-----------+   +-------- cur ---------------------------------+
rmm ---¦           +---¦ +----------- all -------------------------+ +---¦
       +- +folder -+   +-¦   +------------ sequence ------------+   +-+
                         +---¦  one of                          +---+
                             ¦ +-------+   +-----------------+ ¦ ¦
                           ¦ +-¦ num   +---¦      one of     +-+ ¦
                           ¦   ¦ first ¦   ¦¦+-------------+ ¦   ¦
                           ¦   ¦ prev  ¦   +-¦ :num   -prev +-+   ¦
                           ¦   ¦ cur   ¦     ¦ :+num  -cur  ¦     ¦
                           ¦   ¦ .     ¦     ¦ :-num  -.    ¦     ¦
                           ¦   ¦ next  ¦     ¦ -num   -next ¦     ¦
                           ¦   ¦ last  ¦     ¦ -first -last ¦     ¦
                           ¦   +-------+     +--------------+     ¦
                           +--------------------------------------+

rmm --- -help ---¦
```

¦ Do not put a blank between these items.

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rmm** command is used to remove messages from active status. **rmm** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **rmm** command renames the specified message files so that their file names have preceding commas. You can use these files as temporary backups and arrange for the **cron** command to delete your backups periodically.

Flags

+*foldermsgs* Specifies the messages that you want to remove. *msgs* can be several messages, a range of messages, or a single message. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default message is the current message in the current folder. **rmm** does not change the current message.

-help Displays help information for the command.

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.
rmmproc:	Specifies the program used to remove messages from a folder.

Files

.profile	
$HOME/.mh_profile	The MH user profile.

Related Information

See the MH command "rmf" in topic 1.1.379.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.381 rmt

Purpose
Remote magnetic tape protocol module.

Syntax

```
rmt ----¦
```

Description

The **rmt** command is a program used by the remote dump and restore programs in manipulating a magnetic tape drive through an interprocess communication connection. **rmt** is normally started up with an **rexec** or **rcmd** call.

The **rmt** program accepts requests specific to the manipulation of magnetic tapes, performs the commands, then responds with a status indication. All responses are in ASCII and in one of two forms. Successful commands have responses of

Anumber\n

where **number** is an ASCII representation of a decimal number. Unsuccessful commands are responded to with

Eerror-number\error-message\n

where **error-number** is one of the possible error numbers described in **intro** and **error-message** is the corresponding error string as printed from a call to **perror**. The protocol is comprised of the following commands (a space is present between each token).

O device mode	Open the specified **device** using the indicated **mode**. **Device** is a full path name and **mode** is an ASCII representation of a decimal number suitable for passing to **open**. If a device had already been opened, it is closed before a new open is performed.
C device	Close the currently open device. The **device** specified is ignored.
L whence offset	Perform an **lseek** operation using the specified parameters. The response value is that returned from the **lseek** call.
W count	Write data onto the open device. **rmt** reads **count** bytes from the connection, aborting if a premature end-of-file is encountered. The response value is that returned from the **write** call.
R count	Read **count** bytes of data from the open device. If **count** exceeds the size of the data buffer (10 kilobytes), it is truncated to the data buffer size. **rmt** then performs the requested **read** and responds with **Acount-read**\n if the read was successful; otherwise an error in the standard format is returned. If the read was successful, the data read is then sent.

I operation count	Perform a MTIOCOP **ioctl** command using the specified parameters. The parameters are interpreted as the ASCII representations of the decimal values to place in the **mt_op** and **mt_count** fields of the structure used in the **ioctl** call. The return value is the **count** parameter when the operation is successful.
S	Return the status of the open device, as obtained with a MTIOCGET **ioctl** call. If the operation was successful, an "ack" is sent with the size of the status buffer, then the status buffer is sent (in binary).

Any other command causes **rmt** to exit.

Related Information

See the following commands: "rdump" in topic 1.1.362 and "rrestore" in topic 1.1.395.

1.1.382 roffbib

Purpose
Runs off bibliographic data base.

Syntax

```
           +-----------+   +-----------+   +------+   +----------+   +--------
roffbib ---¦ +-------+ +---¦           +---¦      +---¦          +---¦ +----+
           +-¦ -e -r +-+   +- -T term -+   +- -x -+   +- -m mac -+   +-¦ -Q +
             ¦ -h -s ¦                                                  ¦ -V ¦
            ¦¦ -n    ¦¦                                                ¦+----+
            ¦¦ -o    ¦¦                                                +------
            ¦+-------+¦
            +---------+
```

Note: This command does not have MBCS support.

Description

The **roffbib** command prints out all records in a bibliographic data base, in bibliography format rather than as footnotes or endnotes. Generally it is used in conjunction with **sortbib**:

```
     sortbib  database | roffbib
```

The **roffbib** command accepts most of the options understood by **nroff**, most importantly the **-T** flag to specify terminal type.

If abstracts or comments are entered following the %X field key, **roffbib** will format them into paragraphs for an annotated bibliography. Several %X fields may be given if several annotation paragraphs are desired. The **-x** flag will suppress the printing of these abstracts.

A user-defined set of macros may be specified after the **-m** option. There should be a space between the **-m** and the macro file name. This set of macros will replace the ones defined in /usr/lib/tmac/tmac.bib. The **-V** flag will send output to the Versatec; the **-Q** flag will queue output for the phototypesetter.

Four command-line registers control formatting style of the bibliography, much like the number registers of **ms**. The command-line argument **-r**N1 will number the references starting at one (1). The flag **-r**V2 will double space the bibliography, while **-r**V1 will double space references but single space annotation paragraphs. The line length can be changed from the default 6.5 inches to 6 inches with the **-r**L6i argument, and the page offset can be set from the default of 0 to one inch by specifying **-r**O1i (capital O, not zero). Note: with the **-V** and **-Q** flags the default page offset is already one inch.

Files

/usr/lib/tmac/tmac.bib File of macros used by **nroff/troff**.

Related Information

See the following commands: "refer" in topic 1.1.365, "addbib" in topic 1.1.14, "sortbib" in topic 1.1.433, and "lookbib, indxbib" in topic 1.1.244.

1.1.383 rpcgen

Purpose

Compiles a Remote Procedure Call program when NFS is installed on your system.

Syntax

```
                 +----+
                 ¦ -c ¦
             +--¦ -h +------+
             ¦   +----+      ¦   +--------------+
rpcgen ---¦      one of  +---¦              +--- input file ---¦
             ¦      +-----+ ¦   +- -o outfile -+
             +- -s -¦ udp +-+
                     ¦ tcp ¦
                     +-----+
```

Description
The **rpcgen** command generates C Language code for implementing an RPC protocol. Input to **rpcgen** is in Remote Procedure Call Language (RPCL). RPCL is similar to the C Programming Language.

The **rpcgen** command operates in the following modes:

Converts RPCL definitions to C Language definitions and puts them in header file.

Compiles the XDR routines that serialize or convert the data betwee the machine issuing the Remote Procedure Call and the machine carrying it out.

Compiles converted RPCL definitions and puts them in a header fil named **infile.h**. Compiles the XDR routines and puts them in **infile.c**.

Compiles an RPC server skeleton. Using the skeleton, you can writ local procedures that implement RPC servers without invoking RPC protocols.

In each mode, the **inputfile** can contain comments (with the same format as C Language comments) and preprocessor directives. The comments are ignored and the directives are copied into the output header file. You can customize XDR routines by leaving some data types undefined. For every undefined data type, **rpcgen** will assume that a routine with an **xdr** prefix exists.

Flags

-c Compiles XDR routines.

-h Compiles C data-definitions in a header file.

-o {outfile} Specifies the name of the output file. If you do not specify the output file name, **rpcgen** uses the standard output as the default.

-s transport Compiles a server using a transport. This flag must be invoked once for each transport being served.

Related Information

See the following command: "rpcinfo" in topic 1.1.384.

See **xdr** subroutines in the *AIX Technical Reference* (External data representation).

See **rpcgen** in the *AIX Programming Tools and Interface Guide*.

1.1.384 rpcinfo

Purpose
Reports Remote Procedure Call status information when NFS is installed on your system.

Syntax

```
           +--- -p --- host ----------------------------+
rpcinfo ---¦ one of          ¦                          +---¦
           ¦ +----+ +--------+             +-----------+ ¦
           +-¦ -t +--- host --- program ---¦           +-+
             ¦ -u ¦                        +- version -+
             +----+
```

Description
The **rpcinfo** command reports the status of a Remote Procedure Call sent to an RPC sever. The **host** parameter specifies the Remote Procedure Call server. The **program** parameter specifies the program used by the remote procedure. It can be name or a number. The version number of the program used by the remote procedure is specified by **version**. If you do not specify a version number, **rpcinfo** searches for all registered version numbers and calls each one.

Note: The **rpcinfo** command uses **version** 0 to search for all registered versions, since 0 is not usually assigned as a program's version number. If **version** 0 is assigned to a program's version number, **rpcinfo** uses a high level number in its place.

Flags

-p Queries the **host** port map service and displays a list of registered RPC programs. If **host** is not specified, the value returned by **hostname** is the default value.

-t Uses the TCP/IP data transport to make a Remote Procedure Call to **procedure** 0 of the **program** on the specified **host** and report if a response was received.

-u Use the **progname** UDP/IP data transport to make a Remote Procedure Call to **procedure** 0 of the **program** on the specified **host** and report if a response was received.

Files

/etc/rpc RPC program names.

1.1.385 rpc.lockd

Purpose

rpc.lockd - network lock daemon.

Syntax

```
                     +---------------+
/etc/rpc.lockd ---¦ +-----------+ +---¦
                     +-¦ -t timeout +-+
                       ¦ -g          ¦
                       +-----------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rpc.lockd** command processes lock requests that are either sent locally by the kernel or remotely by another lock daemon. The **rpc.lockd** command forwards lock requests for remote data to the server site's lock daemon through the RPC/XDR package. The **rpc.lockd** command then requests the status monitor daemon, **rpc.statd** for monitor service. The reply to the lock request will not be sent to the kernel until the status daemon and the server site's lock daemon have replied.

If either the status monitor or server site's lock daemon is unavailable, the reply to a lock request for remote data is delayed until all daemons become available.

When a server recovers, it waits for a grace period for all client site **rpc.lockds** to submit reclaim requests. Client site **rpc.lockds**, on the other hand, are notified by the statd of the server recovery and promptly resubmit previously granted lock requests. If **rpc.lockd** fails to secure a previously granted lock at the server site, the **rpc.lockd** command sends SIGLOST to a process.

Flags

-t timeout
The **rpc.lockd** command uses **timeout** (seconds) as the interval instead of the default value (15 second) to retransmit lock request to the remote server.

-g graceperiod
The **rpc.lockd** command uses **graceperiod** (seconds) as the grace period duration instead of the default value (45 seconds).

Related Information

See the following commands in the *AIX Technical Reference*: **fcntl**, **lockf**, **signal**.

See the fcntl system call and the signal system call in the *AIX Technical Reference*.

1.1.386 rpc.mountd

Purpose
NFS mount request server.

Syntax

/etc/rpc.mountd ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **rpc.mountd** command is an **rpc** server that answers file system mount requests. It reads the file **/etc/exports** to determine which file systems are available to which machines and users. It also provides information as to which clients have file systems mounted. This information can be printed using the **showmount** command.

The **mountd** daemon is normally invoked by **inetd**.

Related Information

See the following command: "showmount" in topic 1.1.424.

See the **inetd** command in the *AIX TCP/IP User's Guide*.

1.1.387 rpc.rexd

Purpose

RPC-based remote execution server.

Syntax

/etc/rpc.rexd ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rpc.rexd** command is the Sun RPC server for remote program execution. This daemon is started by the **inetd** command whenever a remote execution request is made, if the following line is placed in **/etc/servers**:

```
  rpc tcp /etc/rpc.rexd 100017 1
```

For non-interactive programs, standard file descriptors are connected directly to TCP connections. Interactive programs involve pseudo-terminals, similar to the login sessions provided by the **rlogin** command. This daemon may use the NFS to mount file systems specified in the remote execution request.

Files

/dev/ttypn	Pseudo-terminals used for interactive mode.
/etc/passwd	Authorized users.

Related Information

See the following command: "onrpc" in topic 1.1.305.

See the **inetd** command in the *AIX TCP/IP User's Guide*.

1.1.388 rpc.rquotad

Purpose

Remote quota server.

Syntax

/etc/rpc.rquotad ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rquotad** command is an **rpc** server which returns quotas for a user of a local file system which is mounted by a remote machine over the NFS. The results are used by **quota** to display user quotas for remote file systems.

Files

quotas	Quota file at the file system root.

Related Information

See the following commands: "quota" in topic 1.1.350, "nfsd" in topic 1.1.294, and "onrpc" in topic 1.1.305.

See the **inetd** command in the *AIX TCP/IP User's Guide*.

1.1.389 rpc.rstatd

Purpose

Kernel statistics server.

Syntax

/etc/rpc.rstatd ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rpc.rstatd** command is a server which returns performance statistics obtained from the kernel. These statistics are graphically displayed by **perfmeter**. (The **perfmeter** command is not a part of AIX, but may exist on other UNIX platforms.) The **rstatd** daemon is normally invoked by **inetd**.

Related Information

See the **inetd** command in the *AIX TCP/IP User's Guide*.

1.1.390 rpc.rusersd

Purpose
Displays list of active AIX Network File system users.

```
/etc/rpc.rusersd ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **rpc.rusersd** daemon returns a list of users on the network. The **inetd** daemon invokes **rpc.rusersd**.

Files

/etc/services
/etc/inetd.conf

Related Information

See the following command: "rusers" in topic 1.1.401.

1.1.391 rpc.rwalld

Purpose
Handles requests for the **rwall** and **shutdown** commands.

Syntax

/etc/rpc.walld ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **rpc.rwalld** daemon handles requests from the **rwall** and **shutdown** commands. The **inetd** daemon invokes **rpc.rwalld**.

Files

/etc/services
/etc/inetd.conf

Related Information

See the following command: "rwall" in topic 1.1.402.

1.1.392 rpc.sprayd

Purpose

spray server.

Syntax

/etc/rpc.sprayd ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rpc.sprayd** command is a server which records the packets sent by **spray**. The **rpc.sprayd** daemon is normally invoked by **inetd**.

Related Information

See the following command: "spray" in topic 1.1.441.

See the **inetd** command in the *AIX TCP/IP User's Guide*.

1.1.393 rpc.statd

Purpose
Network status monitor.

Syntax

```
/etc/rpc.statd ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **rpc.statd** daemon is an intermediate version of the status monitor for NFS file locking. It interacts with the **lockd** command to provide the crash and recovery functions.

Files

/etc/services

Related Information

In this book, see the following command: "rpc.lockd" in topic 1.1.385.

1.1.394 rpc.yppasswdd

Purpose

Server for modifying NIS password file.

Syntax

```
                                  +----------------+
/etc/rpc.yppasswdd --- file ---¦                  +---¦
                                  +- -m --- arg ---+
                                                ¦
                                         +-------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **rpc.yppasswdd** command is an NIS server that handles password change requests from **yppasswd**. It changes a password entry in **file**, which is assumed to be in the format of **passwd**. An entry in **file** will only be changed if the password presented by **yppasswd** matches the encrypted password of that entry.

This server is not run by default, nor can it be started up from **inetd**. If it is desired to enable remote password updating for NIS, an entry for **rpc.yppasswdd** should be put in the **/etc/rc.nfs** file of the host serving as the master for the NIS **passwd** file of the current NIS domain.

Notes:

1. This can be accomplished by uncommenting (removing the leading "#" characters from) lines found in the default **/etc/rc.nfs** file.

2. If the NIS master host is a site in a TCF cluster, **rpc.yppasswdd** should only be started on that one host in the cluster which is the master. This is already accounted for in **/etc/rc.nfs** by a check for the existence of the file **/local/yp/<domain>/MASTER**.

You can either choose to make the NIS password input file be **/etc/passwd** or select another input file of the same format. You would select an alternate password input file, for instance, if you wanted to allow additional users to log on to the master NIS host beyond those permitted to log on to other hosts in your NIS domain.

To select an alternate NIS password input file, for example **/etc/yp/passwd**, you should start up **/etc/rpc.yppasswdd** as follows:

```
  /etc/rpc.yppasswdd /etc/yp/passwd -m passwd DIR=/etc/yp
```

Flag

-m After **file** is modified, a **make** will be performed in **/etc/yp**. Any arguments following the flag will be passed to **make**.

Related Information

See the following command: "domainname" in topic 1.1.133, "yppasswd" in topic 1.1.552, "ypinit" in topic 1.1.550, and "passwd, chfn, chsh" in topic 1.1.312.

See the **inetd** command in the *AIX TCP/IP User's Guide*.

See "Configuring the Network Information Services on Your System" - "Starting the **yppasswd** Daemon," in *Managing the AIX Operating System*.

See "passwd" file format in the *AIX Operating Systems Technical Reference*.

1.1.395 rrestore

Purpose
Restores a file system dump across the network.

Syntax

```
             +----------+                           +------------------+   +------
rrestore ---¦  one of  +--- f machine:device ---¦ +--------------+ +---¦
             ¦ +------+ ¦                           +-¦ r            +-+   +--- f:
             +-¦ r  + +-+                             ¦ F input_file ¦¦  +-------
               ¦ R  i ¦                              ¦¦ y            ¦¦
               ¦ x    ¦                              ¦¦ m            ¦¦
               +------+                              ¦¦ h            ¦¦
                                                     ¦¦ b            ¦¦
                                                     ¦¦ c            ¦¦
                                                     ¦¦ d            ¦¦
                                                     ¦¦ s            ¦¦
                                                     ¦+--------------+¦
                                                     +----------------+
```

Description

The **rrestore** command obtains from a magnetic tape or specified device files saved by a previous **rdump** or **dumpbsd**. The command is identical in operation to **restorebsd** except the **f** should be specified in the **key** and the **argument** supplied should be of the form **machine:device**.

The **rrestore** command creates a remote server on the client machine to access the tape device.

Related Information

See the following commands: "restorebsd" in topic 1.1.372, "rmt" in topic 1.1.381, "rdump" in topic 1.1.362, and "dumpbsd" in topic 1.1.143.

1.1.396 rscsmail

Purpose

Delivers mail from AIX users to VM users.

Syntax

```
                                 one of
                       +---------------------------+
/usr/bin/rscsmail ---¦ vmuserid                    +---¦
                       ¦ vmuserid.nodeid             ¦
                       ¦ unixuserid.vmuserid.nodeid ¦
                       +---------------------------+
```

Note: This command is for the System/370 only.

Description

The **mail** program may be used to send messages from an AIX user on one site to a VM user on the same machine, to a VM user on another machine, or to an AIX user on another machine. The **/usr/bin/rscsmail** command is called only from **sendmail** and is not invoked by the user. The **sendmail** program passes addresses in one of the following forms:

vmuserid	the id of a VM user on the same site
vmuserid.nodeid	the id of a VM user on another VM machine
unixuserid.vmuserid.nodeid	the id of a UNIX user on another VM machine

The command specifies that the file sent is treated as a **Note** by the receiving CMS or MVS/TSO user.

To use this facility you must make an entry in the **sendmail.cf** file to define mail routing for VM use. You may define a mailer as shown in the following example:

```
     Mrscs P=/usr/bin/rscmail F=n A=rscsmail $u
```

This defines a mailer named **rscs**, which calls **/usr/bin/rscsmail** with an argument of the receiving user ($u).

A new set of rules must also be inserted into the configuration file to determine when to call this mailer. The default format to send messages is to append .RSCS to the VM address. When parsing the address, the **sendmail** router recognizes the .RSCS and calls the rscs mailer.

Related Information

See the following commands: "uvcp" in topic 1.1.517 and "sendmail, mailq, newaliases" in topic 1.1.417.

1.1.397 rscssrvr

Purpose

Delivers notes sent from VM to an AIX user's mailbox.

Syntax

/usr/bin/rscssrvr --- **file** --- **userid** ---¦

Note: This command is for the System/370 only.

Description

The **rscssrvr** program is called from **rdrdaemon** when it encounters a file of type note or Note (a file sent using the note facility of VM). It translates the note into a mail message and calls **sendmail** to deliver the message to the user's mailbox. This program should not be invoked by the user.

Related Information

See the following commands: "vucp" in topic 1.1.527, "rdrdaemon" in topic 1.1.361, and "sendmail, mailq, newaliases" in topic 1.1.417.

1.1.398 runacct

Purpose
Runs daily accounting.

Syntax

```
                         +---------------------+
/usr/lib/acct/runacct ---¦         +---------+ +---¦
                         +- mmdd --¦         +-+
                                   +- state -+
                                            ¦
                                    +-------+
```

Description
The **runacct** command is the main daily accounting shell procedure. Normally initiated by **cron**, **runacct** processes connect, fee, disk, queueing system and process accounting data files. It also prepares summary files for the **prdaily** procedure or for billing purposes.

The **runacct** command protects active accounting files and summary files in the event of run-time errors. It records its progress by writing descriptive messages into the file **/usr/adm/acct/nite/active**. When **runacct** encounters an error, it writes a diagnostic message to **/dev/console**, sends **mail** to users **root** and **adm**, and exits.

The **runacct** procedure also creates two temporary files, **lock** and **lock1** in the directory **/usr/adm/acct/nite**, which it uses to prevent two simultaneous calls to **runacct**. It uses the file **lastdate** (in the same directory), to prevent more than one invocation per day.

The **runacct** command breaks its processing into separate, restartable ***states***. As it completes each state, it writes the name of the next state in **/usr/adm/acct/nite/statefile**. **runacct** processes the various states in the following order:

State	**Actions**
SETUP	Moves the active accounting files to working files and restarts the active files.
WTMPFIX	Verifies the integrity of the **wtmp** file, correcting date changes if necessary.
CONNECT1	Calls **acctcon1** to produce connect session records.
CONNECT2	Converts connect session records into total accounting records (**tacct.h** format).
PROCESS	Converts process accounting records into total accounting records (**tacct.h** format).
MERGE	Merges the connect and process total accounting records.
FEES	Converts the output of **chargefee** into total accounting records (**tacct.h** format) and merges them with the connect and process total accounting records.
DISK	Merges disk accounting records with connect, process, and

	fee total accounting records.
QUEUEACCT	Sorts the queue (printer) accounting records, converts them into total accounting records (**tacct.h** format), and merges them with other total accounting records.
MERGETACCT	Merges the daily total accounting records in **daytacct** with the summary total accounting records in **/usr/adm/acct/sum/tacct**.
CMS	Produces command summaries in the file **/usr/adm/acct/sum/cms**.
USEREXIT	If the shell file **/<LOCAL>/adm/siteacct** exists, calls it at this point to perform site-dependent processing.
CLEANUP	Deletes temporary files and exit.

To restart **runacct** after a failure, first check the **/usr/adm/acct/nite/active** file for diagnostic messages, then fix any damaged data files such as **pacct** or **wtmp**. Remove the **lock** files and **lastdate** file (all in the **/usr/adm/acct/nite** directory), before restarting **runacct**. You must specify the **mmdd** parameter if you are restarting **runacct**. It specifies the month and day for which **runacct** is to rerun the accounting. **runacct** determines the entry point for processing by reading **statefile**. To override this default action, specify the desired **state** on the **runacct** command line. For a more detailed discussion of restarting **runacct**, see *Managing the AIX Operating System*.

It is not usually a good idea to restart **runacct** in the SETUP **state**. Instead, perform the setup actions manually and restart accounting with the WTMPFIX state, as follows:

```
  runacct  mmdd  WTMPFIX
```

If **runacct** fails in the PROCESS state, remove the last **ptacct** file, because it will be incomplete.

Examples

1. To start **runacct**:

   ```
   nohup  /usr/lib/acct/runacct  2> /usr/adm/acct/nite/accterr  &
   ```

   This starts **runacct** in the background (**&**), ignoring all INTERRUPT and QUIT signals (**nohup**). All standard error output is written to the file **/usr/adm/acct/nite/accterr**.

2. To restart **runacct**:

   ```
   nohup  /usr/lib/acct/runacct  0601  2>> /usr/adm/acct/nite/accterr  &
   ```

   This restarts **runacct** for the day of June 1 (**0601**). **runacct** reads the file **/usr/adm/acct/nite/statefile** to find out the state to begin with. Standard error output is added to the end of the file **/usr/adm/acct/nite/accterr**.

3. To restart **runacct** in a specific state, in this case the **MERGE** state:

   ```
   nohup /usr/lib/acct/runacct 0601 MERGE 2>> /usr/adm/acct/nite/accterr
   ```

Files

/usr/adm/wtmp	Login/logout history file.
/<LOCAL>/adm/pacct*	Process accounting file.
/usr/adm/acct/nite/daytacct	Disk usage accounting file.
/usr/adm/qacct	Active queue accounting file.
/usr/adm/fee	Record of fees charge to users.
/usr/adm/acct/sum/*	Command and total accounting summary files.
/usr/adm/acct/nite/ptacct*.mmdd	Concatenated version of **pacct** files.
/usr/adm/acct/nite/active	**runacct** message file.
/usr/adm/acct/nite/lock*	Prevent simultaneous invocation of **runacct**.
/usr/adm/acct/nite/lastdate	Contains last date **runacct** was run.
/usr/adm/acct/nite/statefile	Contains current state to process.

Related Information

See the following commands: "acct/*" in topic 1.1.6, "acctcms" in topic 1.1.7, "acctcom" in topic 1.1.8, "acctcon1, acctcon2" in topic 1.1.9, "acctmerg" in topic 1.1.11, and "cron" in topic 1.1.97.

See the **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

See "Running System Accounting" in *Managing the AIX Operating System*.

1.1.399 runcat

Purpose
Creates a message catalog using symbolic identifiers.

Syntax

```
                              +-----------+
runcat --- cmd --- descfile ---¦           +---¦
                              +- catfile -+
```

Description

The **runcat** command invokes the **mkcatdefs** command to preprocess the symbolic information in the descriptor file (**descfile**) and then pipes the output to the **gencat** command to produce the actual catalog along with the header file containing the symbolic definitions.

The descriptor file contains the symbolically named messages and sets that are processed and used as input to the **gencat** command. If the catalog file (**catfile**) is omitted, a catalog by the name **cmd.cat** is created (the header file generated for use within a user program is named **cmd_msg.h**).

The **runcat** command first searches the directory **/usr/lib/mbcs/msg/language.codeset/** for **catfile**. The **language.codeset** depends on the current locale. (The locale may be changed using the **LANG** environment variable.) For example, if the current locale is English, the default search directory would be **/usr/lib/mbcs/msg/En.pc850/**. If **catfile** is not found in the default search directory, **runcat** searches the current directory for **catfile**.

Related Information

See the following commands: "gencat" in topic 1.1.184 and "mkcatdefs" in topic 1.1.267.

See the "Message Catalog Generation" section of the "International Character Support" chapter of the *AIX Operating System Programming Tools and Interfaces*.

See the "Messages" chapter in the *AIX MBCS Guide*.

1.1.400 rup

Purpose
Displays the host status of local machines (RPC version) when NFS is installed.

Syntax

```
       +--------+   +----------+   +----------------+
rup ---¦ one of +---¦          +---¦                +---¦
       ¦ +----+ ¦   +- -d num -+   +--- hostname ---+
       +-¦ -h +-+                                 ¦
         ¦ -l ¦                      +------------+
         ¦ -t ¦
         +----+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **rup** command broadcasts a query on the local network and displays the responses it receives. It displays an NFS status report on system usage times (up times) and load averages. To query specific hosts, specify in their names in the **hostname** parameter.

Note: Remote hosts respond only if the **rpc.rstatd** daemon is running. The **rpc.rstatd** daemon is started by the **inetd** program.

Note: When querying specific hosts, they will be displayed in the order listed on the command line.

Flags

When used without flags, the **rup** command displays the responses in the order it receives them. Use the following flags to change the display order.

-d num	Displays the first **num** host names.
-h	Displays responses alphabetically by host name.
-l	Displays responses by load average.
-t	Displays responses by up time.

Files

/etc/servers

1.1.401 rusers

Purpose
Identifies users logged in on local machines when NFS (RPC version) is installed.

Syntax

```
          +--------+   +------------+   +----------+   +----------------+
rusers ---¦ one of +---¦   +----+   +---¦          +---¦                +---¦
          ¦ +----+ ¦   +---¦ -a +---+   +- -d num -+   +--- hostname ---+
          +-¦ -i +-+       ¦ -l ¦ ¦                                     ¦
            ¦ -h ¦       ¦ +----+ ¦                       +-------------+
            ¦ -u ¦       +--------+
            +----+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **rusers** command broadcasts a query on the local network and displays the responses it receives. To query specific hosts, specify their names in the **hostname** parameter.

Note: Remote hosts respond only if they are running the **rpc.rusersd** daemon.

Flags

The **rusers** command displays the responses to the query in the order it receives them. Each machine is listed on a separate line. Use the following flags to change the print order:

-a	Responds for all machines even if no users are logged on.
-d num	Displays the first **num** host names.
-h	Sorts responses alphabetically by host name.
-i	Sorts responses by idle time. Idle time is reported if a user has not typed into the system for more than a minute.
-l	Responds with a longer listing, in the style of the **who** command.
-u	Sorts responses by numbers of users.

Files

/etc/servers

Related Information

See the following commands: "who" in topic 1.1.537 and "rpc.rusersd" in topic 1.1.390.

1.1.402 rwall

Purpose
Writes to all users over a network when NFS is installed.

Syntax

```
         +---------------------+
rwall ---+- -h --- hostname ---+---¦
         ¦                     ¦ ¦
         ¦      +------------+ ¦
         +- -n --- netgroup ---+
                             ¦
                +------------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **rwall** command reads a message from standard input and broadcasts it to all users logged in to the specified host machines. It reads the message from standard input until it reaches the end of file. The **rwall** sends the message with the following introduction line:

```
  Broadcast Message.....
```

Note: Users can receive a message only if they are running **rpc.rwalld**. This daemon is started from the **/etc/servers** file with the **inetd** daemon.

Flag

-h Sends the message to specific **hostnames** only.

-n Sends the message to specific **netgroups** only. Groups in the network are defined in the **netgroup** file.

Files

/etc/netgroup
/etc/servers

Related Information

See the following command: "rpc.rwalld" in topic 1.1.391.

1.1.403 sa

Purpose

sa - system accounting

Syntax

```
           +--------------+   +-----------------+   +----------------+   +--
/etc/sa ---¦ +----------+ +---¦                 +---¦                +---¦
           +-¦ -a -i -r +-+   +- -S savacctfile -+   +- -U usacctfile -+   +-
             ¦ -b -j -s ¦
            ¦¦ -c -k -t ¦¦
            ¦¦ -d -K -u ¦¦
            ¦¦ -D -l -v ¦¦
            ¦¦ -f -n    ¦¦
            ¦+----------+¦
            +------------+
```

Description

The **sa** command reports on, cleans up, and generally maintains accounting files.

The **sa** command is able to condense the information in **/usr/adm/acct** into a summary file **usr/adm/savacct** which contains a count of the number of times each command was called and the time resources consumed. This condensation is desirable because on a large system **/usr/adm/acct** can grow by 100 blocks per day. The summary file is normally read before the accounting file, so the reports include all available information.

If a file name is given as the last argument, that file will be treated as the accounting file; **/usr/adm/acct** is the default.

Output fields are labeled: "cpu" for the sum of user+system time (in minutes), "re" for real time (also in minutes), "k" for cpu-time averaged core usage (in 1k units), "avio" for average number of i/o operations per execution. With options fields labeled "tio" for total i/o operations, "k*sec" for cpu storage integral (kilo-core seconds), "u" and "s" for user and system cpu time alone (both in minutes) will sometimes appear.

Flags

-a	Print all command names, even those containing unprintable characters and those used only once. By default, those are placed under the name '***other.'
-b	Sort output by sum of user and system time divided by number of calls. Default sort is by sum of user and system times.
-c	Besides total user, system, and real time for each command print percentage of total time over all commands.
-d	Sort by average number of disk i/o operations.
-D	Print and sort by total number of disk i/o operations.
-f	Force no interactive threshold compression with **-v** flag.

-i	Don't read in summary file.
-j	Instead of total minutes time for each category, give seconds per call.
-k	Sort by cpu-time average memory usage.
-K	Print and sort by cpu-storage integral.
-l	Separate system and user time; normally they are combined.
-m	Print number of processes and number of CPU minutes for each user.
-n	Sort by number of calls.
-r	Reverse order of sort.
-s	Merge accounting file into summary file **/usr/adm/savacct** when done.
-t	For each command report ratio of real time to the sum of user and system times.
-u	Superseding all other flags, print for each command in the accounting file the user ID and command name.
-v	Followed by a number **n**, types the name of each command used **n** times or fewer. Await a reply from the terminal; if it begins with 'y', add the command to the category '**junk'. This is used to strip out garbage.
-S	The following filename is used as the command summary file instead of **/usr/adm/savacct**.
-U	The following filename is used instead of **/usr/adm/usracct** to accumulate the per-user statistics printed by the **-m** option.

Files

/usr/adm/acct	Raw accounting.
/usr/adm/savacct	Summary.
/usr/adm/usracct	Per-user summaries

Related Information
See the following commands: "ac" in topic 1.1.5 and "acct/*" in topic 1.1.6.

1.1.404 sact

Purpose
Displays current Source Code Control System (SCCS) file editing status.

Syntax

```
sact -- file --¦
               ¦
      +------+
```

Note: This command does not have MBCS support.

Description
The **sact** command reads Source Code Control System (SCCS) **file**s and writes to standard output the contents, if any, for the **p-file** associated with **file** (see "SCCS Files" in topic 1.1.186.1 for information on the contents of the p-file). If **-** (minus) is specified for **file**, **sact** reads standard input, and interprets each line as the name of an SCCS file. If **file** is a directory, **sact** performs its actions on all SCCS files (that is, those files with the **s.** prefix).

Related Information

See the following commands: "delta" in topic 1.1.117, "get" in topic 1.1.186, and "unget" in topic 1.1.493.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.405 sadc, sa1, sa2

Purpose
Provides a system activity report package.

Syntax

```
                    +-------------------+   +-----------+
/usr/lib/sa/sadc ---¦                   +---¦           +---¦
                    +- interval -- num -+   +- outfile -+

                   +-------------------+
/usr/lib/sa/sa1 ---¦                   +---¦
                   +- interval -- num -+

                  ¦
/usr/lib/sa/sa2 ---¦


----------------
¦ See the sar command for the format and flag description.
You cannot use the -o and -f flags with sa2.
```

Description
The operating system contains a number of counters that are incremented as various system actions occur. They include the following:

System unit utilization counter
Buffer usage counter
Disk and tape I/O activity counter
TTY device activity counter
Switching and system-call counter
File-access counter
Queue activity counter
Inter-process communications counter

The **sadc** command and the **sa1** and **sa2** shell procedures sample, save and process this data.

Note: These commands only report on local activities.

1.1.405.1 sadc

The **sadc** command, the data collector, samples system data **num** times every **interval** seconds. It writes in binary format to **outfile** or to the standard output. If you do not specify **interval** or **num**, a special record is written. This facility is used at system startup to mark the time when the counter restarts from zero.

1.1.405.2 sa1

Use the shell procedure **sa1**, a variant of **sadc** to collect and store binary data in the file **/usr/adm/sa/sadd**, where **dd** is the day of the month. The **interval** and **num** parameters specify that the record should be written **num** times at **interval** seconds. If you do not specify these parameters, one record is written. You must have permission to write in the directory **/usr/adm/sa** to use this command.

The **sa1** command is designed to be started automatically by the **cron** command.

1.1.405.3 sa2

Use the shell procedure **sa2**, a variant of the **sar** command, to write a daily report in the file **/usr/adm/sa/sardd**. See "sar" in topic 1.1.407 for a description of the flags.

The **sa2** command is designed to be started automatically by the **cron** command.

Files

/usr/adm/sa/sadd	Daily data file, **dd** represents the day of the month.
/usr/adm/sa/sardd	Daily report file, **dd** represents the day of the month.
/tmp/sa.adrfl	Address file.

Related Information

See the following commands: "cron" in topic 1.1.97, "sag" in topic 1.1.406, "sar" in topic 1.1.407, and "timex" in topic 1.1.472.

1.1.406 sag

Purpose
Displays a graph of system activity.

Syntax

```
       +--- -s08:00 ----+   +--- -e18:00 ----+   +- -f/usr/adm/sa/sadd -+
sag ---¦ +-- -s hh ---+ +---¦ +-- -e hh ---+ +---¦                      +--
       +-¦            +-+   +-¦            +-+   +------ -f file -------+
         +- -s hh:mm -+       +- -e hh:mm -+

    +--------------+   +---- -T$TERM -----+   +---------------+
 ---¦              +---¦                  +---¦ +----------+ +---¦
    +- -i seconds -+   +- -T workstation -+   +-¦ -x spec  +-+
                                                ¦ -y spec ¦¦¦
                                               ¦+----------+¦
                                               +------------+
```

¦ The default for **-y** is '% usr 0 100; % usr + % sys 0 100; % usr + % sys + %wio 0 100'.

Description
The **sag** command displays a graph of system activity. It gets information either from the daily activity file **usr/adm/sa/sadd** or from the binary data file selected by the **-f** flag. You must have already created this file by running the **sar** command with the **-o** flag. (See "sar" in topic 1.1.407.)

The **sag** command calls the **sar** command, selecting the desired data by string-matching the data column header.

Flags
The **sag** command passes the first four of the following flags to **sar** in order to collect the desired data for display. The last three flags specify plotting parameters.

-e hh[:mm]	Selects data up to the time specified by **hh[:mm]**). The default time is 18:00.
-f file	Reads data from **file**. The default **file** is **/usr/adm/sa/sadd**, the current daily data file.
-i seconds	Selects data at intervals as close as possible to **seconds**.
-s hh[:mm]	Selects data later than the specified time. Default is 08:00.
-T work station	Produces output suitable for **work station**. (See "tplot" in topic 1.1.477 for known work stations.) If you do not specify a work station, **sag** uses the value found in the shell variable **$TERM**.
-x spec	Specifies the x axis. **spec** has the following form:

name [**opname**]...[**lo hi**]

where **name** is a character string matching a column header in the **sar**-created data file (with an optional device name in brackets), or it is an integer value. **op** is +, **-**, *****, or / surrounded by blanks, with up to five **name**s specified. Parentheses are not recognized and evaluation is left to right. The + and **-** have precedence over ***** and / in evaluating expressions. **lo** and **hi** specify numeric scale limits. If these limits are unspecified, **sag** gets these limits from the data.

-y spec	Specifies the y axis. **spec** has the same form as **x spec**.

Specify only one **spec** for the x axis. If unspecified, the x axis assumes the time specified with the **-e** and **-s** flags (or their defaults if they are not used) as x axis limits. You can specify up to five **spec**s separated by ; (a semicolon) for **-y**. If unspecified, the y axis has the value:

```
  -y "%usr 0 100; %usr + %sys 0 100; %usr + %sys + %wio 0 100"
```

If you include blanks or an escaped carriage return (\Enter) within the **-x** and **-y spec**s, enclose them in " " (double quotation marks).

Files

/usr/adm/sa/sadd	Daily data file for day **dd**.

Related Information

See the following commands: "sar" in topic 1.1.407 and "tplot" in topic 1.1.477.

1.1.407 sar

Purpose
Collects, reports, or saves system activity information.

Syntax

```
      +- -u ---------+   +-----------+             +--- 1 ----+
sar ---¦ +- -A ------+ +---¦           +-- interval --¦          +---¦
      +-¦ +-------+ +-+   +- -o file -+             +- number -+
        +-¦ -a -u +-+
          ¦ -b -v ¦
         ¦¦ -c -w ¦¦
         ¦¦ -m -y ¦¦
         ¦¦ -q -r ¦¦
         ¦+-------+¦
         +---------+

      +- -u ------------+   +--- -s08:00 ----+   +----------+
sar ---¦ +- -A ---------+ +---¦     one of     +---¦          +---
      +-¦ +----------+ +-+   ¦ +------------+ ¦   +- -i sec -+
        +-¦ -a -m -w +-+     +-¦ -shh       +-+
          ¦ -b -q -y ¦¦       ¦ -shh:mm    ¦
         ¦¦ -c -u -r ¦¦       ¦ -shh:mm:ss ¦
         ¦¦ -v       ¦¦       +------------+
         ¦+----------+¦
         +------------+

    +--- -e18:00 ----+   +- -f/usr/adm/sa/sadd -+
 ---¦     one of     +---¦                      +---¦
    ¦ +------------+ ¦   +------ -f file -------+
    +-¦ -shh       +-+
      ¦ -shh:mm    ¦
      ¦ -shh:mm:ss ¦
      +------------+
```

Description
The first format of the **sar** command writes to standard output the contents of selected cumulative activity counters in the operating system. It writes information a total of **number** times spaced **interval** seconds apart. The default value of **number** is 1. You can also save the collected data in the file specified by **-o file**.

In the second format (with no sampling interval specified), **sar** extracts and writes to standard output records previously saved in a file. This file can be either the one specified by the **-f** flag or, by default, the standard system activity daily data file, **/usr/adm/sa/sadd**, for the current day, **dd**.

You can select information about specific system activities with flags. Not specifying any flags selects only **cpu** activity. Specifying the **-A** flag selects all activities.

Note: This command only reports on local activities.

Flags

-a Reports use of file access system routines:

iget/s	Calls per second to the inode look-up routine.
namei/s	Calls per second to the directory search routine.
dirblk/s	Directory blocks read per second by namei().

-A Report all data.

-b Reports buffer activity for transfers, accesses, and cache hit ratios:

lread/s, lwrit/s	Number of logical read/write requests per interval.
bread/s, bwrit/s	Number of block read/write operations per interval.
%rcache, %wcache	Cache hit ratios (for example, 1 - bread/lread).
pread/s, pwrit/s	Read/writes per interval on seekable raw devices.

-c Reports system calls:

scall/s	Total number of system calls per second.
rchar/s, wchar/s	Characters transferred per interval by read/write calls.
sread/s, swrit/s	
fork/s, exec/s	Specific system calls per second.

-e hh[:mm[:ss]]
Sets the ending time of the report. The default ending time is 18:00.

-f file Extracts records from **file** (created by **-o file**). The default **file** is the current daily data file, **/usr/adm/sa/sadd**.

-i seconds Selects data records at intervals as close as possible to the specified number of **seconds**. Otherwise, **sar** reports all intervals found in the data file.

-m Reports message and semaphore activities:

msg/s	IPC message primitives per second.
sema/s	IPC semaphore primitives per second.

-o file Saves the readings in **file** in binary form. Each reading is in a separate record and each record contains a tag identifying the time of the reading.

-q Reports average queue length while occupied, and percentage of time occupied:

runq-sz, %runocc	Runs queue of processes in memory and able to be run.

-r Reports paging statistics:

freeman	average pages available to user

	processes
freeswap	disk blocks available for process swapping
fault/s	address translation page faults.

-s hh[:mm[:ss]]
Sets the starting time of the data. That is, extract records time-tagged at or following the time specified. The default starting time is 08:00.

-u Reports CPU activity (this flag is on by default):

%usr	Percentage of CPU time devoted to the user.
%sys	Percentage of CPU time devoted to the kernel.
%wio	Percentage of CPU time waiting for block I/O to complete.
%idle	Percentage of CPU time idle.

-v Reports status of process, inode, and file tables:

proc-sz, inod-sz, file-sz	Entries in use at each sample point for each table.
proc-ov, inod-ov, file-ov	Overflows occurring at each sample point for each table.

-w Reports system switching activity:

pswch/s	Process switches per second.

-y Reports TTY device activity:

rawch/s	TTY raw input queue characters per second.
canch/s	TTY canonical input queue characters per second.
outch/s	TTY output queue characters per second.
revin/s	TTY receive interrupts per second.
xmtin/s	TTY transmit interrupts per second.
mdmin/s	TTY modem interrupts per second.

Files

/usr/adm/sa/sadd	Daily data file, where **dd** are numbers representing the day of the month.

Related Information

See the following command: "sag" in topic 1.1.406.

See the discussion of monitoring system activity in *Managing the AIX Operating System*.

1.1.408 savecore

Purpose
Manages core dumps of the operating system on the dump device.

Syntax
For AIX PS/2

```
                                      +-----------+
/etc/savecore ------------ dirname ---¦           +---¦
                ¦ +----+ ¦            +- sysname -+
               +-¦ -v +-+
                 ¦ -a ¦¦
                ¦+----+¦
                +------+
```

For AIX/370

```
                 +-------- -t ---------+   +- -d --- /dev/dump -+
/etc/savecore ---¦ +- -r --- dumpid --+ +---¦                     +---¦
                 +-¦                 ¦+-+   +- -d --- special ---+
                   ¦      +--- , ----+¦
                   ¦     one of       ¦
                   ¦  +-----------+   ¦
                   +--¦ -e dumpid +---+
                      ¦ -I   -i   ¦
                      ¦ -t   -l   ¦
                      +-----------+
```

Description

The **savecore** command on both AIX/370 and AIX PS/2 is used to copy a core dump of the operating system from the dump device into the AIX file system.

Note: You must have read/write access to the dump device.

SAVECORE ON AIX PS/2:

savecore checks the core dump to be certain it corresponds with the current running unix. If it does it saves the core image in the file **dirname/core.n** and the namelist, **dirname/unix.n**. The trailing **.n** in the path names is replaced by a number which grows every time **savecore** is run in that directory. **Savecore** is often included near the end of the **/etc/rc** file so that a core dump, if available, can be saved when the system reboots.

Before **savecore** writes out a core image, it reads a number from the file **dirname/minfree**. If the number of free kilobytes on the file system which contains **dirname** is less than the number obtained from the minfree file, the core dump is not saved. If the minfree file does not exist, **savecore** always writes out the core file (assuming that a core dump was taken).

savecore also logs a reboot message using facility LOG_AUTH. If the system crashed as a result of a panic, **savecore** logs the panic string too.

If the core dump was from a system other than **/unix**, the name of that system must be supplied as **sysname**.

SAVECORE ON AIX/370:

On AIX/370 it is possible for multiple core dumps to be stored concurrently on the dump device, so on AIX/370 there are additional arguments with **savecore** to allow one to manipulate these multiple dumps.

Flags

The following flags are for AIX/370 only:

-e dumpid	Copies the dump specified by **dumpid** from the *dump device*, along with its symbol table information, and send it to the standard output.
-r dumpid,...	Removes the dump(s) specified by **dumpid(s)** from the *dump device* and reclaims all space. The dumpid will not be reused until the *dump device* is reinitialized by the -I option.
-I or -i	Clears all dumps from the *dump device* and initializes it for use again.
-l	Display the number of the latest dump having been produced in the dump device. This is also the highest number being allocated so far.
-t	Prints a summary of all dumps in the *dump device*. The summary contains 1. **dumpid** 2. size 3. AIX version 4. date 5. time The listing is written to the standard output.
-d special	Name of the dump device. The default is **/dev/dump**.

The following flags are for PS/2s only:

-a	Saves the dump if the date of the dump is greater than three days from the current date.
-v	Verbose option.

Restrictions

The following restrictions apply to AIX/370 only:

Highest dumpid in the dump device can be 254. These dumpids are no reused until **/dev/dump** is re-initialized.

The AIX/370 **/dev/dump** as delivered can hold about four PANIC dumps.

The usable size of the dump device is limited by the size of a in-core map; therefore, it is equal to the smaller of

- the physical extents of the dump device
  or

- the parameter NDUMPS.

Example

The following example applies to AIX/370 only:

A typical sequence of **savecore** commands is

```
       /etc/savecore -t
       /etc/savecore -e 12 >dump12
       /etc/savecore -r 12
       crash -d dump12
```

This sequence will

1. list the dumpids and other data about dumps present in the dump device

2. copy dump 12 to an ordinary file

3. remove dump 12 from the dump device to allow its space to be reused

4. execute crash to interactively examine the dump.

Files

/unix	Current UNIX.
/dev/dump	default dump device.

Related Information

See the **AIX Dump Facility** in the **AIX Managing Guide**.

1.1.409 scan

Purpose
Produces a one-line-per-message scan listing.

Syntax

```
        +-----------+   +-------- all ---------------------------------+
scan ---¦           +---¦ +----------- all --------------------------+ +---
        +- +folder -+   +-¦   +------------ sequence ------------+   +-+
                          +---¦  one of                          +---+
                              ¦ +-------+   +------------------+ ¦ ¦
                            ¦ +-¦ num   +---¦      one of      +-+ ¦
                            ¦   ¦ first ¦   ¦¦+--------------+ ¦   ¦
                            ¦   ¦ prev  ¦   +-¦ :num   -prev +-+   ¦
                            ¦   ¦ cur   ¦     ¦ :+num  -cur  ¦     ¦
                            ¦   ¦ .     ¦     ¦ :-num  -.    ¦     ¦
                            ¦   ¦ next  ¦     ¦ -num   -next ¦     ¦
                            ¦   ¦ last  ¦     ¦ -first -last ¦     ¦
                            ¦   +-------+     +--------------+     ¦
                            +--------------------------------------+

    +--------------------+   +-- -noheader --+   +-- -noclear --+
 ---¦       one of       +---¦    one of     +---¦    one of    +---
    ¦ +----------------+ ¦   ¦ +-----------+ ¦   ¦ +----------+ ¦
    +-¦ -form file     +-+   +-¦ -header   +-+   +-¦ -clear   +-+
      ¦ -format string ¦       ¦ -noheader ¦       ¦ -noclear ¦
      +----------------+       +-----------+       +----------+

    +-- -noreverse --+  +----------------+
 ---¦ +------------+ +--¦                +--¦
    ¦ ¦ -reverse   ¦ ¦  +-- -width num --+
    +-¦ -noreverse +-+
      +------------+


scan --- -help ---¦
```

```
----------------
¦ Do not put a blank between these items.
```

Note: This command does not have MBCS support.

Description

The **scan** command is used to display information about the messages in a specified folder. **scan** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **scan** command displays a line of information about each specified message in the specified folder. Each line gives the message number, the date, the sender, the subject, and as much of the message body as possible. If a + symbol is displayed after the message number, the message is the current message for the folder. If a - symbol is displayed, you have replied to the message. If a * symbol is displayed after the date, the **Date:** field was not present and the displayed date is the last date the message was changed.

Flags

-clear	Clears the display after sending output. **scan** uses the values of the **$TERM** and **$TERMCAP** environment variables to determine how to clear the display. If standard output is not a display, **scan** sends a formfeed character after sending the output.
+*folder msgs*	Displays information about each specified message in the specified folder. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default folder is the current folder. If a folder is specified, it becomes the current folder. The default for *msgs* is **all**.

-form *file*	Displays the **scan** command output in the alternate format described by *file*
-format *string*	Displays the **scan** command output in the alternate format described by *string* The default format is given as an example in the *AIX Operating System Technical Reference* under "mh-format".
-header	Displays a heading that lists the folder name and the current date and time.
-help	Displays help information for the command.
-noclear	Does not clear the terminal after sending output. This is the default.
-noheader	Does not display a heading. This is the default.
-noreverse	Does not reverse the order the messages are displayed in. This is the default.
-reverse	Reverses the order messages are displayed in.
-width *num*	Sets the number of columns in the **scan** command output. The default is the width of the display.

Profile Entries

Alternate-Mailboxes:	Specifies your mailboxes.
Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.

Files

.profile	
$HOME/.mh_profile	The MH user profile.

Related Information

See other MH commands: "inc" in topic 1.1.206, "pick" in topic 1.1.316, "show" in topic 1.1.423.

See the **mh-format** and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.410 sccsdiff

Purpose
Compares two versions of a Source Code Control System (SCCS) file.

Syntax

```
                                 +------------+
sccsdiff -- -r SID1 -- -r SID2 --¦ +--------+ +-- file --¦
                                 +-¦ -p     +-+        ¦
                                   ¦ -s nu  ¦¦  +------+
                                  ¦+--------+¦
                                  +----------+
```

Description
The **sccsdiff** command reads two versions of an SCCS file, compares them, and writes to standard output the differences between the two versions. Any number of SCCS files can be specified, but the same arguments apply to all files.

Flags

-p	Pipes the output through **pr**.
-rSID1	Specifies **SID1** as one delta of the SCCS file for **sccsdiff** to compare.
-rSID2	Specifies **SID2** as the other delta of the SCCS file for **sccsdiff** to compare.
-snum	Specifies the file segment size for **bdiff** to pass to **diff**. This is useful when **diff** fails due to a high system load.

Related Information

See the following commands: "bdiff" in topic 1.1.37, "get" in topic 1.1.186, "sccshelp" in topic 1.1.411, and "pr" in topic 1.1.322.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.411 sccshelp

Purpose
Provides information about a Source Code Control System (SCCS) message, command, or certain non-SCCS commands.

Syntax

```
            +- errorcode -+
sccshelp ---¦             +---¦
            +-- comman  --+ ¦
          +-----------------+
```

Note: This command does not have MBCS support.

Description
The **sccshelp** command writes to standard output information about the use of a specified SCCS **command** or about messages generated while using the commands. Each message has an associated **errorcode**, which can be supplied as a argument to the **sccshelp** command. Zero or more arguments may be supplied. If you do not supply a argument, **sccshelp** prompts for one. You may include any of the SCCS commands as arguments to **sccshelp**.

The **errorcode** consists of numbers and letters, and is found at the end of the message. For example, in the message **no ID keywords (ge6)**, the error code is **ge6**.

Files

/usr/lib/help	Directory containing files of message text.
/usr/lib/help/helploc	File containing locations of help files not in **/usr/lib/help**.

Related Information

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.412 script

Purpose
Makes typescript of terminal session.

Syntax

```
          +------+   +--------+
script ---¦      +---¦        +---¦
          +- -a -+   +- file -+
```

Description

The **script** command makes a typescript of everything printed on your terminal. The typescript is written to **file**, or appended to **file** if the **-a** option is given. It can be sent to the line printer later. If no file name is given, the typescript is saved in the file **typescript**.

The script ends when the forked shell exits.

This program is useful when using a CRT and a hard-copy record of the dialog is desired, as for a student handing in a program that was developed on a CRT when hard-copy terminals are in short supply.

The **script** command, when run on a block mode terminal such as the 3270 console of an AIX/370 system, will echo characters only as they are sent to the system. This is usually line at a time rather than character at a time. If logging of keystrokes on the console is desired, the **tlogger** command is recommended in favor of the **script** command.

Related Information

See the following commands: "tlogger" in topic 1.1.475, "tlog" in topic 1.1.474, and "tee" in topic 1.1.467.

1.1.413 sdiff

Purpose
Compares two files and displays the differences in a side-by-side format.

Syntax

```
        +--------+   +- -w130 --+   +--------------+
sdiff ---¦ one of +---¦          +---¦              +-- file1 -- file2 --¦
        ¦ +----+ ¦   +- -w num -+   +- -o outfile -+
        +-¦ -l +-+
          ¦ -s ¦
          +----+
```

Note: This command does not have MBCS support.

Description
The **sdiff** command reads **file1** and **file2**, uses **diff** to compare them, and writes the results to standard output in a side-by-side format. **sdiff** displays each line of the two files with a series of blanks between them if the lines are identical, a **<** (less than sign) in the field of blanks if the line only exists in **file1**, a **>** (greater than sign) if the line only exists in **file2**, and a **|** (vertical bar) for lines that are different.

When you specify the **-o** flag, **sdiff** produces a third file by merging **file1** and **file2** according to your instructions.

Flags

-l Displays only the left side when lines are identical.

-o outfile Creates a third file, **outfile**, by a controlled line-by-line merging of **file1** and **file2**. The following subcommands govern the creation of this file:

- **l** Adds the left side to **outfile**.
- **r** Adds the right side to **outfile**.
- **s** Stops displaying identical lines.
- **v** Begins displaying identical lines.
- **e l**
- **e r**
- **e b** Starts **ed** with the left side, the right side, both sides, or an empty file, respectively.
- **e** Each time you exit from **ed**, **sdiff** writes the resulting edited file to the end of **outfile**. If you fail to save the changes before exiting, **sdiff** writes the initial input to **outfile**.
- **q** Exits the program.

-s Does not display identical lines.

-w num	Sets the width of the output line to **num**, 130 characters, by default.

Examples

1. To print a comparison of two files:

   ```
   sdiff  chap1.bak  chap1  |  print
   ```

   This prints a side-by-side listing that compares each line of **chap1.bak** and **chap1**. The **| print** sends the listing to the **print** command. **sdiff** assumes that your printer has wide paper (130 columns).

2. To display only the lines that differ:

   ```
   sdiff  -s  -w 80  chap1.bak  chap1
   ```

   This displays the differences at the work station. The **-w 80** sets page width to 80 columns. The **-s** flag tells **sdiff** not to display lines that are identical in both files.

3. To selectively combine parts of two files:

   ```
   sdiff -s  -w 80  -o chap1.combo  chap1.bak  chap1
   ```

   This combines **chap1.bak** and **chap1** into a new file called **chap1.combo**. For each group of differing lines, **sdiff** asks you which group to keep or whether you want to edit them using **ed**.

Related Information

See the following commands: "diff" in topic 1.1.124 and "ed, red" in topic 1.1.147.

1.1.414 sec2prim

Purpose

Converts a secondary or backbone copy of a replicated system into a primary copy of a replicated system.

Syntax

```
             +--------+
sec2prim ---¦ +----+ +--- device ---¦
             +-¦ -a +-+
               ¦ -  ¦¦
              ¦+----+¦
              +-------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **sec2prim** program converts a secondary or backbone copy of a replicated file system into a primary copy of a replicated file system.

A secondary copy of a replicated file system need not store a copy of every file in the primary file system. Typically the secondary copies of a replicated file system as stored on a 386 site, will only include files that are pertinent to 386 operation, and will exclude 370 files. The directory entries in the secondary file system contain the inode numbers for files which are not stored in the secondary file system. The inodes will indicate that the file is stored locally. The **sec2prim** program removes directory entries which point to inodes of files that are not stored in the secondary file system. After all the directories have been updated, the free inode list and the committed transaction list will be rebuilt. The file system has now become the primary copy of a replicated file system.

The **sec2prim** program parses the command line arguments, sets the appropriate switch values, and maintains a pointer to the device name to use. If an invalid switch is specified or a device is not specified, **sec2prim** generates the usage message.

The **sec2prim** program allocates memory for I/O buffers and creates pointer to buffers guaranteeing that the buffers are allocated on 4K boundaries.

Each disk device has both a block and raw device name. If the file system stored on the device is mounted, **sec2prim** uses the block device name. Otherwise the raw device name is used.

The **sec2prim** program opens the device and verifies that the device contains a secondary or backbone copy of a replicated file system. If an invalid device or file system is specified, **sec2prim** displays an error message and aborts the program.

If file replication updates are pending, **sec2prim** generates a warning to the user and asks if the program should continue.

The **sec2prim** program first reads each inode in the file system and stores a code indicating the status of each inode. The status codes are:

IS_NOTSTO	Not Stored
IS_NOTDIR	Not Directory
IS_NMLDIR	Normal Directory
IS_HIDDIR	Hidden Directory

The **sec2prim** program then scans the status array looking for inodes pointing to directories. When an inode pointing to a directory is found, **sec2prim** reads in the directory and scans each directory entry to determine if it points to a file stored in the file system. If the file is not stored in the file system, the directory entry is deleted and the directory block is updated on disk.

After all the directory entries are updated, **sec2prim** updates the superblock by setting the low and high water marks to the previous high water mark, rebuilds the free inode list and then rewrites the superblock to disk.

Secondary filesystems only store directories and files which have been properly marked with the store (for user-replicated filesystems) or chfstore (for system-replicated filesystems) commands. Care should be taken that all files which need to be saved are stored on the secondary filesystem. If parent directories of stored files are not correctly marked, the stored files on the secondary filesystem will be reattached by fsck in the lost+found directory. The lost+found directory itself must be stored on the secondary, or no unreferenced files can be reattached. Backbone filesystems store all files by design, so this marking is not an issue there.

Note that fsck will need to be run to finish the conversion of the secondary or backbone filesystem begun by **sec2prim**. In the case where extensive rebuilding of a secondary filesystem is required, fsck may need to run more than once. In all cases, it is better to run **sec2prim** and fsck on a quiescent filesystem (unmounted or in single user or system maintenance mode), to insure the integrity of stored filesystem data.

Flags

-a list all files to be removed by inode and filename. If this flag is omitted, list the inode and filename of files to be removed which have the filename i386 or are not in a hidden directory.

-u update disk. If this flag is omitted, the changes are not written to disk.

Examples

Before using **sec2prim**, the filenames should be examined to ensure that all critical files have been replicated in the local file system. The full path can be determined from the inode using ncheck.

```
     % ncheck -i 7155  /dev/hd02

     /dev/hd02:
     7155       /bin/join@/i386

    % sec2prim /dev/hd02

   Device /dev/hd02
     Volume Name              VOLXXX
```

```
            File system Name         (1) /
            Device Size              30720000
            I_Nodes                      9984
            Fstore Flags             00000e00
```

The output below shows the inode number and the file name of the files which will be removed. The following files would be removed if the **-u** was specified.

```
            3583   mp
            7155   i386
            2344   . .
      1712   Stored and not linked
```

Error Conditions

The following error conditions can occur. Any of these cause the program to abort immediately.

```
            Unable to open source device
              Open failed on source device.

            Invalid Header in Super Block
              The superblock in the file system did not contain the magic r
              indicating it was a superblock.

            Primary Pack of File system
              The source device is already a primary copy of a replicated
            Non Replicated File system
              The source device is not a replicated file system.

            Unable to allocate status table
              There was insufficient RAM to allocate a status table for each

            Disk seek error
            Disk read error
            Disk write error
```

Related Information

See the following commands: "fsck, dfsck" in topic 1.1.177, "rdevcvt" in topic 1.1.359, "store" in topic 1.1.443, and "chfstore" in topic 1.1.64.

See *AIX Administration Guide*: Chapter 3: "File System Backup."

See *AIX Technical Reference Vol II*: "fs" and "inode" in File Formats.

1.1.415 sed

Purpose
Provides a stream editor.

Syntax

```
       +------+   +------ "script" -------+   +------------+
sed ---¦      +---¦ +- -e --- "script" -+ +---¦            +---¦
       +- -n -+   +-¦                   +-+   +--- file ---+
                    +- -  --- sfile ----+¦              ¦
                   +---------------------+     +--------+
```

Description
The **sed** command modifies lines from the specified **file** according to an edit script and writes them to standard output. The **sed** command includes many features for selecting lines to be modified and making changes only to the selected lines.

The **sed** command uses two work spaces for holding the line being modified: the **pattern space**, where the selected line is held, and the **hold space**, where a line can be stored temporarily.

An edit script consists of individual subcommands, each one on a separate line. The general form of **sed** subcommands is:

[**address-range**] **function[modifiers**]

The **sed** command processes each input **file** by reading an input line into a pattern space, applying all **sed** subcommands in sequence whose addresses select that line, and writing the pattern space to standard output. It then clears the pattern space and repeats this process for each line in the input **file**. Some of the subcommands use a hold space to save all or part of the pattern space for subsequent retrieval.

When a command includes an address, either a line number or a search pattern, only the addressed line or lines is affected by the command. Otherwise, the command is applied to all lines.

An address is either a decimal line number, a **$** (dollar sign), which addresses the last line of input, or a context address. A context address is a regular expression similar to those used in **ed** except for the following differences:

You can select the character delimiter for patterns. The general for of the expression is:

\**?pattern?**

where **?** is a character delimiter you select. This delimiter must be a 1-byte character. The default form for the pattern is:

/**pattern**/

The sequence **\n** matches a new-line character in the pattern space, except the terminating new line.

A . (dot) matches any character except a terminating new-lin character. That is, unlike **ed**, which cannot match a new-line character in the middle of a line, **sed** can match a new-line character in the pattern space.

Certain commands allow you to specify one line or a range of lines to which the command should be applied. These commands are called addressed commands. The following rules apply to addressed commands:

A command line with no address selects every line

A command line with one address, expressed in context form, select each line that matches the address.

A command line with two addresses separated by commas selects th entire range from the first line that matches the first address through the next line that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter the process is repeated, looking again for the first address.

Notes:

1. The **text** parameter accompanying the **a**, **c**, and **i** commands can continue onto more than one line provided all lines but the last end with a \ to quote the new-line character. Backslashes in text are treated like backslashes in the replacement string of an **s** command and can be used to protect initial blanks and tabs against the stripping that is done on every script line. The **rfile** and **wfile** parameters must end the command line and must be preceded by exactly one blank. Each **wfile** is created before processing begins.

2. The **sed** command can process up to 99 commands in a pattern file.

Flags

-e "script"	Uses the text "**script**" as the editing script. If you are using just one **-e** flag and no **-f** flag, the **-e** flag may be omitted.
-f sfile	Uses **sfile** as the source of the edit script. **sfile** is a prepared set of editing commands to be applied to **file**.
-n	Suppresses all information normally written to standard output.

Subcommands

In the following list of functions, the maximum number of permissible addresses for each function is indicated in parentheses. The **sed** script subcommands are as follows:

(1) **a**
text Places **text** on the output before reading the next input line.

(2)**b[label]** Branches to the : command bearing the **label**. If **label** is empty, it branches to the end of the script.

(2)**c**
text Deletes the pattern space. With 0 or 1 address or at the end of a 2-address range, places **text** on the output. Starts the next cycle.

(2)**d** Deletes the pattern space. Starts the next cycle.

(2)**D** Deletes the initial segment of the pattern space through the first new-line character. Starts the next cycle.

(2)**g** Replaces the contents of the pattern space by the contents of the hold space.

(2)**G** Appends the contents of the hold space to the pattern space.

(2)**h** Replaces the contents of the hold space by the contents of the pattern space.

(2)**H** Appends the contents of the pattern space to the hold space.

(1)**i**
text Writes **text** to standard output before reading the next line into the pattern space.

(2)**l** Writes the pattern space to standard output showing nondisplayable characters as two-digit octal values. Long lines are folded.

(2)**n** Writes the pattern space to standard output. Replaces the pattern space with the next line of input.

(2)**N** Appends the next line of input to the pattern space with an embedded new-line character. (The current line number changes.) You can use this to search for patterns that may be split onto two lines.

(2)**p** Writes the pattern space to standard output if a replacement was made and the default output has been suppressed (by using the **-n** option on the command line or the **#n** command in the script).

(2)**P** Writes the initial segment of the pattern space through the first new-line character to standard output.

(1)**q** Branches to the end of the script. Does not start a new cycle.

(1)**r rfile**
Reads the contents of **rfile**. Places contents on the output before reading the next input line.

(2)s**/pattern/replacement/flags**
Substitutes the **replacement** string for the first occurrence of the **pattern** in the pattern space. Any character appearing after the **s** can substitute for the **/** separator.

You can add zero or more of the following **flags**:

g Substitutes all non-overlapping instances of the **pattern** rather than just the first one.

p Writes the pattern space to standard out if a replacement was made.

w wfile
Writes the pattern space to **wfile** if a replacement was made.

Appends the pattern space to **wfile**. If **wfile** was not already created by a previous write by this **sed** script, **sed** creates it.

(2)**tlabel**
Branches to **:label** in the script file if any substitutions were made since the most recent reading of an input line execution of a **t** subcommand. If you do not specify **label**, control transfers to the end of the script.

(2)**wwfile**
Appends the pattern space to **wfile**.

(2)**x** Exchanges the contents of the pattern space and the hold space.

(2)**y/pattern1/pattern2/**
Replaces all occurrences of characters in **pattern1** with the corresponding characters **pattern2**. The byte lengths of **pattern1** and **pattern2** must be equal.

(2)**!sed-cmd**
Applies the specified **sed** subcommand only to lines **not** selected by the address or addresses.

(0)**:label**
This script entry simply marks a branch point to be referenced by the **b** and **t** commands. This label can be any sequence of eight or fewer bytes.

(1)**=** Writes the current line number to standard output as a line.

(2)**{subcmd**
.
.
.

} Groups subcommands enclosed in **{}** (braces).
# If a # appears as the first character on the first line of a script file, then that entire line is treated as a comment, with one exception. If the character after the # is an **n**, then the default output is suppressed. The rest of the line after **#n** is also ignored. A script file must contain at least one non-comment line.

Examples

1. To perform a global change:

```
sed  "s/happy/enchanted/g"  chap1  >chap1.new
```

This replaces each occurrence of **happy** found in the file **chap1** with **enchanted**, and puts the edited version in a separate file named **chap1.new**. The **g** at the end of the **s** subcommand tells **sed** to make as many substitutions as possible on each line. Without the **g**, **sed** replaces only the first **happy** on a line.

The **sed** stream editor operates as a filter. It reads text from standard input or from the files named on the command line (**chap1** in this example), modifies this text, and writes it to standard output. Unlike most editors, it does not replace the original file. This makes **sed** a powerful command when used in pipelines.

2. To use **sed** as a filter in a pipeline:

   ```
   pr  chap2 | sed  "s/Page  *[0-9]*$/(&)/" | print
   ```

   This encloses the page numbers in parentheses before printing **chap2**. The **pr** command puts a heading and page number at the top of each page, then **sed** puts the page numbers in parentheses, and the **print** command prints the edited listing.

   The **sed** pattern **/Page *[0-9]*$/** matches page numbers that appear at the end of a line. The **s** subcommand changes this to **(&)**, where the **&** (ampersand) stands for the page number that was matched.

3. To display selected lines of a file:

   ```
   sed  -n  "/food/p" chap3
   ```

   This displays each line in **chap3** that contains the word **food**. Normally, **sed** copies every line to standard output after it is edited. The **-n** flag stops **sed** from doing this. You then use subcommands like **p** to write specific parts of the text. Without the **-n**, this example would display **all** the lines in **chap3**, and it would show each line containing **food** twice.

4. To perform complex editing:

   ```
   sed  -f  script.sed  chap4  >chap4.new
   ```

   It is always a good idea to create a **sed** script file when you want to do anything very complex. You can then test and modify your script before using it. You can also reuse your script to edit other files. Create the script file with an interactive text editor.

5. A sample **sed** script file:

   ```
   :join
   /\\$/{N
   s/\\\n//
   b join
   }
   ```

   This **sed** script joins each line that ends with a \ (backslash) to the line that follows it. First, the pattern **/\\$/** selects a line that ends with a \ for the group of commands enclosed in **{ }**. The **N** subcommand then appends the next line, imbedding a new-line character. The **s/\\\n//** deletes the \ and imbedded new-line character. Finally, **b join** branches back to the label **:join** to check for a \ at the end of the newly joined line. Without the branch, **sed** writes the joined line and read the next one before checking for a second \.

   Note: The **N** subcommand causes **sed** to stop immediately if there are no more lines of input (that is, if **N** reads the end-of-file character). It does not copy the pattern space to standard output before stopping. This means that if the last line of the input ends with a \, it is not copied to the output.

Related Information

See the following commands: "awk, nawk, oawk" in topic 1.1.29, "ed, red" in topic 1.1.147, and "grep, egrep, fgrep" in topic 1.1.193.

1.1.416 send

Purpose
Sends a message.

Syntax

```
        +-----------------------+   +-----------------+
send ---¦        one of         +---¦ +-------------+ +---
        ¦ +-------------------+ ¦   +-¦ -alias file +-+
        +-¦ file              +-+     ¦ -width num  ¦
          ¦ -draft            ¦       +-------------+
          ¦ -draftmessage msg ¦
          +-------------------+

    +---------------------------------+
 ---¦ +-----------------------------+ +---¦
    +-¦ -[no]draft folder + folder  +-+
      ¦ -[no]filter file            ¦
      ¦ -[no]format file            ¦
      ¦ -[no]forward                ¦
      ¦ -[no]push                   ¦
      ¦ -[no]verbose                ¦
      ¦ -[no]watch                  ¦
      ¦ -[no]msgid                  ¦
      +-----------------------------+

send --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **send** command is used to route messages to the mail delivery system by way of the **post** command or the **spost** command. **send** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **send** command causes **From:** and **Date:** fields to be added to each specified message. **send** places the sender's address in the **From:** field unless a **$SIGNATURE** environment variable is set or a **signature:** profile entry is present. In either of these cases send uses the signature. **send** puts the current date in the **Date:** field. If the **dist** command calls **send**, **send** prepends **Resent-** to the **From:**, **Date:**, and **Message-ID:** fields.

The **send** command then takes each of the specified message files and calls the **post** command or the **spost** command to deliver them. After successful delivery, **send** renames the message file by placing a comma in front of the file name. This comma removes the message from the folder without actually deleting the file. The file is retrievable until you send another message. If the delivery fails, **send** displays an error message.

Flags

-alias *file*	Specifies that *file* is a mail alias file to be searched for aliases. The default alias file is **/usr/lib/mh/MailAliases**.
-draft	Uses the current draft message if no file is

specified. Without this flag, **send** asks the user if the current draft message is the one to use when no file is specified.

-draftfolder +*folder*	Specifies the draft folder that contains the draft message to be sent. If **-draftfolder +***folder* is followed by *msg*, *msg* represents the **-draftmessage** attribute.
-draftmessage *msg*	Specifies the draft message to be sent. You can use one of the following message references as *msg*:

num	*sequence*	**firs**
prev	**cur**	**.**
next	**last**	**new**

The default draft message is **cur**.

-filter *file*	Uses the format instructions in the specified file to reformat copies of the message sent to **Bcc:** recipients.
-format	Puts all recipient addresses in a standard format for the delivery transport system. This flag is the default.
-forward	Adds a failure message to the draft message and returns it to the sender if the **send** command fails to deliver the draft. This flag is the default.
-help	Displays help information for the command.
-nodraftfolder	Undoes the last occurrence of **-draftfolder +***folder*. This flag is the default.
-nofilter	Removes the **Bcc:** header from the message for recipients listed in the **To:** and **cc:** fields, and sends the message with minimal headers to recipients listed in the **Bcc:** field. This flag is the default.
-noformat	Does not alter the format of the recipient addresses.
-noforward	Does not return the draft message to the sender if delivery fails.
-nopush	Runs the **send** command in the foreground (see the **-push** flag). This flag is the default.
-noverbose	Does not display information during the delivery of the message to the **sendmail** command. This flag is the default.
-nowatch	Does not display information during delivery by the **sendmail** command. This flag is the default.
-push	Runs the **send** command in the background. **send** does not display error messages on the terminal if delivery fails. You can use the **-forward** flag to

	return messages to you that fail on delivery.
-verbose	Displays information during the delivery of the message to the **sendmail** command. This information allows you to monitor the steps involved.
-watch	Displays information during the delivery of the message by the **sendmail** command. This information allows you to monitor the steps involved.
-width *num*	Sets the width of components that contain addresses. The default is 72 columns.

Profile Entries

Draft-Folder:	Sets your default folder for drafts.
mailproc:	Specifies the program used to post failure notices.
Path:	Specifies your **user_mh_directory**.
postproc:	Specifies the program used to post messages.
Signature:	Sets your mail signature.

Files

.profile	
$HOME/.mh_profile	The MH user profile.

Related Information

See the following commands: "ali" in topic 1.1.17, "comp" in topic 1.1.85, "dist" in topic 1.1.131, "forw" in topic 1.1.174, "post" in topic 1.1.320, and "sendmail, mailq, newaliases" in topic 1.1.417.

See the **mh-alias**, **mh-format**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.417 sendmail, mailq, newaliases

Purpose
Routes mail for local or network delivery.

Syntax

```
+--------------------+   +----------- default ¦------------+
¦ /usr/lib/sendmail   +---¦ +---------+   +-------------+   +--- address ---¦
¦ /usr/lib/mailq      ¦   +-¦ one of  +---¦ -C file     +---+             ¦
¦ /usr/lib/newaliases ¦     ¦ +-----+ ¦   ¦ -d X        ¦ ¦   +-----------+
+--------------------+     +-¦ -ba +-+ ¦ ¦ -f name     ¦ ¦
                              ¦ -bd ¦   ¦ ¦ -F fullname ¦ ¦
                              ¦ -bi ¦   ¦ ¦ -h N        ¦ ¦
                              ¦ -bm ¦   ¦ ¦ -n          ¦ ¦
                              ¦ -bp ¦   ¦ ¦ -o x[value] ¦ ¦
                              ¦ -bs ¦   ¦ ¦ -q [time]   ¦ ¦
                              ¦ -bt ¦   ¦ ¦ -r name     ¦ ¦
                              ¦ -bv ¦   ¦ ¦ -t          ¦ ¦
                              ¦ -bz ¦   ¦ ¦ -v          ¦ ¦
                              +-----+   ¦ +-------------+ ¦
                                        +-----------------+
```

```
----------------
¦ default = -bm for sendmail
            -bp for mailq
            -bi for newaliases
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **sendmail** command receives formatted text messages and routes the message to one or more other users on the local system, or if connected to a network, to users on other systems. The program translates the format of message heading information to match the requirements of the destination system. It determines which network to use based on the syntax and content of the addresses.

The program can deliver messages to:

Users on the local syste
Users connected to the local system using the TCP/IP protoco
Users connected to the local system using the **uucp** protocol.

The **sendmail** command operates mainly as a background, mail-routing program. Other programs, such as **Mail** and the message handler routines, provide user interfaces for generating and receiving mail that **sendmail** handles. However, if you enter the **sendmail** command with no flags, it reads standard input for the message text until it receives a **Ctrl-D** or a line with only a single period, designating the end of the message. Then it sends a copy of the message to all addresses listed. For example, the following input at the command line sends the message **This is a test message** to the mail box for user **george** on the local system:

```
$ /usr/lib/sendmail george
This is a test message
.
$
```

The **sendmail** command can be run to service TCP/IP mail delivery either as a daemon (**-bd** flag) or automatically by the **inetd** daemon (**-bn** flag specified in the file **/etc/inetd.conf**). If multiple **sendmail** commands are running at the same time, only one of these may be servicing TCP/IP mail delivery; that is, at most one **sendmail** process using either the **-bd** or **-bn** flag may be run concurrently. The choice of whether to run **sendmail** from **inetd** (the default when TCP/IP is installed) or as a daemon depends on the amount of TCP/IP mail traffic you expect. If the load is heavy, running the **sendmail** command as a daemon avoids the start-up costs associated with multiple invocations by **inetd**. To disable **sendmail** being run by **inetd**, comment out the **smtp** entry in **/etc/inetd.conf**.

The **sendmail** command uses a configuration file (**/usr/adm/sendmail/sendmail.cf** by default) to set many operational parameters and to determine how the program parses addresses. This file is a text file that you can edit. However, **sendmail** uses a frozen form of this file (**/usr/adm/sendmail/sendmail.fc**). For any changes made to this file to be effective, you must build a new copy of the frozen configuration file by running **sendmail** with the **-bz** flag. This must be done on each CPU separately.

The **sendmail** command also allows you to define aliases to use when addressing mail handled by the local **sendmail** command. **Aliases** are alternate names that can be used in place of elaborate network addresses. You can also use aliases to build distribution lists. Define aliases in **/usr/adm/sendmail/aliases**. This file is a text file that you can edit. However, **sendmail** uses a data base version of this file that is kept in the directory. For any changes made to the aliases file to be effective, you must build a new alias data base by running **sendmail** with the **-bi** flag. If the **sendmail** daemon is running, you must also stop that process and start the daemon again before it recognizes the new alias data base file. Normally the sender of a message is not included when **sendmail** expands an alias address. For example, if **amy** sends a message to alias **D998** and she is defined as a member of that alias, **sendmail** does not send a copy of the message to **amy**.

Every system must have a user or user alias designated **postmaster**. Assign this alias in the file **/usr/adm/sendmail/aliases**. Unless you change the entry in this file, this alias is assigned to **root**. This alias allows other users outside your system to send mail to a known ID (for example **postmaster**) to get information about mailing to users on your system. Also, users on your system can send problem notifications to this ID.

Two additional commands are links to **sendmail**:

/usr/lib/mailq	Prints the contents of the mail queue. This command is the same as running **sendmail** with the **-bp** flag.
/usr/lib/newaliases	Builds a new copy of the alias data base from the file **/usr/adm/sendmail/aliases**. This command is the same as running **sendmail** with the **-bi** flag.

Subtopics

1.1.417.1 Mail Addresses

Mail addresses are based on the domain address (Arpanet) protocol. These addresses have the following form:

user@host.domain

Note: The configuration file provided with **sendmail** specifies that blanks in addresses be converted to periods before being transmitted. This convention follows the Arpanet mail protocol as described in RFC822, but does not match the Arpanet mail protocol as described in RFC733 (NIC41952). You can change this setting by setting the **OB** option in the **sendmail** configuration file.

A **domain** is a logical grouping of systems that are connected together by physical network links. No direct relationship exists between the actual physical interconnections and the way in which the systems are grouped in the domain. The **domain name** identifies a specific domain within a larger group of domains. The domain name has the format of a tree structure. Each node (or leaf) on the tree corresponds to a resource set, and each node can create and contain new domains below it. The actual domain name of a node is the path from the root of the tree to that node. Domain names do not correspond to system names, host addresses, or any other type of information.

For example, if node **hera** is part of the domain **IBM**, which is in turn a subdomain of **COM**, and a message is sent to **geo** at that address it is sent to:

```
  geo@hera.IBM.COM
```

The message router (usually **sendmail**) must determine how to send the message to its final destination. If the router is at **hera**, it delivers the message to user **geo**. If the router is at another system within the **IBM** domain, it corresponds with the name server for that domain to find out how to deliver the message. If the router is not a part of the **IBM** domain, but is in a domain that is under the **COM** domain, it corresponds with the name server for the **COM** domain to find out how to deliver the message. The respective name server returns a network address to the router. That network address determines the actual path that the message takes to its destination.

The domain address is read from right to left, with each domain in the address separated from the next domain with a . (period). This format does not imply any routing. Thus, although the example is specified as a **COM** address, the message might actually travel by a different route if that were more convenient or efficient. At some sites, the message associated with the sample address might go directly from the sender to node **hera** over a local area network. At other sites it might be sent over a **uucp** network or a combination of other delivery methods.

Normally, the actual routing of a message is handled automatically. However, you can route the message manually through several specified hosts to get it to its final destination. An address using intermediate hosts, called a **route address**, has the following form:

@hosta,@hostb:user@hostc

This address specifies that the message should go first to the remote system represented by **hosta**, then to the remote system represented by

hostb, and finally to the remote system represented by **hostc**. This path is forced even if there is a more efficient route to **hostc**.

In some cases the user may abbreviate the address rather than typing the entire domain name. In general, systems in the same domain do not need to use the full domain name. For example, a user on node **zeus.IBM.COM** can send a message to **geo@hera.IBM.COM** by typing only **geo@hera**, because they are in the same local domain, **IBM.COM**.

Other mail address formats exist that are used by other mail routing programs (such as, **uucp**). The mail routing program (**sendmail**) converts most of these other formats to a format that the network routing system can use. However, if you use the domain address format, the routing program operates more efficiently.

For example, if **sendmail** receives an address in the following format:

@host:user

It converts it to the corresponding domain address format:

user@host

Similarly, if **sendmail** receives an address in the following format:

host!user

the mail routing program routes the message directly to the **uucp** command (part of the Basic Networking Utilities (BNU)). However, when sending mail via **uucp**, you must include a route address that indicates which BNU host(s) to send the message through to get to the final destination.

To route messages through the BNU network, use one of the following domain address formats. Your choice depends on the way in which the systems at your site are connected:

1. **@system_name.domain_name:uucp-route!user-ID**

   For example, the address:

   ```
   @zeus:hera!amy
   ```

   sends a message to user **amy** on **uucp** host **hera** by way of system **zeus**. The address:

   ```
   @apollo.802:merlin!lgh
   ```

   sends a message to user **lgh** on **uucp** host **merlin** via system **apollo** under the local domain **802**.

2. **uucp-route!user-ID@system_name.domain_name**

   In this case, the address:

   ```
   merlin!arthur!amy@hera.802
   ```

   sends a message to user **amy**. on system **hera** under domain **802**. via the BNU link **merlin**. through **arthur**.

3. **system_name.domain_name:uucp-route!user-ID@system_ name.domain_name**

In this example, the address:

```
  @apollo.802:merlin!arthur!amy@hera.802
```

sends a message to user **amy**. on system **hera**. under domain **802**. that first goes through **apollo**., the gateway node for domain **802**., and then through the BNU link **merlin**. through **arthur**. (Including **802** in this example is optional, since the two domain names are identical.)

4. **hosta!hostb!hostc!user**

   This example is a purely **uucp** route address:

   ```
     zeus!hera!kronos!amy
   ```

   sends a message to **amy**. on **kronos**., via the BNU link **zeus** through **hera**.

5. @**hosta**.UUCP:@**hostb**.UUCP:**user@hostc**

   This example, like the previous one, is a purely **uucp** route address:

   ```
     @zeus.UUCP:@hera.UUCP:amy@kronos.UUCP
   ```

   sends a message to **amy**. on **kronos**., via the BNU link **zeus** through **hera**.

1.1.417.2 Return Codes

The **sendmail** program returns an exit status code. These exit codes are defined in **/usr/include/bsd/sysexits.h**. The following table summarizes the meanings of these return codes:

Return Code	**Meaning**
EX_CANTCREAT	The **sendmail** program cannot create a file that the user specified.
EX_DATAERR	The user's input data was not correct.
EX_DB	A data base was inaccessible or was malformed and unusable.
EX_IOERR	An error occurred during I/O.
EX_NOHOST	The **sendmail** program could not recognize the specified host name.
EX_NOINPUT	The **sendmail** program either could not find, or could not read, the specified input file.
EX_NOPERM	The user does not have the needed permissions to perform the requested operation.
EX_NOUSER	The **sendmail** program could not recognize a specified user ID.
EX_OK	The **sendmail** program successfully completed the operation for all addresses.
EX_OSERR	A temporary operating system error occurred. An example of such an error is **cannot fork because too many processes are currently running**.
EX_OSFILE	A system file error occurred. For example, a system file (such as **/etc/passwd**) does not exist, cannot be opened or has another type of error preventing it from being used.
EX_PROTOCOL	The remote system returned something that was incorrect during a protocol exchange.
EX_SOFTWARE	An internal software error occurred (including bad arguments).
EX_TEMPFAIL	Temporary failure, for example, the **sendmail** program could not create a connection. Try the request again later.
EX_UNAVAILABLE	A service or resource that **sendmail** needed was not available.
EX_USAGE	The command syntax was not correct.

Flags

-ba	Starts **sendmail** in Arpanet mode. All input lines to the program must end with a carriage return and a line feed (CR-LF). The program generates messages with a CR-LF at the end. The program looks at the **From**. and the **Sender**. fields to find the name of the sender.
-bd	Starts **sendmail** to run in the background as a TCP/IP mail routing daemon. The **smtp** line in the **/etc/inetd.conf** file must be commented out before this option will work.
-bi	Builds the alias data base files from information defined in **/usr/adm/sendmail/aliases**. Running **sendmail** with this flag is the same as running the command, **/usr/lib/newaliases**.
-bm	Delivers mail in the usual way. This is the default.
-bn	Starts **sendmail** to be run by **inetd**. This option should be specified only on the **smtp** entry in **/etc/inetd.conf** if TCP/IP mail delivery is to be serviced by **inetd**.
-bp	Prints a listing of the mail queue. Running **sendmail** with this flag is the same as running the command, **/usr/lib/mailq**.
-bs	Uses the simple mail transfer protocol (SMTP) as described in RFC821. This flag implies all of the operations of the **-ba** flag that are compatible with SMTP.
-bt	Starts **sendmail** in address test mode. This mode allows you to enter addresses interactively and watch as **sendmail** displays the steps it takes to parse the address. Use this mode for debugging the address parsing rules in a new configuration file.
-bv	Starts **sendmail** with a request to verify the user IDs provided in the **address** field of the command. The program responds with a message telling which IDs can be resolved to a mailer program. It does not try to collect or deliver a message. Use this mode to validate the format of user IDs, aliases or mailing lists.
-bz	Builds the frozen version of the configuration file from information in **/usr/adm/sendmail/sendmail.cf**.
-Cfile	Starts **sendmail** using an alternate configuration file specified by the **file** parameter. Use this flag together with **-bt** to test a new configuration file before installing it as the running configuration file.
-dX	Sets debugging value to **X**.
-Ffullname	Sets the full name of the sender to be the string provided in the **fullname** parameter.
-fname	Sets the name of the **from** person (the sender of the mail). This flag can be used only by those administrative user IDs designated in the configuration file with the **T** control line, or if your ID is the ID supplied in **name**.
-hN	Sets the hop count to **N**. The **hop count** is the number of

	times that the message has been processed by a **sendmail** program (not just the local copy of **sendmail**). The program increments the hop count every time the message is processed. When it reaches a limit, the message is returned with an error message usually caused by alias looping.
-n	Does not alias.
-ox[value]	Sets option **x**. If the option is a valued option, you must also specify **value**. See the following table for possible options, values and their meanings.
-q[time]	Processes saved messages in the queue at the intervals specified by **time**. If **time** is not specified, this flag processes the queue at once. You can specify **time** as a tagged number using the tag **s** for seconds, **m** for minutes, **h** for hours, **d** for days, and **w** for weeks. If the tag letter is omitted and just a number is specified, **sendmail** uses days as the unit of time. For example, **-q2m** processes the queue every two minutes, but **-q2**. processes the queue every two days.
-rname	An alternate and obsolete form of the **-f** flag.
-t	Reads the **To:**., **Cc:**., and **Bcc:**. lines of the message header to determine where to send the message; deletes the **Bcc:**. line before transmitting the message; and deletes any addresses in the argument list of the **sendmail** command.
-v	Starts **sendmail** in verbose mode. The program displays messages regarding the status of transmission, the expansion of aliases and any errors that may occur during the sending of the message.

You can also set or remove a number of **sendmail** processing options. Normally, the person responsible for the mail system uses these options. To set these options, use the **-o** flag on the command line or the **O** control line in the configuration file (**sendmail.cf**).

Option	**Function**
Afile	Uses the named **file** as an alternate alias file.
Bc	Sets the blank substitution character to the character specified in the parameter **c**. The **sendmail** program replaces unquoted spaces in addresses with this character. The supplied configuration file uses the period (.) for this character.
c	If an outgoing mailer program is specified in the configuration file as being expensive to use, this option causes **sendmail** to queue messages for that mailer program without sending them. The queue can be run later when costs are lowest or when the queue is large enough to send the messages efficiently.
dx	Sets the delivery mode to **x**. Delivery modes are **i** for interactive (synchronous) delivery, **b** for background (asynchronous) delivery, and **q** for queue only (next time the queue is run) delivery. The supplied configuration

file uses a value of **b**.

ex	Sets error processing to mode **x**. Valid modes are:

m	Mails the error message to the user's mail box.
w	Writes the error message to the terminal or mails it if the user is not logged in.
p	Displays the error message on the terminal (default).
q	Throws away error message and returns the exit status only.
e	Mails the error message to the user's mail box, but always exits with a zero exit status (normal return).

If the text of the message is not mailed by modes **m** or **w** and if the sender is a local user, a copy of the message is appended to the file **dead.letter** in the sender's home directory.

f	Saves **From.** lines at the front of messages. These lines are normally discarded.
gN	Sets the default group ID to use when calling mailers to the value specified by **N**.
Hfile	Specifies the name of the SMTP help file (**/usr/adm/sendmail/sendmail.hf** by default).
i	Does not interpret a period (.) on a line by itself as a message terminator.
Ln	Specifies the log level to be the value supplied in the **n** parameter. Valid levels and the activities that they represent are (each number includes the activities of all numbers of lesser value and adds the activity that it represents):

Level	**Activity Logged**
0	Prevents logging.
1	Logs major problems only.
2	Logs message collections and failed deliveries.
3	Logs successful deliveries.
4	Logs messages deferred (for example, because the host is down).
5	Logs messages that are placed in the queue (normal event).
6	Logs unusual but benign incidents (for example, trying to process a locked file).
9	Logs internal queue ID to external message ID mappings. This can be useful for tracing a message as it travels between several hosts.
12	Logs messages that are of interest when debugging.
16	Logs verbose information regarding the queue.

Mx value	Sets the macro **x** to **value**. Use this option from the command line only (with the **-o** flag).

m	Sends to the sender (me) also, if the sender is in an alias expansion. Normally, the sender does not receive a copy of the message.
Nnetname	Sets the name of the host network to **netname**. The **sendmail** program compares the argument of an SMTP **HELO** command to **hostname.netname** (it gets the value of **hostname** from the kernel). If these values do not match, it adds the **hostname.netname** string to the **Received:**. lines in the message so that messages can be traced accurately.
o	This option indicates that this message may have old style headers. Without this option, the message has new style headers (commas instead of spaces between addresses). If this option is set, an adaptive algorithm correctly determines the header format in most cases.
Qdir	Sets the directory in which to queue messages to the directory specified by the **dir** parameter. That directory must exist.
rtime	Sets the timeout for reads from a mailer program to the value specified by **time**. If no timeout value is set, **sendmail** waits indefinitely for a mailer to respond. The default value for this timeout is 5 minutes.
Sfile	Sets the mail statistics file to the **file**. If this file exists, **sendmail** stores statistics about mail traffic in a data base format in this file. Use the **mailstats** command to read the information in this file. If the indicated file does not exist, no statistic information is saved.
s	Interactive mode delivers mail without going through the mail queue. When this option is specified, mail is passed through the mail queue in interactive mode also. This action ensures that the message being sent is not lost if a delivery problem occurs.
Ttime	Sets the timeout on messages in the queue to the specified **time**. After a message has been in the queue for this amount of time, **sendmail** returns the message to the sender. In **sendmail.cf** that is provided with **sendmail**, this value is set to three days.
uN	Sets the default user ID to use when calling mailers to the value specified by **N**.
v	Run in verbose mode.
Y	When this option is specified, **sendmail** delivers each message in the mail queue from a separate process. This option uses less memory to process the mail queue. Use of this option is not recommended.

Files

/usr/lib/sendmail	Contains the **sendmail** program.

/usr/lib/mailq	Displays list of the mail queue.
/usr/lib/newaliases	Builds alias data base.
/usr/lib/mailstats	Displays **sendmail** statistics found in **/usr/adm/sendmail/sendmail.st**.
/usr/adm/sendmail/aliases	Contains the text version of **sendmail** aliases.
/usr/adm/sendmail/aliases.dir	Contains one of the alias data base files.
/usr/adm/sendmail/aliases.pag	Contains one of the alias data base files.
/usr/adm/sendmail/sendmail.hf	Contains the SMTP help file.
/usr/adm/sendmail/sendmail.cf	Contains the text version of the **sendmail** configuration file.
/usr/adm/sendmail/sendmail.fc	Contains the **sendmail** configuration frozen file.
/usr/adm/sendmail/sendmail.st	Contains the **sendmail** statistics file.
/usr/adm/sendmail/smdemon.cleanu	Maintains aging copies of the **log** file in **/usr/spool/mqueue**.
/etc/rc.sendmail	Contains the shell script to start the **sendmail** daemon.
/usr/spool/mqueue	Contains the **log** file and temporary files associated with the messages in the mail queue (the mail queue directory). Temporary files have names that include the mail queue ID (**mqid**) of the message for which the file was created:

df**mqid**	Data file
lf**mqid**	Lock file
nf**mqid**	Backup file
qf**mqid**	Queue control file
tf**mqid**	Temporary control file
xf**mqid**	Transcript file for session.

/usr/spool/cron/crontabs/root	Contains a commented entry to run **sendmail** periodically for use when not routing mail to a network. Uncomment that entry to process the mail queue at the interval specified in that **cron** file.
/bin/uux	Contains the mailer program to deliver **uucp** mail.
/bin/bellmail	Contains the mailer program to deliver local mail.

Related Information

See the following commands: "bellmail" in topic 1.1.38, "mail, Mail" in topic 1.1.253, "uux" in topic 1.1.515.

See the chapter about managing the mail system in *Managing the AIX Operating System*.

See the chapter about sending and receiving mail in *Using the AIX Operating System*.

See the file **sendmail.cf** in *AIX Operating System Technical Reference*.

1.1.418 setmaps

Purpose
Sets terminal maps.

```
              +---------------------------------------------+
              ¦                                             ¦
setmaps ----¦                one of                        +--¦
              ¦ +------+  +------------+  +--------------+ ¦
              +-¦      +--¦ -i mapname +--¦              +-+
                +- -r -+  ¦ -I mapname ¦  +- -k keyname -+
                          ¦ -o mapname ¦
                          ¦ -O mapname ¦
                          ¦ -t mapname ¦
                          +------------+

              one of
              +----+
setmaps ----¦ -h +--¦
              ¦ -c ¦
              +----+
```

Note: This command does not have MBCS support.

Description
The **setmaps** command assigns a terminal map to the standard input device. AIX uses input and output terminal maps to convert multibyte characters in the data stream to the ASCII characters supported by asynchronous terminals. If you call **setmaps** with no arguments, it displays the names of the current input and output terminal maps.

A terminal map is a text file containing a list of rules that associate a pattern string with a replacement string. This file normally resides in the directory **/etc/nls/termmap**. AIX uses an input map file to map input from the keyboard to an application and an output map file to map output from an application to the display.

Terminal mapping works as follows. The system collects characters in a buffer until a pattern specified by a rule in the map file matches a substring in the buffer. The system then constructs and returns the replacement string specified by the rule. This processing continues with the remaining characters in the buffer.

The rules of a terminal map can test and change the ***state*** of the pattern processor. The state is identified by a single-byte character, conventionally a digit (0-9). The state is reset to zero, the initial state, whenever the system loads a new map or flushes the terminal input or output buffer (as when it processes a **kill** or **intr** character or a program issues an **ioctl** system call). A terminal map can use states to do such things as detect multibyte escape sequences. You can test for state **x** by specifying @**x** in a pattern; you can set the state to **x** by including @**x** in a replacement string.

The special name **NOMAP** will clear the corresponding mapping.

File Format
The text of a terminal map file is a set of rules. Each rule has the following format:

pattern:replacement

The **pattern** string can include the following special characters:

?	Matches any single byte
@x	Matches this rule only if the pattern processor is in state **x**, where **x** is any single byte. (This sequence does not match a character in the input buffer.)
\?	
\@	
\\	Prevents the pattern processor from interpreting **?**, @, or \ as special characters.
\ddd	Represents any byte in octal notation.
\xdd	Represents any byte in hexadecimal notation.

The **replacement** string can also include the following special characters:

$**n**	Uses the **n**th character in the input string that matched this pattern, where **n** is a decimal digit.
@**x**	Moves the pattern processor into state **x**. (This sequence does not become part of the replacement string.)
\$	
\@	
\\\\	Prevents the pattern processor from interpreting **$**, @, or \ as special characters.
\ddd	Represents any byte in octal notation.
\x**dd**	Represents any byte in hexadecimal notation.

Flags

-i mapname	Selects **/etc/nls/termmap/mapname.in** as the input map.
-I mapname	Selects **mapname** as the input map, where **mapname** is a file residing in the current directory. You must have superuser authority to specify this flag.
-o mapname	Selects **/etc/nls/termmap/mapname.out** as the output map.
-O mapname	Selects **mapname** as the output map, where **mapname** is a file residing in the current directory. You must have superuser authority to specify this flag.
-t mapname	Selects **/etc/nls/termmap/mapname.in** as the input map and **/etc/nls/termmap/mapname.out** as the output map.
-k keyname	Associates the specified **keyname** with the map being selected (see "stty, STTY" in topic 1.1.447). This key name overrides the default key, which is normally set to the value of

	mapname. You must have superuser authority to specify this flag.
-r mapname	Specifies that the kernel is to reload the specified map file. Terminals using the old map will continue to do so until they are logged off or until their maps are explicitly reset. If you do not specify this flag, a map is loaded only if it has not already been loaded into the kernel. You must have superuser authority to specify this flag.
-h	Display help message.
-c	Clears all mapping for the terminal.

Files

/etc/nls/termmap/*.in	Input map files.
/etc/nls/termmap/*.out	Output map files.

Related Information

See the following commands: "ctab" in topic 1.1.103 and "stty, STTY" in topic 1.1.447.

See the **ports** and **termio** files in *AIX Operating System Technical Reference*.

See "Overview of International Character Support" in *Managing the AIX Operating System*.

1.1.419 setmnt

Purpose

Creates a mount table.

Syntax

```
setmnt ---------------¦
          ¦        ¦
          +-- -a --+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **setmnt** command reads lines from standard input and writes the **/etc/mtab** table (see the **mtab** file in *AIX Operating System Technical Reference*). The **/etc/mtab** is needed for both the **mount** and **unmount** commands. **setmnt** creates an **mtab** entry for each line read. Input lines have the format:

filesys **directory**

where **filesys** is the name of the file system's special file and **directory** is the root name of that file system. Thus, **filesys** and **directory** become the first two strings in the **mtab** entry. The maximum number of characters per **mtab** entry is set to 95.

This command can only be run by the superuser.

Flags

-a	Append to existing **/etc/mtab** rather than overwriting it.

Examples

To set the mount table after it has been destroyed or made invalid:

```
setmnt  <<end
hd1  /u
fd0  /a
fd1  /b
end
```

This command sets the mount table to show **/dev/hd1** mounted on **/u**, **/dev/fd0** on **/a**, and **/dev/fd1** on **/b**.

The **<<end** and **end** define a Here Document, which uses the text entered before the **end** line as the standard input for the **setmnt** command. For more details, see "Inline Input Documents" in topic 1.1.420.19.

Files

/etc/mtab	Record of mounted file systems.

Related Information

See the **mtab** file in *AIX Operating System Technical Reference*.

1.1.420 sh, Rsh

Purpose
Interprets commands read from a file or entered at the keyboard.

Syntax

```
one of
+-----+   +--------+   +-----------------+   +-------------------------------+
¦ sh  +---¦ +----+ +---¦ one of  one of  +---¦           +--------------+   +-
¦ Rsh ¦   +-¦ -i +-+   ¦ +---+ ¦ +-----+ ¦   ¦ +- file --¦              +-+ ¦
+-----+     ¦ -r ¦¦    +-¦ + +---¦ a n +-+   ¦ ¦         +- parameters -+ ¦ ¦
           ¦+----+¦      ¦ - ¦   ¦ e t ¦¦    +-¦      one of            +-+
            +------+    ¦+---+   ¦ f u ¦¦      ¦ +--------------+       ¦
                        ¦        ¦ h v ¦¦     +-¦ -c cmdstring +----------+
                        ¦        ¦ k x ¦¦       ¦ -s           ¦
                        ¦        +-----+¦       +--------------+
                        +---------------+
```

¦ Do not put a blank between these items.

Description
The **sh** command is a system command interpreter and programming language. It is not a part of the operating system **kernel**, but an ordinary user program that reads commands entered at the keyboard and arranges for their execution. In addition, it can read commands that you have saved in a file. Such a file is usually called a **shell procedure** or a **command file**. For a complete description of how to write shell procedures to take advantage of this useful tool, see *Using the AIX Operating System*.

A restricted version of shell (the **Rsh** command) is available that allows you to create user environments with a limited set of privileges and capabilities. See "Restricted Shell" in topic 1.1.420.25 for additional information on the restricted shell.

Subtopics

1.1.420.1 Commands

A command is either a simple command or control command (see "Control Commands" in topic 1.1.420.22).

A simple command is a sequence of words separated by blanks. A blank is a space or a tab. A word is a sequence of characters and/or numerals that contains no blanks unless they occur within quote marks. Japanese characters must be separated by spaces if they are to be treated as separate words. The first word in the sequence usually specifies the name of a command. Any remaining words, with a few exceptions, are passed to that command. The command name is passed as argument 0 (see the **exec** system call in the *AIX Operating System Technical Reference*).

The **value** of a simple command is its exit value if it ends normally or (octal) 200 + **status** if it ends abnormally. For a list of **status** values, see the **sigaction** system call in *AIX Operating System Technical Reference*.

A **pipeline** is a sequence of one or more commands separated by a | (vertical bar) or, for historical compatibility, by a ^ (circumflex). In a pipeline, the standard output of each command but the last becomes the standard input of the next command. Each command runs as a separate process, and the shell waits for the last command in the pipeline to end. A **filter** is a command that reads its standard input, transforms it in some way, then writes it to its standard output. A pipeline normally consists of a series of filters. Although the processes in a pipeline (except the first process) can execute in parallel, they are synchronized to the extent that each program needs to read the output of its predecessor.

The exit value of a pipeline is the exit value of the last command.

A **list** is a sequence of one or more pipelines separated by ; (semicolon), **&** (ampersand), **&&** (two ampersands), or || (two vertical bars) and optionally ended by a ; (semicolon) or an **&** (ampersand). These separators and terminators have the following effects:

;	Causes **sequential execution** of the preceding pipeline (the shell waits for the pipeline to finish).
&	Causes **asynchronous execution** of the preceding pipeline (the shell does **not** wait for the pipeline to finish).
&&	Causes the list following it to be executed **only** if the preceding pipeline returns a zero exit value.
||	Causes the list following it to be executed **only** if the preceding pipeline returns a nonzero exit value.

Note: The **cd** command is an exception. If it returns a nonzero exit value, no subsequent commands in a list are executed, regardless of the separator characters.

The ; and **&** separators have equal precedence, as do **&&** and ||. The single-character separators have lower precedence than the double-character separators. An unquoted new-line character following a pipeline functions the same as a ; (semicolon).

Note: The shell treats as a comment any word that begins with a # character and ignores that word and all characters following up to the next new-line character.

1.1.420.2 Command Execution

Each time the shell executes a command, it carries out the substitutions discussed in the following text. Substitutions are performed in the order they are described below. If the command name matches one of the built-in commands discussed in "Built-in Commands" in topic 1.1.420.23, it executes it in the shell process. If the command name does not match a built-in command but matches the name of a defined function, it executes the function in the shell process. The shell sets the positional parameters $1, $2, etc. to the parameters of the function.

If the command name matches neither a built-in command nor the name of a defined function and the command names an executable file that is a compiled (binary) program, the shell (as **parent**) spawns a new (**child**) process that immediately runs the program. If the file is marked executable but is not a compiled program, the shell assumes that it is a shell procedure. In this case, the shell spawns another instance of itself (a **subshell**), to read the file and execute the commands included in it (note how this differs from the execution of functions). The shell also executes a parenthesized command in a subshell (see page 1.1.420.22). From your point of view as a user, a compiled program is run in exactly the same way as a shell procedure.

The shell parameter PATH defines the search path for the directory containing the command. Alternative directory names are separated by a colon (:). The default path is **/bin:/usr/bin:/usr/ucb/:**. The current directory is specified by a null path name, which can appear immediately after the equal sign or between the colon delimiters anywhere else in the path list. If the command name contains a /, the search path is not used; such commands are not executed by the restricted shell. Otherwise, each directory in the path is searched for an executable file.

You can change the particular sequence of directories searched by resetting the **PATH** variable (page 1.1.420.10).

The shell remembers the location in the search path of each executed command (to avoid unnecessary **exec**s later). If the command was found in a **relative directory** (one whose name does not being with /), the shell must redetermine its location whenever the current directory changes. The shell forgets all remembered locations whenever you change the **PATH** variable or execute the **hash -r** command (page 1.1.420.23).

1.1.420.3 Signals

The shell ignores **INTERRUPT** and **QUIT** signals for an invoked command if the command is terminated with a **&** (that is, if it is running in the background). Otherwise signals have the values inherited by the shell from its parent, with the exception of SIGSEGV (see also the built-in **trap** command on page 1.1.420.23).

1.1.420.4 The .profile File

A shell acts as a "login shell" if it is invoked through the exec system call and the first character of argument zero is - (minus sign). Before a login shell reads your commands, it checks to see if a file named **/etc/profile** exists on the system, and if it does, it reads commands from it (this file should set variables needed by all users). After this, the shell looks for a file named **.profile** in your login directory. If it finds one, it executes commands from it. Finally, the shell is ready to read commands from your standard input.

1.1.420.5 Command Substitution

To capture the output of any command as an argument to another command, place that command line within grave accents (` `). This concept is known as command substitution. The shell first executes the command or commands enclosed within the grave accents, and then replaces the whole expression, grave accents and all, with their output. This feature is often used in assignment statements:

```
  today=`date`
```

This statement assigns the string representing the current date to the variable **today**. The following assignment saves, in the variable **files**, the number of files in the current directory:

```
  files=`ls | wc -l`
```

You can enclose any command that writes to standard output in grave accents. You can nest command substitutions as long as you quote the inside sets of grave accents with a preceding \ (backslash):

```
  logmsg=`echo Your login directory is \`pwd\``
```

1.1.420.6 Substitutions

The character $ is used to introduce substitutable parameters. There are two types of parameters: positional and keyword. The shell has several mechanisms for creating variables (assigning a string value to a name), as described in the following sections.

1.1.420.7 Positional Parameters

When you run a shell procedure, the shell implicitly creates positional parameters that reference each word on the command line by its position on the command line. The word in position **0** (the procedure name), is called **$0**, the next word (the first parameter) is called **$1**, and so on up to **$9**. To refer to command line parameters numbered higher than 9, use the built-in **shift** command (page 1.1.420.23).

You can also assign values to these positional parameters explicitly by using the built-in **set** command (page 1.1.420.23), or with shell functions.

Notes:

1. When an argument for a position is not specified, its positional parameter is set to null.

2. Positional parameters are global and are reset by calling nested shell functions. They are not restored when the function returns.

1.1.420.8 Keyword Parameters

The shell also recognizes alphanumeric variables to which string values can be assigned. You assign a string value to a **name**, as follows:

name=string

A name is a sequence of letters, digits, and underscores that begins with a letter or underscore. To use the value that you have assigned to a variable, add a **$** (dollar sign) to the beginning of its name. Thus, **$name** yields the value **string**. No blanks surround the = (equal sign) in an assignment statement. (Positional parameters cannot appear in a assignment statement; they can only be set as described earlier.) You can put more than one assignment on a command line. The shell performs the assignments from right to left. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, **name** and **string** can contain multibyte characters; the = (equal sign) must be in ASCII.

If you surround **string** with quotation marks, either double or single (" " ' '), the shell does not treat spaces, tabs, semicolons, and new-line characters within it as word delimiters but imbeds them literally in the string.

If you surround **string** with double quotation marks (" "), the shell still recognizes variable names in the string and performs **parameter substitution**; that is, it replaces references to positional parameters and other variable names that are prefaced by **$** with their corresponding values, if any. The shell also performs **command substitution** (see "Command Substitution" in topic 1.1.420.5) within strings that are surrounded by double quotation marks.

If you surround **string** with single quotation marks (' '), the shell does no variable or command substitution within the string. The following sequence illustrates this difference:

```
     You: stars=*****
          asterisks1="Add $stars"
          asterisks2='Add $stars'
          echo $asterisks1

  System: Add *****

     You: echo $asterisks2

  System: Add $stars
```

The shell does not reinterpret blanks in assignments after parameter substitution (see "Blank Interpretation" in topic 1.1.420.12). Thus the following assignments result in **$first** and **$second** having the same value:

```
  first='a string with embedded blanks'
  second=$first
```

When you reference a parameter, you can enclose the parameter name (or the digit designating a positional parameter) in **{ }** (braces) to delimit the parameter name from any following string. In particular, if the character immediately following the name is a letter, digit, or underscore and the parameter is not a positional parameter, the braces are required:

```
    You: a='This is a'
         echo "${a}n example"

Display: This is an example

    You: echo "$a test"

Display: This is a test
```

See "Conditional Substitution" in topic 1.1.420.9 for a different use of braces in parameter substitutions.

1.1.420.9 Conditional Substitution

Normally, the shell replaces **$name** with the string value assigned to **name**, if there is one. However, there is a special notation that allows **conditional substitution**, depending on whether the variable is set and/or not null. By definition, a variable is **set** if it has ever been assigned a value. The value of a variable can be the null string, which you can assign to a variable in any one of the following ways:

```
A=
bcd=""
Efg=''
set '' ""
```

The first three of these examples assign the null string to each of the corresponding variable names. The last example sets the first and second positional parameters to the null string and unsets all other positional parameters.

The following is a list of the available expressions you can use to perform conditional substitution. The **parameter** and **string** variables can contain multibyte characters.

${parameter-string}
: If the parameter is set, substitute the value of **parameter** in place of this expression. Otherwise, replace this expression with the value of **string**.

${parameter:-string}
: If the parameter is set and is not null, substitute the value of **parameter** in place of this expression. Otherwise, replace this expression with the value of **string**.

${parameter=string}
: If the parameter is set, substitute the value of **parameter** in place of this expression. Otherwise, set **parameter** to **string** and then substitute the value of the **parameter** in place of this expression. You cannot assign values to positional parameters in this fashion.

${parameter:=string}
: If the parameter is set and is not null, substitute the value of **parameter** in place of this expression. Otherwise, set **parameter** to **string** and then substitute the value of the **parameter** in place of this expression. You cannot assign values to positional parameters in this fashion.

${parameter?string}
: If the parameter is set, substitute the value of **parameter** in place of this expression. Otherwise, display a message of the form:

  parameter: **string**

  and act as if the current simple command failed with a status of -1. If you do not specify **string**, the shell displays the following message:

  parameter: **parameter null or not set**

${parameter:?string}

If the parameter is set and not null, substitute the value of **parameter** in place of this expression. Otherwise, display a message of the form:

parameter: string

and act as if the current simple command failed with a status of -1. If you do not specify **string**, the shell displays the following message:

parameter: **parameter null or not set**

${parameter+string}

If the parameter is set, substitute the value of **string** in place of this expression. Otherwise, substitute the null string.

${parameter:+string}

If the parameter is set and not null, substitute the value of **string** in place of this expression. Otherwise, substitute the null **string**.

In conditional substitution, the shell does not evaluate **string** until it uses it as a substituted string, so that, in the following example, the shell executes the **pwd** command only if **d** is not set or is null:

```
  echo ${d:-`pwd`}
```

1.1.420.10 Parameters Used by the Shell

The shell uses the following parameters. The shell sets some of them and you can set or reset all of them. This set of parameter names must contain only ASCII characters. The values that correspond to the parameters can contain multibyte characters.

CDPATH	The search path for the **cd** (change directory) command (see the **PATH** variable in the following list for an explanation of search paths).
HOME	The name of your **login directory**, the directory that becomes the current directory upon completion of a login. The **login** program initializes this variable. The **cd** command uses the value of **$HOME** as its default value. If you use this variable in your shell procedures rather than using explicit full path name, your procedures run even if your login directory is changed or if another user runs them.
LANG	If used to initialize the shell's international environment, specifies a valid locale name. **LANG** identifies general requirements for language, cultural data, and character code set.
LC_COLLATE	Identifies requirements for text collation.
LC_CTYPE	Identifies requirements for character classification and case conversion.
LC_MESSAGE	Determines the language in which the system error messages appear.
LC_MONETARY	Specifies requirements for formatting monetary values.
LC_NUMERIC	Identifies requirements for number formats.
LC_TIME	Specifies the locale's requirements for data and time formats.
MAIL	The path name of the file used by the mail system to detect the arrival of new mail. If **MAIL** is set, the shell periodically checks the modification time of this file and displays the value of **$MAILMSG** if this time changes and the length of the file is greater than zero. You should set **MAIL** in your **.profile** file. The value normally assigned to it by users of the **mail** command is **$HOME/.newmail**.
MAILCHECK	The number of seconds that the shell lets elapse before checking again for the arrival of mail in the files specified by the **MAILPATH** or **MAIL** parameters. The default value is 600 seconds (10 minutes). If you set **MAILCHECK** to 0, the shell checks before each prompt.
MAILPATH	A colon-separated list of file names (see **PATH**). If you set this parameter, the shell informs you of the

arrival of mail in any of the files specified in the list. You can follow each file name by a **%** (percent sign) and a message to be displayed when mail arrives. Otherwise, the shell uses the value of **MAILMSG** or by default "**you have mail**".

Note: When **MAILPATH** is set, these files are checked instead of the file set by **MAIL**. To check the files set by **MAILPATH** and the file set by **MAIL**, specify the **MAIL** file in your list of **MAILPATH** files.

MAILMSG

The mail notification message. If you explicitly set **MAILMSG** to a null string (**MAIL=""**), no message is displayed.

PATH

An ordered list of directory path names separated by colons. The shell searches these directories in the specified order when it looks for commands. A null string anywhere in the list represents the current directory.

PATH is normally initialized in the **/etc/profile** file, usually to **/bin:/usr/bin:/usr/ucb:**. You can reset this variable to suit your own needs. Thus if you wish to search your current directory first, rather than last, you would enter:

```
PATH=:/bin:/usr/bin:/usr/ucb
```

where, by definition, a null string is assumed in front of the leading colon. If you have a personal directory of commands (say, **$HOME/bin**) that you want searched before the standard system directories, set your **PATH** as follows:

```
PATH=$HOME/bin:/bin:/usr/bin:/etc::
```

The best place to set your **PATH** to something other than the default value is in your **.profile** file (see "The .profile File" in topic 1.1.420.4). You cannot reset **PATH** if you are executing commands under the restricted shell.

PS1

The string to be used as the primary system prompt. An interactive shell displays this prompt string when it expects input. The default value of PS1 is "**$**" (a **$** followed by a space).

PS2

The value of the secondary prompt string. If the shell expects more input when it encounters a new-line character in its input, it prompts with the value of PS2. The default value of PS2 is "> " (a > followed by a blank space).

IFS

The characters that are **internal field separators** (the characters that the shell uses during blank interpretation, see "Blank Interpretation" in topic 1.1.420.12). The shell initially sets **IFS** to include the space, tab, and new-line characters.

SHACCT	The name of a file that you own. If this parameter is set, the shell writes an accounting record in the file for each shell script executed. You can use accounting programs such as **acctcom** and **acctcms** to analyze the data collected.
SHELL	A path name whose simple part (the part after the last /) contains an "r" if you want the shell to become restricted when invoked. This should be set and exported by the **$HOME/.profile** file of each restricted login.
TIMEOUT	A number of minutes. After the shell displays its prompt, you have **TIMEOUT** minutes to enter a command. If you fail to do so, the shell exits; in the login shell, such an exit is a logout. Setting TIMEOUT to 0 inhibits automatic logout.

1.1.420.11 Predefined Special Parameters

Several variables have special meanings; the following are set **only** by the shell:

$#	The number of positional parameters passed to the shell or most recently called function, not counting the name of the shell procedure or function itself. **$#** thus yields the number of the highest-numbered positional parameter that is set. One of the primary uses of this variable is to check for the presence of the required number of arguments.
$?	The exit value of the last command executed. Its value is a decimal string. Most AIX commands return 0 to indicate successful completion. The shell itself returns the current value of **$?** as its exit value.
$$	The process number of the current process. Because process numbers are unique among all existing processes, this string of up to seven digits is often used to generate unique names for temporary files. The following example illustrates the recommended practice of creating temporary files in a directory used only for that purpose:

```
temp=$HOME/temp/$$
ls >$temp
   .
   .
   .
rm $temp
```

$*	All positional parameters to this shell starting with $1.
$@	Same as $* except that $@ is treated specially inside double quotation marks (see "Quoting Mechanisms" in topic 1.1.420.14).
$!	The process number of the last process run in the background (using the **&** terminator). Again, this is a string of up to seven digits.
$-	A string consisting of the names of the execution flags (page 1.1.420.23) currently set in the shell.

1.1.420.12 Blank Interpretation

After the shell performs parameter and command substitution, it scans the results for internal field separators (those defined in the shell variable **IFS**, see page 1.1.420.10). It splits the line into distinct words at each place it finds one of these characters. It retains explicit null arguments ("" '') and discards implicit null arguments (those resulting from parameters that have no values).

1.1.420.13 File-name Substitution

Command parameters are very often file names. You can automatically produce a list of file names as parameters on a command line by specifying a pattern that the shell matches against the file names in a directory.

Most characters in such a pattern match themselves, but you can also use some special pattern-matching characters in your pattern. These special characters are:

*	Matches any string, including the null string.
?	Matches any one character.
[...]	Matches any one of the characters enclosed in square brackets.
[!...]	Matches any character **other than** one of the characters that follow the exclamation mark within square brackets.

Inside square brackets, a pair of characters separated by a - (minus) specifies a set of all characters lexically within the inclusive range of that pair, according to the current collating sequence (see "ctab" in topic 1.1.103). The **LANG** or **LC_COLLATE** environment variable controls the collating sequence.

The current collating sequence may group characters into **equivalence classes** for the purpose of defining the end points of a range of characters. For example, if the collating sequence defines the lexical order to be **AaBbCc...** and groups uppercase and lowercase characters into equivalence classes, then all the following have the same effect: **[a-c]**, **[A-C]**, **[a-C]**, and **[A-c]**.

Pattern matching has some restrictions. If the first character of a file name is a . (dot), it can be matched only by a pattern that literally begins with a dot. For example, ***** matches the file names **myfile** and **yourfile** but not the file names **.myfile** and **.yourfile**. To match these files, use a pattern such as the following:

```
  .*file
```

The character @ at the end of the pattern or immediately preceding a \ is ignored during the matching. However, the @ is appended to the corresponding component of the matched file names to allow hidden directories to be referenced directly (see Overview of the Technical Reference, Hidden Directories section).

When the **onsite** control command is used, the <LOCAL> alias name for the specified site is used for file name generation. Otherwise, the invoking shell's current <LOCAL> alias name is used (see the **getlocal** system call in the *AIX Operating System Technical Reference*).

If a pattern does not match any file names, the pattern itself is returned as the result of the attempted match.

File and directory names should not contain the characters *****, **?**, [, or] because this can cause confusion.

1.1.420.14 Quoting Mechanisms

The following characters have special meaning to the shell and cause termination of a word unless quoted.

;	semicolon	<	less than
&	ampersand	>	greater than
( )	parenthesis		newline
|	vertical bar		space
			tab

Sometimes you want to conceal this special meaning. Single (' ') and double (" ") quotation marks surrounding a string or a backslash (\) before a single character provide this function in somewhat different ways.

Within single quotation marks, all characters (except the single quotation character itself), are taken literally, with any special meaning removed. Thus:

```
  stuff='echo $? $*; ls * | wc'
```

results only in the literal string **echo $? $*; ls * | wc** being assigned to the variable **stuff**; the **echo**, **ls**, and **wc** commands are not executed, nor are the variables **$?** and **$*** and the special character ***** expanded by the shell.

Within double quotation marks, the special meaning of certain characters (the **$**, `, \, and ") does persist, while all other characters are taken literally. Thus, within double quotation marks, command and variable substitution takes place. In addition, the quotation marks do not affect the commands within a command substitution that is part of the quoted string, so characters there retain their special meanings. Even if the value of a substituted parameter contains blanks, the string enclosed in double quotation marks is treated as a single word. An exception is that while "$*" is equivalent to "$1 $2 $3", etc., "$@" is equivalent to "$1", "$2", etc.

Consider the following sequence:

```
     You: ls *

  Display: file1 file2 file3

     You: message="This directory contains `ls * ` "
          echo $message

  Display: This directory contains file1 file2 file3
```

This shows that the ***** special character inside the command substitution was expanded.

To hide the special meaning of **$**, `, and " within double quotation marks, precede these characters with a \ (backslash). Outside of double quotation marks, preceding a character with \ is equivalent to placing it within single quotation marks. Hence, a \ immediately preceding the new-line character (that is, a \ at the end of the line) hides the new-line character and allows you to continue the command line on the next physical line.

1.1.420.15 A Command's Environment

The *environment* is a list of name-value pairs inherited from the parent process. On invocation, the shell scans the environment and creates a parameter for each name found, giving it the corresponding value.

The shell passes to its child processes the variables that have been named as arguments to the built-in **export** command. **export** places the named variables in the environments of both the shell and all its future child processes. All parameters picked up from the shell's environment at start-up are included in the export list.

The environment for any simple command may be augmented by prefixing it with one or more assignments to parameters. If the **-k** flag is set, all such assignments are placed in the environment, even if they occur after the command name.

For example, given the following simple procedure that echoes the values of two variables (saved in a command file named **key_command**):

```
#          key_command
echo $a $b
```

the following command lines produce the output shown:

```
   You: a=key1 b=key2 key_command

Display: key1 key2

   You: a=tom b=john key_command
Display: tom john
```

Parameters passed through the environment are not included in the parameter count stored in **$#**.

A procedure can access the values of any variables in its environment; however, if it changes any of the parameters associated with these environment variables, these changes are not reflected in the shell environment. They are local to the procedure in question. To place these changes in the environment that the procedure passes to its child processes, you must **export** these values within that procedure (but see also the **-a** on page 1.1.420.23).

A parameter can be removed from the environment with the built-in **unset** command.

To obtain a list of variables that have been made exportable from the current shell, enter:

```
export
```

To get a list of name-value pairs in the current environment, enter:

```
env
```

1.1.420.16 Redirection of Input and Output

In general, most commands do not know or care whether their input or output is associated with the keyboard, the display screen, or a file. Thus a command can be used conveniently either at the keyboard or in a pipeline.

1.1.420.17 Standard Input and Standard Output

When a command begins running, it usually expects that three files are already open: **standard input**, **standard output**, and **diagnostic output** (sometimes called **error output** or **standard error output**). A number called a **file descriptor** is associated with each of these files as follows:

File descriptor 0	Standard input
File descriptor 1	Standard output
File descriptor 2	Diagnostic (error) output

A child process normally inherits these files from its parent; all three files are initially assigned to the work station (0 to the keyboard, 1 and 2 to the display). The shell permits them to be redirected elsewhere before control is passed to a command. Any argument to the shell in the form <**file** or >**file** opens the specified file as the standard input or output, respectively. In the case of output, this process destroys the previous contents of **file**, if it already exists. An argument in the form >>**file** directs the standard output to the end of **file**, thus allowing you to append data. If **file** does not exist, the shell creates it.

Such redirection arguments are subject only to variable and command substitution; neither blank interpretation nor pattern matching of file names occurs after these substitutions. Thus:

```
  echo 'this is a test' > *.ggg
```

produces a one-line file named ***.ggg** (a disastrous name for a file), and:

```
  cat < ?
```

produces an error message, unless you have a file named **?** (also a bad choice for a file name). If <> is used in place of >, the standard input is opened with both read and write enabled. The file must already exist as with <.

1.1.420.18 Diagnostic and Other Output

Diagnostic output from UNIX commands is normally directed to the file associated with file descriptor 2. You can redirect this error output to a file by immediately preceding either output redirection arrow (> >>) with a **2** (the number of the file descriptor). For example, the following line appends error messages from the **cc** command to the file **ERRORS**:

```
  cc testfile.c 2>> ERRORS
```

There must be no blanks between the file descriptor and the redirection symbol; otherwise, the shell interprets the number as a separate argument to the command.

You can also use this method to redirect the input or output associated with any of the first 10 file descriptors (numbered 0 through 9). Any redirection symbol (<, <<, >, >>, <>, or >&digit) may be preceded by a single digit to override the default use of standard input or output. For instance, if a command (**cmd**) writes to file descriptor 9 (although this is not a recommended programming habit), you can capture that output in a file **savedata** as follows:

```
  cmd 9> savedata
```

If a command writes to more than one output, you can independently redirect each one. Suppose that a command directs its standard output to file descriptor 1, directs its error output to file descriptor 2, and builds a data file on file descriptor 9. The following command line redirects each of these outputs to a different file:

```
  cmd > standard 2> error 9> data
```

1.1.420.19 Inline Input Documents

Upon seeing a command line of the form:

cmd << eofstring

where **eofstring** is any string that does not contain any pattern-matching characters, the shell takes the subsequent lines as the standard input of **cmd** until it reads a line consisting of only **eofstring**. The lines between the first **eofstring** and the second are frequently referred to as a **here document**. If a - (minus) immediately follows the **<<**, the shell strips leading tab characters from each line of the input document before it passes the line to the command.

The shell creates a temporary file containing the input document and performs variable and command substitution on its contents before passing it to the command. It performs pattern matching on file names that are a part of command lines in command substitutions. If you want to prohibit all substitutions, quote any character of **eofstring**:

cmd << \eofstring

The here document is especially useful for a small amount of input data that is more conveniently placed in the shell procedure rather than kept in a separate file (such as editor "scripts"). For instance, you could enter:

```
  cat <<- xyz
                                                          This message is shown on the
                                                          display with leading tabs remove
  xyz
```

This feature is most useful in shell procedures. Inline input documents cannot appear within grave accents (command substitution).

A << may also be preceded by a digit to use an input file descriptor other than 1.

1.1.420.20 Input and Output Redirection Using File Descriptors

As discussed previously, a command occasionally directs output to some file associated with a file descriptor other than **1** or **2**. The shell also provides a mechanism for creating an output file associated with a particular file descriptor. By entering:

fd1>&fd2

where **fd1** and **fd2** are valid file descriptors, you can direct the output that would normally be associated with file descriptor **fd1** to the file associated with **fd2**. The default value for **fd1** and **fd2** is **1** (standard output). If, at execution time, no file is associated with **fd2**, the redirection is void. The most common use of this mechanism is to **direct** standard error output to the same file as standard output, as follows:

```
  cmd 2>&1
```

If you want to **redirect** both standard output and standard error output to the same file, enter:

```
  cmd > file  2>&1
```

The order here is significant. First, the shell associates file descriptor **1** with **file**; then it associates file descriptor **2** with the file that is currently associated with file descriptor **1**. If you reverse the order of the redirections, standard error output goes to the display and standard output goes to **file** because at the time of the error output redirection, file descriptor **1** is still associated with the display.

You can also use this mechanism to redirect standard input. You could enter:

fda<&fdb

to cause both file descriptors to be associated with the same input file. For commands that run sequentially, the default value of **fda** and **fdb** is **0** (standard input). For commands that run asynchronously (commands terminated by **&**), the default value of **fda** and **fdb** is **/dev/null**. Such input redirection is useful for commands that use two or more input sources.

1.1.420.21 Summary of Redirection Options

The following can appear anywhere in a simple command or can precede or follow a command, but they are not passed to the command:

<file Use **file** as standard input.

>file Use **file** as standard output. Create the file if it does not exist; otherwise truncate it to zero length.

<> Same as >**file**, but open for read-write and default to standard input.

>>file Use **file** as standard output. Create the file if it does not exist; otherwise add the output to the end of the file.

<<[-]eofstr
Read as standard input all lines from **eofstr** up to a line containing only **eofstr** or up to an end-of-file character. If any character in **eofstr** is quoted, the shell does not expand or interpret any characters in the input lines; otherwise, it performs variable and command substitution and ignores a quoted new-line character (\new-line). Use a \ to quote characters within **eofstr** or within the input lines.

If you add a - (minus) to << then all leading tabs are stripped from **eofstr** and from the input lines.

<&digit
Associate standard input with file descriptor **digit**. (generalizes to **fd1<&fd2** as does <**&digit**).

>&digit
Associate standard output with file descriptor **digit**.

<&- Close standard input.

>&- Close standard output.

Any of the above can be preceded by a digit to specify a file descriptor to use instead of standard input or output. The restricted shell does not allow the redirection of output.

In each of the above, any open of a file uses the shell's current <LOCAL> alias, even when the **on** control command is used. If the file word contains backquote command substitution, this expansion is performed on the local site even if the **on** command is used.

1.1.420.22 Control Commands

The following are *reserved words* which are only recognized as special as the first word of a command and when not quoted:

if then else elif fi case esac for while until do done onsite { }

The shell provides several flow control commands that are useful in creating shell procedures:

for name [in word...]

do list

done For each **word**, sets **name** to **word** and executes the commands in **list**. If you omit **in word . . .**, the **for** command executes **list** for each positional parameter that is set. Execution ends when there are no more words in the word list.

case word in

pattern [|pattern]...**) list;;**

[.

.

.
.

pattern [|pattern]...**) list;;]**

esac Executes the commands in the **list** associated with the first **pattern** that matches **word**. Uses the same character-matching notation in patterns that you use for file-name substitution (see "File-name Substitution" in topic 1.1.420.13), except that you do not need to match explicitly a slash, a leading dot, or a dot immediately following a slash. Also, reserved words cannot be used unless quoted.

if list

then list

[elif list]...

[else list]

fi Executes the **list** following the **if** keyword. If it returns a zero exit value, execute the **list** following the first **then**. Otherwise, execute the **list** following each **elif** (if there is an **elif**), and if its exit value is zero, execute the next **then**. Failing that, execute the **list** following the **else**. If no **else**

list or **then list** is executed, the **if** command returns a zero exit value.

while list

do list

done Executes the **list** following the **while**. If the exit value of the list is zero, executes the **list** following **do**. Continue looping through the **list**s until the exit value of the **while list** is nonzero. If no commands in the **do list** are executed, the **while** command returns a zero exit value.

until list

do list

done Executes the **list** following the **until**. If the exit value of the list is nonzero, executes the **list** following **do**. Continues looping through the **list**s until the exit value of the **until list** is zero. If no commands in the **do list** are executed, the **until** command returns a zero exit value.

onsite [-v] sitename|sitenumber|sitetype command
The command is run on the specified site, or on a site of the specified **sitetype**. The command can be either a simple command or a control command as described on page 1.1.420.22. If a **sitename** or **sitenumber** are given, the command is run on the specified site if it is accessible in the current partition. The command is run using the <LOCAL> alias for the specified site (see **getlocal** in the *AIX Technical Reference Vol II*). The site may be specified either by name, number, or cpu type. Command substitution may be used on the site argument to allow a program to chose the best (for example, least loaded) site on which to execute. The <LOCAL> alias for the specified site is used in file name generation; but when opening files specified for input/output redirection, the invoking shell's current <LOCAL> alias and current site number are used for filename expansion. Because the value of the <LOCAL> alias may effect the directories in the PATH, the internal hash table is not used during execution of this command (as if the hash **-r** command had been used). If the **-v** option is given, the shell tells the user which site is being used in the command.

(list) Executes the commands in **list** in a subshell.

{ list; } Executes the commands in **list** in the current shell process (does not spawn a subshell).

name () { list; }
Defines a function that is referenced by **name**. The body of the function is the **list** of commands between the braces.

1.1.420.23 Built-in Commands

: Does nothing. This null command returns a zero exit value.

. file Reads and executes commands from **file** and returns. Does not spawn a subshell. The search path specified by **PATH** is used to find the directory containing **file**.

break [n] Exits from the enclosing **for**, **while**, or **until** loop, if any. If **n** is specified, breaks **n** levels.

continue [n] Resumes the next iteration of the enclosing **for**, **while**, or **until** loop. If **n** is specified, resumes at the **n**th enclosing loop.

cd [dir] Changes the current directory to **dir**. The value of the shell variable **HOME** is the default **dir**. The shell variable **CDPATH** defines the search path for the directory containing **dir**. Alternative directory names appear in a colon-separated list. A null path name specifies the current directory (which is the default path). This null path name can appear immediately after the equal sign in the assignment or between the colon delimiters anywhere else in the path list. If **dir** begins with a slash, the shell does not use the search path. Otherwise, the shell searches each directory in the path. **cd** cannot be executed by the restricted shell.

echo [**arg**...]
Writes arguments to standard output. See "echo" in topic 1.1.146 for a discussion of its usage and parameters.

eval [**arg**...]
Reads arguments as input to the shell and executes the resulting command(s).

exec [**arg**...]
Executes the command specified by argument in place of this shell without creating a new process. Input and output arguments can appear and, if no other arguments appear, cause the shell input or output to be modified (not a good idea with your login shell).

exit [**n**] Causes a shell to exit with the exit value specified by **n**. If you omit **n**, the exit value is that of the last command executed (an end of file also causes a shell to exit).

export [**name**...]
Marks the specified **name**s for automatic export to the environments of subsequently executed commands. If you do not specify a **name**s, **export** displays a list of all names that are exported in this shell. You **cannot** export function names.

hash [**-r**] [**name**...]
For each **name**, finds and remembers the location in the search path of the command specified by **name**. The **-r** flag causes the shell to forget all locations. If you do not specify the flag or any **name**s, the shell displays information about the remembered commands. In this information, **hits** is the number of times a command has been run by the shell process. **Cost**

is a measure of the work required to locate a command in the search path. There are certain situations that require that the stored location of a command be recalculated (for example, the location of a relative path name when the current directory changes). Commands for which that might be done are indicated by an asterisk next to the **hits** information. **Cost** is incremented when the recalculation is done.

migrate [-site] [pid ...]
A request is made for the specified processes to be migrated to the specified site. Pid is the process ID of the process to be migrated. If pid is negative, it is the process group ID of the group to be migrated (the site must be explicitly specified if the first pid argument is negative). 'Site' may be specified by either name, number or machine type (for example i370, i386). In any case, a dash must precede the site name, number or type. The migrate does not begin until the process(es) have a chance to run. In particular, if they are stopped, they do not begin migrating until they are continued. If no site is specified, the default is to "pull" the process to the shell's site. If no pids are specified, the shell migrates itself (and also calls setlocal(S)). The command "migrate -site $$" will use migrate the shell will call setlocal. Whenever setlocal is called, the hash **-r** command is automatically executed in case any files in PATH depend on the <LOCAL> alias.

newgrp [**arg**...]
Executes the **newgrp** command in the current shell process. See "newgrp" in topic 1.1.290 for a discussion of command options.

pwd Displays the current directory. See "pwd" in topic 1.1.344 for a discussion of command options.

read [**name**...]
Reads one line from standard input. Assigns the first word in the line to the first **name**, the second word to the second **name**, and so on, with leftover words assigned to the last **name**. This command returns a 0 unless it encounters an end of file.

readonly [**name**...]
Marks the specified **name**s **readonly**. The values of these **name**s cannot be reset. If you do not specify any **name**s, **readonly** displays a list of all **readonly** names.

return [**n**] Cause a function to exit with a return value of **n**. If you do not specify **n**, the function returns the status of the last command executed in that function. This command is valid only when executed within a shell function.

set [**flag**...[**arg**]...]

-a Marks for export all variables that are modified or changed.

-e Exits immediately if any invoked command (other than a built-in command) exits with a nonzero exit value. Without this flag, the shell does not exit if any invoked command returns a nonzero exit status.

Note: The shell always exits if any built-in command returns a nonzero exit status, even if the **-e** flag is set.

-f Disables file-name substitution.

-h Locates and remembers the commands called within functions as the functions are defined (normally these commands are located when the function is executed. See the **hash** command on page 1.1.420.23).

-k Places all keyword parameters in the environment for a command, not just those that precede the command name.

-n Reads commands but does not execute them.

-t Exits after reading and executing one command.

-u Treats an unset variable as an error when performing variable substitution.

-v Displays shell input lines as they are read.

-x Displays commands and their arguments as they are executed.

-- Does not change any of the flags. This is useful in setting **$1** to a string beginning with a - (minus).

Using a + (plus) rather than a - (minus) unsets flags. You can also specify these flags on the shell command line. The special variable **$-** contains the current set of flags.

Any arguments to **set** are positional parameters and are assigned, in order, to **$1**, **$2**, and so on. If you do not specify flags or parameters, **set** displays the values of all parameters.

setxvers [**string**]
If **string** is provided, it sets the experimental version prefix to **string**. If **string** is not provided, it removes any previous experimental version string. The prefix is used when searching hidden directories for executable files. A hidden directory component that has a name beginning with the prefix string is selected for execution in preference to a component that has a name that doesn't begin with the experimental version string. The experimental version prefix is inherited by child processes. See *AIX Operating System Technical Reference* for more information on hidden directories and searching for executable files.

setspath [LOCAL | site | cpu] ...
Sets the site path. Sites may be specified by name or by number. CPU types may be specified by the same names which are used in hidden directories. If an argument is specified

as **LOCAL**, a **nullsite** site path entry is created (see **setspath** in the *AIX Operating System Technical Reference*). The arguments to this command are file name and command substituted.

Use this command with caution. A poorly formed site path can make it difficult to execute any commands.

shift [**n**] Shifts command line arguments to the left; that is, reassign the value of the positional parameters by discarding the current of value of **$1** and assigning the value of **$2** to **$1**, of **$3** to **$2**, and so on. If there are more than 9 command line arguments, the tenth is assigned to **$9** and any that remain are still unassigned (until after another **shift**). If there are 9 or fewer arguments, a **shift** unsets the highest-numbered positional parameter.

$0 is never shifted. The command **shift n** is a shorthand notation for **n** consecutive shifts. The default value of **n** is 1.

test expr ¦ **[expr]**
Evaluates conditional expressions. See "test" in topic 1.1.469 for a discussion of command options.

times Displays the accumulated user and system times for processes run from the shell.

trap [**arg**] [**n...**]
Runs the command specified by **arg** when the shell receives signal(s) **n**, where **n** is any signal other than 11. (The shell scans **parm** once when the trap is set and once when the trap is taken.) **trap** commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective.

If you do not specify an **arg**, all trap(s) **n** are reset to their current values. If **arg** is the null string, this signal is ignored by the shell and by the commands it invokes. If **n** is 0, **arg** is executed on exit from the shell. If you specify no **arg** and no **n**, **trap** displays a list of commands associated with each signal number.

type [name...]
For each **name**, indicates how the shell would interpret it as a command name.

ulimit [-b [m]] [-f] [n] [-s [m]]
Sets or queries size limits. The **-b** flag sets the break value to **m**. This limits the size of data segment to **m** pages. If you specify **-b** with no **m**, **ulimit** displays the current break value. The **-f** flag imposes a size limit of **n** blocks on files written by the child processes (files of any size can be read). Without **n**, **ulimit** displays the current limit (this is the default action of **ulimit**).

Notes:

1. Since the shell rounds **n down** to the nearest cluster size, it is best to make it a multiple of 8 (for example,

specifying values of 1 thru **n** all result in a size limit of 0).

2. Any user can decrease this limit, but only a user operating with superuser authority can increase the limit.

The **-s** flag imposes a size limit of **m** pages on the stack. If you specify **-s** with no **m**, **ulimit** displays the current stack size limit.

umask [**nnn**] Sets the user file-creation mask to **nnn** (see the **umask** system call). If you omit **nnn**, **umask** displays the current value of the mask.

unset [**name**...]
For each **name**, removes the corresponding variable or function. The variables **PATH**, **PS1**, **PS2**, **MAILCHECK** and **IFS** cannot be unset.

wait [**n**] Waits for the child process whose process number is **n** to end and reports its termination status. If you do not specify **n**, the shell waits for all currently active child processes and the return value is 0.

1.1.420.24 Running the Shell

The **sh** command can be run either as a **login shell** or as a command. Only the **login** command can call **sh** as a login shell. It does this by using a special form of the **sh** command name: **-sh**. When called with an initial - (minus), the shell first reads and runs commands found in the system **profile** file and your **$HOME/.profile**, if one exists. It then accepts commands as described in the following discussion of flags. Once logged in and working under a login shell, you can call **sh** with the command name **sh**. This command runs a subshell, a second shell running as a child of the login shell.

1.1.420.25 Restricted Shell

The restricted shell, **Rsh**, is used to set up login names and environments whose capabilities are more controlled than those of the standard shell. The actions of **Rsh** are identical to those of **sh**, except that the following are not allowed:

Changing directory (see "cd" in topic 1.1.53)
Setting the value of **$PATH**
Specifying path or command names containing **/**
Redirecting output (>, >> and <>)

The restrictions above are enforced after **.profile** is interpreted, meaning that the restrictions can override settings originally set in **.profile**.

When a command to be run is found to be a shell procedure, **Rsh** starts **sh** to run it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and run permissions in the same directory.

The net effect of these rules is that the writer of the **.profile** has complete control over user actions, by performing setup actions and leaving the user in an appropriate directory (probably not the login directory).

When called with the name **-Rsh**, **Rsh** reads the user's **.profile** (from $HOME**/.profile**). It acts as the standard **sh** while doing this, except that an interrupt causes an immediate exit instead of a return to command level.

Flags

The following flags are interpreted by the shell only when you call it. Unless you specify either the **-c** or **-s** flag, the shell assumes that the next parameter is a command file (shell procedure). It passes anything else on the command line to that command file (see "Positional Parameters" in topic 1.1.420.7).

-c cmdstring
Runs commands read from **cmdstring**. The shell does not read additional commands from standard input when you specify this flag.

-i Makes the shell interactive, even if input and output are not from a work station. In this case the shell ignores the **TERMINATE** signal (so that **kill 0** does not stop an interactive shell) and traps an **INTERRUPT** (so that you can interrupt **wait**). In all cases, the shell ignores the **QUIT** signal. (See the **sigaction** system call in *AIX Operating System Technical Reference* and "kill" in topic 1.1.221 for more information about signals.)

-r Creates a restricted shell (the same as running **Rsh**).

-s Reads commands from standard input. Any remaining parameters specified are passed as positional parameters to the new shell. Shell output is written to standard error, except for the output of built-in commands (see "Built-in Commands" in topic 1.1.420.23).

The remaining flags and parameters are described in the built-in **set** command on page 1.1.420.23.

Files

/etc/environment
/etc/profile
$HOME/.profile
/tmp/sh*
/dev/null
/etc/site

Related Information

See the following commands: "cd" in topic 1.1.53, "echo" in topic 1.1.146, "env, printenv" in topic 1.1.151, "login" in topic 1.1.241, "newgrp" in topic 1.1.290, "pwd" in topic 1.1.344, "test" in topic 1.1.469 and "umask" in topic 1.1.490.

See the **dup**, **exec**, **fork**, **statx**, **pipe**, **sigaction**, **ulimit**, and **umask** system calls and the **a.out**, **.profile**, and **environ** files in *AIX Operating System Technical Reference*.

See "Using the Shell with Processes" and "Advanced Shell Features" in *Using the AIX Operating System*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*. See also "Programming for an MBCS Environment" in *AIX Operating System Programming Tools and Interfaces*.

1.1.421 shlibrpt

Purpose
Generates a report file that aids in constructing a shared library.

Syntax

```
                                         +---------------+
shlibrpt --- list of object files ---¦               +---¦
                                         +- target file -+
```

Description
Given a list of object files ending in **.o** files that are to comprise a shared library, this utility generates:

a shared lib specification file template
an **import.c** file template;
an **import.h** file template; and
a complete cross reference, listing the imported and global variable and the files in which each variable is referenced or defined.

All output is directed to stdout by default, but can be placed into a single output file by using I/O redirection.

After the output file is created you will have to edit it to produce the three files required to make a shared library:

the specification fil
import.h and
import.c.

In addition, the template files requires you to fill in certain items. (The general rule is to fill in items enclosed within < > braces.) Since there is no way to determine the type of a variable this must be filled in by you. The level of indirection can not be determined, so you must fill in the [*] selection - the rule here is that the use of imported variables/routines adds one extra level of indirection. This extra level is taken care of in the template, you must fill in the original level of indirection.

Limitations

All **.o** files must have a symbol table.
All **.o** files must be produced by the compiler or if assembled, the assembly language file must include the **.file** assembly directive; otherwise, the cross reference may fail.

Examples
In the directory containing the **.o** files which are to be placed into a shared library, issue the following command:

```
  shlibrpt 'ls *.o' > REPORT
```

Related Information

See the following command: "shlib2" in topic 1.1.422.

See the "Making a shared library" tutorial in *AIX Operating System Programming Tools and Interfaces*.

1.1.422 shlib2

Purpose
Creates a shared library.

Syntax

```
          +------------+
shlib2 ---¦ +--------+ +--- -sspecfile --- -ttarget ---¦
          +-¦ -hhost +-+
            ¦ -n     ¦¦
           ¦¦ -q     ¦¦
           ¦+--------+¦
           +----------+
```

Description
The **shlib2** command creates a shared library, similar in function to an unshared library except that multiple programs linking with a shared library will use the same library code during execution.

The **shlib2** command creates both the host and target libraries. The host shared library is an archive, used to link-edit user programs with the shared library. It is used as an unshared library is used. See "ar" in topic 1.1.23 and "cc" in topic 1.1.52 for information on creating an archive library and linking a file.

The target shared library is an executable attached to the user's process. It must be fully resolved as it contains the code for all of the library routines.

To use **shlib2**, it is necessary to specify options on the command line and describe the contents of the shared library in a library specification file.

The **shlib2** command invokes **ar**, the archiver, **as**, the assembler and **ld**, the loader.

The shared library specification file is created to specify the contents of both the host and target shared libraries. To create this file, the following are needed:

#address section address
Defines the start address of the given target file section.

#target path name
Defines the absolute path name of the target shared library.

#branch Defines the start of the branch table specifications. The syntax is as follows:

```
symbol position
```

Where **symbol** is the name of the branch table entry symbol and **position** is the position of the entry. Position may be specified as an integer, or a range of integers. Only external functions can be given branch table entries.

#objects Defines the names of the object files making up the target shared library. This list consists of the file names followed by blanks. The order for the host shared library is given via running the list through the **lorder** and **tsort** commands.

#init object Defines that the object file requires initialization code. The following is the format of initialization coding:

```
pointer symbol
```

Where **pointer** points to the "import" symbol, and must be defined in **object**. The following is the format of the initialization code generated:

```
pointer = &symbol
```

#ident string Defines a string to be included in the comment section of the target shared library.

Defines a comment; all information following this string is ignored.

Flags

-s specfile	Contains the information needed to build a shared library.
-t target	Defines the name of the **target** shared library.
-h host	Defines the name of the **host** shared library. The host shared library is not created if this flag is not specified.
-n	This prevents the generation of a new target shared library. The **-t** flag is still required to build the host shared library even though the target shared library is not generated.
-q	Suppresses warning messages.

Files

TEMPDIR/*	Temporary files. **TEMPDIR** is usually **/usr/tmp**. **TEMPDIR** can be redefined by setting its environment variable.

Related Information

See the discussion of shared libraries in *AIX Operating System Programming Tools and Interfaces*.

1.1.423 show

Purpose
Shows messages.

Syntax

```
        +-----------+   +----- cur ----------------------------------------+
show ---¦           +---¦ +-------- all -------------------------------+ +--
        +- +folder -+   +-¦ +----------- -draft ----------------------+ +-+
                          +-¦    +------------ sequence ------------+   +-+
                            +---¦  one of                           +---+
                                ¦ +-------+   +------------------+ ¦ ¦
                              ¦ +-¦ num   +---¦      one of      +-+ ¦
                              ¦   ¦ first ¦   ¦¦+--------------+ ¦   ¦
                              ¦   ¦ prev  ¦   +-¦ :num   -prev +-+   ¦
                              ¦   ¦ cur   ¦     ¦ :+num  -cur  ¦     ¦
                              ¦   ¦ .     ¦     ¦ :-num  -.    ¦     ¦
                              ¦   ¦ next  ¦     ¦ -num   -next ¦     ¦
                              ¦   ¦ last  ¦     ¦ -first -last ¦     ¦
                              ¦   +-------+     +--------------+     ¦
                              +--------------------------------------+

    +--- -header ---+   +-------------------------+
 ---¦    one of     +---¦         one of          +---¦
    ¦ +-----------+ ¦   ¦ +---------------------+ ¦
    +-¦ -header   +-+   +-¦ -showproc cmdstring +-+
      ¦ -noheader ¦       ¦ -noshowproc         ¦
      +-----------+       +---------------------+

show --- -help ---¦


-----------------
¦ Do not put a blank between these items.
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **show** command is used to display a list of messages. **show** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **show** command sends a listing of each of the specified messages to standard output. If standard output is not a display, **show** lists each message with a one-line header and two separation lines.

The **show** command invokes a program to perform the listing. The system default is **/bin/pg**. You can define your own default with the **showproc:** entry in **$HOME/.mh_profile**. If you set **showproc:** entry to **mhl**, **show** calls an internal **mhl** routine instead of the **mhl** command. You can also specify the program to perform a listing in the **cmdstring** of the **-showproc** flag.

The **show** command passes any flags that it does not recognize to the program performing the listing. Thus, you can specify flags for the listing program, as well as the flags described in this command section.

If the **Unseen-Sequence** entry is present in **$HOME/.mh_profile** and is not empty, **show** removes each of the messages shown from each sequence named by the profile entry.

Flags

-draft	Shows the file **user_mh_directory/draft** if it exists.
+*folder msgs*	Specifies the messages that you want to show. *msgs*can be several messages, a range of messages, or a single message. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default message is the current message in the current folder. If several messages are specified, the last message shown becomes the current message.

-header	Displays a one-line description of the message being shown. The description includes the folder name and the message number. If you show more than one message, this flag does not produce message headers. This flag is the default.
-help	Displays help information for the command.
-noheader	Does not display a one-line description of each message being shown.
-noshowproc	Uses **/bin/cat** to perform the listing.
-showproc *cmdstring*	Uses the specified command string to perform the listing.

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.
showproc:	Specifies the program used to show messages.
Unseen-Sequence:	Specifies the sequences used to keep track of your unseen messages.

Files

$HOME/.mh_profile	The MH user profile.
user_mh_directory/draft	The draft file.

Related Information

See the following commands: "mhl" in topic 1.1.263, "next" in topic 1.1.293, "pick" in topic 1.1.316, "prev" in topic 1.1.323, "scan" in topic 1.1.409, "sendmail, mailq, newaliases" in topic 1.1.417.

See the **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX*

Operating System.

1.1.424 showmount

Purpose
Shows all file systems that are mounted remotely in an NFS environment.

Syntax

```
             +--------------------------+
showmount ---¦ +----------------------+ +---¦
             +-¦ -a hostname:directory +-+
               ¦ -d                   ¦
               ¦ -e                   ¦
               +----------------------+
```

Description
The **showmount** command lists the clients that have mounted a file system from an NFS server. This information is maintained by the network daemon **mountd** which runs continuously on the server. The daemon saves this information to the file **/etc/rm tab** for use if the server crashes.

The default value for the server is returned by **hostname**.

Flags

-d Lists directories that have been remotely mounted by clients.

-a Prints all remote mounts in the format hostname:directory.

The **hostname** parameter specifies the name of the client, and **directory** specifies the root of the mounted file system.

-e Prints the list of exported file systems.

1.1.425 shutdown

Purpose
Ends system operation.

Syntax

```
              +-----------+   +-----------------------+
shutdown ---¦ +-------+ +---¦           +-----------+ +---¦
            +-¦ -c -m +-+   +- + time -¦           +-+
              ¦ -F -n ¦               +- message -+
              ¦ -h -r ¦
              ¦ -i -v ¦
              ¦ -k    ¦
              +-------+
```

Description
The **shutdown** command brings the AIX Operating System from multi-user mode to maintenance mode, or halts the operating system completely. You can run **shutdown** only if you are a member of the system group, have superuser authority or are logged in at the console of a PS/2.

Default shutdown halts the operating system. During the default shutdown, users are notified (by a **wall** command) of the impending system shutdown with a message. However, the shutdown is not complete until the user receives a shutdown completion message. Do not attempt to reboot the system or turn off the system before the shutdown completion message is displayed; otherwise, file system damage may result. **Time** is the time at which **shutdown** will bring the system down and may be the word **now** (indicating an immediate shutdown) or specify a future time in one of two formats: +number and hour:min. The first form brings the system down in **number** minutes and the second brings the system down at the time of day indicated (as a 24-hour clock).

At intervals which get closer together as **shutdown** approaches, warning messages are displayed at the terminals of all users on the system.

During the shutdown, the **phold** command prevents any new logins. After the specified number of **minutes** (1 by default), if none of the **-r, -m** or **-k** flags are specified, shutdown invokes /etc/halt. If the **-r** flag is specified, shutdown invokes /etc/reboot. In both cases, all processes are terminated and all file systems are unmounted. If the **-n** flag is not specified, all memory resident disk blocks are flushed to disk.

If the **-r, -m** or **-k** flag was specified, then shutdown will invoke the command **reboot,** bring the system down to maintenance (for example single user) mode, or avoid shutting down the system, respectively. (If it isn't obvious, **-k** is to make people **think** the system is going down!)

The **-i** option invokes an interactive shutdown procedure.

Note: Only users logged onto the machine or logged onto another machine in the same TCF cluster will be notified about the shutdown. Users using this machine as a file server (for example, through AIX/NFS or AIX Access for DOS Users) will not be notified.

If there are no other users on the system and you want to shutdown the system quickly, you may want to use **shutdown -F**. This option bypasses messages to users and brings the system down as quickly as possible.

If **message** is specified, **time** must be specified.

Warning: For **shutdown -m**, for example, if you are bringing the system down to maintenance mode, you must run shutdown from the root directory to ensure that it can cleanly unmount the file systems.

Flags

-c Does not check file system upon rebooting.

-F Does a fast shutdown, bypassing the messages to other users and bringing the system down as quickly as possible. If you do not specify this flag, **shutdown** sends a message to each logged-in user and waits a certain amount of time before bringing the system down, to allow each user to log off cleanly.

-h Halts the operating system completely.

-i Interactive shutdown. Guides the user through the shutdown procedure.

-k Avoid shutting down the system.

-m Bring the system down to maintenance mode.

-n Don't sync disks

-r Do a reboot.

-v Halts the operating system completely (same as **-h**).

Examples

1. To tell the operating system you are about to turn off the machine:

   ```
   shutdown
   ```

   This shuts down the system, waiting 1 minute before stopping the user processes and the **init** process.

2. To give users and the system more time to finish what they are doing:

   ```
   shutdown  -m  +5
   ```

   This brings the system down from multi-user mode to maintenance mode after waiting 5 minutes.

Files

/etc/shutdown.sh

Related Information

See the following commands: "acct/*" in topic 1.1.6, "errstop" in topic 1.1.156, "fastboot, fasthalt" in topic 1.1.162, "halt" in topic 1.1.195, "init, telinit" in topic 1.1.208, "kill" in topic 1.1.221, "killall" in topic 1.1.222, "reboot" in topic 1.1.363, and "pdisable, phold" in topic 1.1.314.

See the **sigaction** system call in *AIX Operating System Technical Reference*.

1.1.426 site, sitechar, sitelocal, sitename, sitenum, ptn

Purpose

Display information about cluster sites.

Syntax

```
one of       +----------------+
site --------¦ +------------+ +---¦
sitechar     +-¦ sitenumber +-+
sitelocal      ¦ sitename   ¦¦
sitename      ¦¦ -a         ¦¦
sitenum       ¦¦ -pcputype  ¦¦
ptn           ¦¦ -t         ¦¦
              ¦¦ -n         ¦¦
              ¦¦ -c         ¦¦
              ¦¦ -l         ¦¦
              ¦¦ -#         ¦¦
              ¦¦ -f         ¦¦
              ¦¦ -m         ¦¦
              ¦+------------+¦
              +--------------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **/etc/site** command is used to report on information about sites within the cluster. This information is obtained from the cluster site data base file, **/etc/site**. The command line options and arguments control which sites are reported and what information about them is displayed.

Options

The command line arguments and options for **site** are of two major types. The first type are those that select the specific sites about which information is printed. If none of these options are specified, the cluster site where **site** is running is selected.

sitename Displays information about the specific cluster site whose name is **sitename**. The **sitename** argument is not prefixed by the - symbol.

sitenumber Displays information about the specific cluster site whose site number is **sitenumber**. The **sitenumber** argument is not prefixed by the - symbol.

-a Displays information about all cluster sites.

-pcputype Displays information on all sites of a particular cpu type. This option may appear more than once on the command line with different cpu type arguments.

-t Displays information of all sites in the cluster that are currently active. The **-t** option overrides all **sitename** and **sitenumber** arguments, as well as the **-a** option. If used with the **-p** option, then information is displayed for all sites in the cluster of the specified cpu types that are currently active.

The second type of arguments select what information from the cluster site data base is displayed. If more than one of these options appear on the command line, the rightmost one is used. If none of these options are specified, the cluster site name is displayed.

-n Display the name of the cluster site of each site selected.

-c Display the first character of the name of each cluster site selected.

-l Display the name of the <LOCAL> file system of each cluster site selected.

-# Display the cluster site number of each cluster site selected.

-f Display all information from the site file data base for each cluster site selected.

-m Display the cpu type of each cluster site selected, in addition to either the cluster site name, the name of the <LOCAL> filesystem of the cluster site, the number of the cluster site, or the first character of the name of the cluster site, as selected by other options. If specified with the **-f** option, the **-m** option has no effect.

Alternate Invocations

The **site** command can be invoked using alternate names to produce different outputs or output formats.

sitechar is equivalent to **site -c**

sitelocal is equivalent to **site -l**

sitenum is equivalent to **site -#**

sitename is equivalent to **site -n**

ptn is equivalent to **site -t**

Examples

1. To display the names of all cluster sites

   site **-a**

   This lists the name of each cluster site, regardless whether it is currently active in the cluster.

2. To display the names of all the cluster sites that are currently active

   site **-t** (or ptn)

3. To list all currently active cluster sites that are AIX PS/2 CPU sites

   site **-t** -pi386

4. To list all information about cluster site number 6

   site **-f** 6

   If there is no cluster site number 6 in the site file, an error message is displayed.

Files

/etc/site

Related Information

See the **site** file format in the *AIX Operating System Technical Reference*.

1.1.427 size

Purpose
Displays the section sizes of common object files.

Syntax

```
       +--------+   +------+   +-- a.out ---+
size ---¦ one of +---¦      +---¦           +---¦
       ¦ +----+ ¦   +- -f -+   +--- file ---+
       +-¦ -o +-+                        ¦
         ¦ -x ¦                  +--------+
         ¦ -V ¦
         ¦ -d ¦
         +----+
```

Description
The **size** command writes to the standard output the number of bytes required by the text, initialized data, and uninitialized data, along with their sum for each **file**. The default **file** is **a.out**. The **size** command must be run on a machine of the same machine type as the object file being examined.

Flags

The output is in decimal notation unless you change the output with the following flags:

-d Writes decimal notation (default).

-f Writes the section name in parenthesis following its size.

-o Writes in octal notation.

-V Prints version number.

-x Writes in hexadecimal notation.

Examples

1. To display the size of **a.out** in decimal:

   ```
   size
   ```

   This displays the size in bytes of the executable file **a.out**. The size of each section of the object file is given, followed by the total. The three sections are program text, data, and bss (uninitialized data). The values are in decimal.

2. To display the size of an object file in octal:

   ```
   size  -o  driver.o
   ```

   This displays the size of the object file **driver.o** in octal.

3. To display the size of several object files in hex:

   ```
   size  -x -f  *.o
   ```

This displays in hexadecimal the sizes and section names of each file in the current directory ending with **.o**.

Related Information

See the following commands: "ar" in topic 1.1.23, "as" in topic 1.1.25, "cc" in topic 1.1.52, "dump" in topic 1.1.142, "ld" in topic 1.1.226, "nm" in topic 1.1.298, and "strip" in topic 1.1.445.

See the discussion of the **a.out** and **ar** files in *AIX Operating System Technical Reference*.

1.1.428 skulker

Purpose
Cleans up file systems by removing unwanted files.

Syntax

/etc/skulker ---¦

Description

Warning: Because this command file runs with superuser authority and its whole purpose is to remove files, it has the potential for unexpected results. Before installing a new **skulker**, test any additions to its file removal criteria by running it manually using the **xargs -p** command. After you have verified that the new **skulker** removes only the files you want removed, you can install it.

The **skulker** command is a shell command file for periodically purging obsolete or unneeded files from file systems. Candidate files include those in **/tmp**, **.bak** files older than a specified age, and files named **a.out**, **core**, or **ed.hup**.

The **skulker** command is normally invoked daily, often as part of an accounting procedure run by **cron** during off-peak periods. Individual sites should modify **skulker** to suit local needs following the patterns shown in the distributed version. Local users should be made aware of the criteria for automatic file removal.

The **find** and **xargs** commands form a powerful combination for use in **skulker**. Most file selection criteria can be expressed conveniently with **find** expressions. The resulting file list, generated with the **-print** flag of the **find** command, can then be segmented and inserted into **rm** commands using **xargs** to reduce the overhead that would result if each file were deleted with a separate command.

Related Information

See the following commands: "cron" in topic 1.1.97, "find" in topic 1.1.165, "rm, delete" in topic 1.1.375, and "xargs" in topic 1.1.544.

1.1.429 sleep

Purpose
Suspends execution for an interval.

Syntax

sleep -- **seconds** --¦

Description
The **sleep** command suspends execution of a process for the interval specified by **seconds**. **seconds** can range from 1 to 65,536 seconds.

Examples

1. To run a command after a certain amount of time has passed:

   ```
   echo  "SYSTEM SHUTDOWN IN 10 MINUTES!" | wall
   (sleep 300; echo "SYSTEM SHUTDOWN IN 5 MINUTES!" | wall) &
   (sleep 540; echo "SYSTEM SHUTDOWN IN 1 MINUTE!" | wall) &
   (sleep 600; shutdown)  &
   ```

   This command sequence warns all users 10 minutes, 5 minutes, and 1 minute before the system is shut down.

2. To run a command at regular intervals:

   ```
   while true
   do
      date
      sleep 60
   done
   ```

   This shell procedure displays the date and time once a minute. To stop it, press INTERRUPT (**Ctrl-C**).

Related Information

See the following commands: "shutdown" in topic 1.1.425 and "wall" in topic 1.1.529.

See the **alarm** system call and the **sleep**, **pause**, and **signal** subroutines in *AIX Operating System Technical Reference*.

1.1.430 slocal

Purpose
Processes incoming mail.

Syntax

```
          +----------------+   +----------+
slocal ---¦     one of     +---¦          +---¦
        ¦ ¦ +------------+ ¦   +- -debug -+ ¦
        ¦ +-¦ -verbose   +-+                ¦
        ¦   ¦ -noverbose ¦                  ¦
        ¦   +------------+                  ¦
        +-------------- -help --------------+
```

Description

The **slocal** command is used to perform a set of actions each time a message is sent to the user. **slocal** is not designed to be run directly by the user; it is designed to be called by the **sendmail** command. The **slocal** command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **sendmail** command invokes the **slocal** command when it encounters the following line in **$HOME/.forward**:

```
  ¦ /usr/lib/mh/slocal
```

For each new incoming message, **slocal** performs the actions specified in the **.mail delivery** file. If **slocal** cannot find the file **$HOME/.maildelivery**, **slocal** uses the default file **/usr/lib/mh/maildelivery**. If the delivery request fails, **slocal** delivers the message to **$HOME/$USER**.

The actions that can be specified in **.mail delivery** are described in *AIX Operating System Technical Reference* under **mhooks**.

Flags

-debug	Provides information for debugging.
-help	Displays help information for the command.
-noverbose	Does not display information as the system executes commands in the maildelivery file. This flag is the default.
-verbose	Displays information as the system executes commands in the maildelivery file.

Files

/usr/lib/mh/mtstailor	The MH tailor file.
/usr/lib/mh/maildelivery	The default *message handling*local delivery instructions file.
$HOME/.maildelivery	The user's local delivery instructions file.
$HOME/.forward	The user's default message filter file.

Related Information

See the following commands: "rcvdist" in topic 1.1.355, "rcvpack" in topic 1.1.356, "rcvstore" in topic 1.1.357, "rcvtty" in topic 1.1.358, "sendmail, mailq, newaliases" in topic 1.1.417.

See the **mh-format**, **mhook**, **mh-mail** and **mh-profile** file formats in *AIX Operating System Technical Reference*.

1.1.431 soelim

Purpose
Eliminates .so's from **nroff** input.

Syntax

```
          +--------+
soelim ---¦        +---¦
          +- file -+
```

Note: This command does not have MBCS support.

Description

The **soelim** command reads the specified files or the standard input and performs the textual inclusion implied by the **nroff** directives of the form

```
  .so somefile
```

when they appear at the beginning of input lines. This is useful since programs such as **tbl** do not normally do this; it allows the placement of individual tables in separate files to be run as a part of a large document.

An argument consisting of a single minus (**-**) is taken to be a file name corresponding to the standard input.

Inclusion can be suppressed by using '~' instead of '.', for example

```
  'so /usr/lib/tmac.s
```

A sample usage of **soelim** would be

```
soelim exum?.n | tbl | nroff -ms | col | print
```

Related Information

See the following commands: "colcrt" in topic 1.1.78, "nroff, troff" in topic 1.1.301, and "tbl" in topic 1.1.463.

1.1.432 sort

Purpose
Sorts files.

Syntax

```
        +----------------------+
sort ---¦ +------------------+ +---
        +-¦ -b -f -o outfile +-+
          ¦ -c -i -r -       ¦¦
         ¦¦ -d -m -t char    ¦¦
         ¦¦ -A -n -T -u      ¦¦
         ¦+------------------+¦
         +--------------------+

    +-------------------------------------------------------------+   +-------
 ---¦ +---- +fskip ----+   +--------------------+   +-----------+ +---¦
    +-¦                +---¦ +---- -fskip ----+ +---¦ +-------+ +-+   +--- fi
      +- +fskip.cskip -    +-¦                +-+   +-¦ -b -i +-+¦
     ¦                       +- -fskip.cskip -+       ¦ -d -n ¦¦        +-----
     ¦                                               ¦¦ -f -r ¦¦ ¦
     ¦                                               ¦+-------+¦ ¦
     ¦                                               +---------+ ¦
     +-----------------------------------------------------------+
```

Description
The **sort** command sorts lines in its input **file**s and writes the result to standard output. It treats all of its input **file**s as one file when it performs the sort. A **-** (minus) in place of a file name specifies standard input. If you do not specify any file names, it sorts standard input.

The default sort key (the part of the line used for sorting) is an entire line. Default ordering is lexicographic by characters in the collating sequence. The collation order is locale-dependent; the behavior of **sort** is modified by setting the environment variables **LANG** or **LC_COLLATE**.

The two numbers, **fskip** and **cskip**, specify the sort key. Both numbers have two parts, as follows:

+fskip.cskip
-fskip.cskip

The **fskip** specifies the number of fields to skip from the beginning of the input line, and **cskip** specifies the number of additional characters to skip to the right beyond that point. For both the starting point (**+fskip.cskip**) and the ending point (**-fskip.cskip**) of a sort key, **fskip** is measured from the beginning of the input line, and **cskip** is measured from the last field skipped. If you omit **.cskip**, **.0** is assumed. If you omit **fskip**, **0** is assumed. If you omit the ending field specifier (**-fskip.cskip**), the end of the line is the end of the sort key.

You can supply more than one sort key by repeating **+fskip.cskip** and **-fskip.cskip**. In cases where you specify more than one sort key, keys specified further to the right on the command line are compared only after all earlier keys are sorted. For example, if the first key is to be sorted in numerical order and the second in dictionary order, all strings that start with the number one are sorted alphabetically before the strings that start with the number two. Lines that are identical in all keys are sorted with all characters significant. You can also specify different flags for different sort keys in multiple sort keys. See the examples for illustration.

A field is one or more characters bounded by the beginning of a line and the current field separator, or one or more characters bounded by a the field separator on either side. The space character is the default field separator.

Notes:

1. Lines longer than 1024 are truncated.
2. The maximum number of fields on a line is 10.

Flags

-A	Sorts on a byte-by-byte basis using ASCII character values.
-b	Ignores leading blanks, spaces, and tabs in sort key comparisons.
-c	Checks that the input is sorted according to the ordering rules specified in the flags. Displays nothing unless the file is not sorted.
-d	Sorts in dictionary order. Only letters, digits and blanks are considered in comparisons.
-f	Merges uppercase and lowercase letters. Case is not considered in the sorting, so that initial-capital words and all-capital words are not grouped together at the beginning of the output.
-i	Sorts only by characters in the ASCII range octal 040-0176 (all printable characters and the space character) in non-numeric comparisons.
-M	Compare as months. The first three non-blank characters of the field are folded to uppercase and compared so that "JAN" < "FEB" < ...< "DEC". Invalid fields compare low to "JAN". The language used for month names are affected by locale.
-m	Merges only; the input is already sorted.
-n	Sorts any initial numeric strings (consisting of optional blanks, optional minus signs, and zero or more digits with optional decimal point) by arithmetic value. The **-n** flag automatically gives you the **-b** flag.
-o outfile	Directs output to **outfile** instead of standard output. **outfile** can be the same as one of the input **files**.
-r	Reverses the order of the specified sort.
-tchar	Sets field separator character to **char**. To specify the tab character as the field separator, you must enclose it in single quotation marks (' ').
-T	Uses current directory instead of default directory for temporary files.

-u Suppresses all but one in each set of equal lines. Ignored characters (such as leading tabs and spaces) and characters outside of sort keys are not considered in this type of comparison.

Examples

1. To perform a simple sort:

   ```
   sort  fruits
   ```

   This displays the contents of **fruits** sorted in ascending lexicographic order. This means that the characters in each column are compared one by one, including spaces, digits, and special characters. For instance, if **fruits** contains the text:

   ```
   banana
   orange
   Persimmon
   apple
   %%banana
   apple
   ORANGE
   ```

   then **sort** displays:

   ```
   %%banana
   ORANGE
   Persimmon
   apple
   apple
   banana
   orange
   ```

   This order follows from the fact that in the ASCII collating sequence, **%** (percent sign) precedes the uppercase letters, which precede the lowercase letters. If the system uses a character set other than ASCII, your results may be different.

2. To sort in dictionary order:

   ```
   sort  -d  fruits
   ```

   This sorts and displays the contents of **fruits**, comparing only letters, digits, and blanks. If **fruits** is the same as in Example 1, **sort** displays:

   ```
   ORANGE
   Persimmon
   apple
   apple
   %%banana
   banana
   orange
   ```

   The **-d** flag tells **sort** to ignore the **%** character because it is not a letter, digit, or blank. This puts **%%banana** next to **banana**.

3. To group lines that contain uppercase and special characters with

similar lowercase lines:

```
sort  -d  -f  fruits
```

This ignores special characters (**-d**) and differences in case (**-f**). Given the **fruits** of Example 1, this displays:

```
apple
apple
%%banana
banana
ORANGE
orange
Persimmon
```

4. To sort as in Example 3 and remove duplicate lines:

```
sort  -d  -f  -u  fruits
```

The **-u** flag tells **sort** to remove duplicate lines, making each line of the file unique. This displays:

```
apple
%%banana
orange
Persimmon
```

Not only was the duplicate **apple** removed, but **banana** and **ORANGE** as well. These were removed because the **-d** told **sort** to treat **%%banana** as if it were **banana**, and the **-f** told it to treat **ORANGE** as **orange**. Thus, **sort** considered **%%banana** to be a duplicate of **banana** and **ORANGE** a duplicate of **orange**.

Note: There is no way to predict which duplicate lines **sort -u** will keep and which it will remove.

5. To sort as in Example 3 and remove duplicates, unless capitalized or punctuated differently:

```
sort  -u  +0  -d  -f       +0  fruits
```

The **+0 -d -f** does the same type of sort done with **-d -f** in Example 3. Then the **+0** performs another comparison to distinguish lines that are not actually identical. This prevents **-u** from removing them.

Given the **fruits** file shown in Example 1, the added **+0** distinguishes **%%banana** from **banana** and **ORANGE** from **orange**. However, the two instances of **apple** are identical, so one of them is deleted.

```
apple
%%banana
banana
ORANGE
orange
Persimmon
```

6. To specify the character that separates fields:

```
  sort  -t:  +1  vegetables
```

This sorts **vegetables**, comparing the text that follows the first colon on each line. The **+1** tells **sort** to ignore the first field and to compare from the start of the second field to the end of the line. The **-t:** tells **sort** that colons separate fields.

If **vegetables** contains:

```
  yams:104
  turnips:8
  potatoes:15
  carrots:104
  green beans:32
  radishes:5
  lettuce:15
```

then **sort** displays:

```
  carrots:104
  yams:104
  lettuce:15
  potatoes:15
  green beans:32
  radishes:5
  turnips:8
```

The numbers are not in numeric order. This happened because a lexicographic sort compares each character from left to right. In other words, "**3**" comes before "**5**" and "**2**" comes before " ", so "**32**" comes before "**5** ".

7. To sort numbers:

```
  sort  -t: +1  -n  vegetables
```

This sorts **vegetables** numerically on the second field. If **vegetables** is the same as in Example 6, **sort** displays:

```
  radishes:5
  turnips:8
  lettuce:15
  potatoes:15
  green beans:32
  carrots:104
  yams:104
```

8. To sort on more than one field:

```
  sort  -t: +1  -2  -n  +0  -1  -r  vegetables
```

This performs a numeric sort on the second field (**+1 -2 -n**). Within that ordering, it sorts the first field in reverse alphabetic order (**+0 -1 -r**). The output looks like this:

```
  radishes:5
  turnips:8
  potatoes:15
```

```
lettuce:15
green beans:32
yams:104
carrots:104
```

Now the lines are sorted in numeric order. When two lines have the same number, they appear in reverse alphabetic order.

9. To replace the original file with the sorted text:

   ```
   sort  -o  vegetables  vegetables
   ```

   This stores the sorted output into the file **vegetables** (**-o vegetables**).

Files

sort.c	Contains sort definitions.

Related Information

See the following commands: "comm" in topic 1.1.83, "join" in topic 1.1.219, and "uniq" in topic 1.1.495.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.433 sortbib

Purpose
Sorts bibliographic data base.

Syntax

```
           +-----------+
sortbib ---¦           +--- database ---¦
           +- -s KEYs -+
```

Note: This command does not have MBCS support.

Description

The **sortbib** command sorts files of records containing **refer** key-letters by user-specified keys. Records may be separated by blank lines, or by .[and .] delimiters, but the two styles may not be mixed together. This program reads through each **database** and pulls out key fields, which are sorted separately. The sorted key fields contain the file pointer, byte offset, and length of corresponding records. These records are delivered using disk seeks and reads, so **sortbib** may not be used in a pipeline to read standard input.

By default, **sortbib** alphabetizes by the first %A and the %D fields, which contain the senior author and date. The **-s** option is used to specify new **KEYS**s. The most common key-letters and their meanings are listed with the **addbib** command. For instance, **-s**ATD will sort by author, title, and date, while **-s**A+D will sort by all authors, and date. Sort keys past the fourth are not meaningful. No more than 16 data bases may be sorted together at one time. Records longer than 4096 characters will be truncated.

The **sortbib** command sorts on the last word on the %A line, which is assumed to be the author's last name. A word in the final position, such as "jr." or "ed.", will be ignored if the name beforehand ends with a comma. Authors with two-word last names or unusual constructions can be sorted correctly by using the **nroff** convention "\0" in place of a blank. A %Q field is considered to be the same as %A, except sorting begins with the first, not the last, word. **sortbib** sorts on the last word of the %D line, usually the year. It also ignores leading articles (like "A" or "The") when sorting by titles in the %T or %J fields; it will ignore articles of any modern European language. If a sort-significant field is absent from a record, **sortbib** places that record before other records containing that field.

Related Information

See the following commands: "refer" in topic 1.1.365, "addbib" in topic 1.1.14, "roffbib" in topic 1.1.382, and "lookbib, indxbib" in topic 1.1.244.

1.1.434 sortm

Purpose
Sorts messages.

Syntax

```
         +-----------+   +-------- all --------------------------------+
sortm ---¦           +---¦ +----------- all -------------------------+ +---
         +- +folder -+   +-¦   +----------- sequence ------------+   +-+
                           +---¦  one of                          +---+
                               ¦ +-------+   +-----------------+ ¦ ¦
                             ¦ +-¦ num   +---¦      one of     +-+ ¦
                             ¦   ¦ first ¦   ¦¦+-------------+ ¦   ¦
                             ¦   ¦ prev  ¦   +-¦ :num   -prev +-+  ¦
                             ¦   ¦ cur   ¦     ¦ :+num  -cur  ¦    ¦
                             ¦   ¦ .     ¦     ¦ :-num  -.    ¦    ¦
                             ¦   ¦ next  ¦     ¦ -num   -next ¦    ¦
                             ¦   ¦ last  ¦     ¦ -first -last ¦    ¦
                             ¦   +-------+     +--------------+    ¦
                             +-------------------------------------+

    +- -datefield date --+   +----------------+
 ---¦                    +---¦     one of     +---¦
    +- -datefield field -+   ¦ +------------+ ¦
                             +-¦ -verbose   +-+
                               ¦ -noverbose ¦
                               +------------+

sortm --- -help ---¦
```

¦ Do not put a blank between these items.

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **sortm** command is used to sort messages. The **sortm** command is part of the Message Handling (MH) package and can be used with MH and AIX commands.

The **sortm** command sorts the messages according to the date in a header field. By default, **sortm** parses the **Date:**, but you can use the **-datefield** flag to specify another field. **sortm** numbers the sorted messages consecutively beginning with 1 (one). Messages that are in the folder, but not specified to be sorted, are placed after the sorted messages. **sortm** displays a message if it cannot parse a date field.

Flags

-datefield *field*	Specifies the header field to be used in the sort. The default field is **Date:**.
+*folder msgs*	Specifies the messages to be sorted. You can use the following message references when specifying *msgs*:

num	**first**	**prev**
cur	**.**	**next**
last	**all**	**sequence**

The default is all messages in the current folder. If you specify a **folder**, it becomes the current folder. The current message remains the current message, even if it moves during the sort.

-help	Displays help information for the command.
-noverbose	Does not display information during the sort. This flag is the default.
-verbose	Displays information during the sort. This information allows you to monitor the steps involved.

Profile Entries

Current-Folder:	Sets your default current folder.
Path:	Specifies your **user_mh_directory**.

Files

$HOME/.mh_profile	The MH user profile.

Related Information

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.435 sound

Purpose
Controls the volume and click of the keyboard speaker.

Syntax

```
         +-----------+   +--------+
sound ---¦  one of   +---¦ one of +---¦
         ¦ +-------+ ¦   ¦ +----+ ¦
         +-¦ -h -m +-+   +-¦ -c +-+
           ¦ -l -o ¦       ¦ -q ¦
           +-------+       +----+
```

Note: This command is for the PS/2 only.

Description
The **sound** command controls the volume of the sound output (the console bell). The system startup process sets the sound volume to on. You can run **sound** only from the console (**/dev/console**).

Flags

You must select at least one flag from the following two groups of flags or, optionally, one flag from each of the two groups. The first group of flags sets the volume of all sound output on or off:

-h, -l or -m
Turn the volume on.

-o Turns the volume off.

Note: The h (high), m (medium) and l (low) settings are for RT compatibility only.

The second group of flags is for RT compatibility only. They control whether click sounds are produced. On the PS/2, the click is mechanically produced and therefore cannot be controlled by the **sound** command.

-c Ignored

-q Ignored.

Example

To turn the volume on:

```
  sound -l
```

1.1.436 spell, spellin, spellout

Purpose
Finds spelling errors.

Syntax

```
        +-----------+   +------------+   +------------+
spell ---¦ +-------+ +---¦            +---¦            +---
        +-¦ -b -v +-+   +- -d hlist -+   +- -s hstop -+
          ¦ -i -x ¦¦
         ¦¦ -l    ¦¦
         ¦+-------+¦
         +---------+

    +----------------+   +-------------+   +--------+
 ---¦                +---¦             +---¦        +---¦
    +- -h spellhist -+   +- +wordlist -+   +- file -+ ¦
                                         +------------+

          +------------------+
spellin ---¦ +--------------+ +---¦
          +-¦ spellinglist +-+
            ¦ num          ¦¦
           ¦+--------------+¦
           +----------------+

           +------+
spellout ---¦      +--- list ---¦
           +- -d -+
```

Note: This command does not have MBCS support.

Description
The **spell** command reads words in **file** and compares them to those in a spelling list. Words that cannot be matched in the spelling list or derived from words in the spelling list (by applying certain inflections, prefixes, and/or suffixes) are written to standard output. If you do not specify a file to read, **spell** reads standard input.

The **spell** command ignores the same **troff**, **tbl**, and **eqn** constructs as the **deroff** command.

The coverage of the spelling list is uneven. You should create your own dictionary of special words used in your files.

Certain auxiliary files can be specified by file name parameters; see "Files" Copies of all output are accumulated in the history file.

The **spellin** command combines the words from the standard input and the prexisting **list** file and places a new list on the standard output. If no **list** file is specified, the new list is created from scratch.

The **spellout** command looks up each word from the standard input and prints on the standard output those that are missing from the hashed **list** file. Option **-d** reverses this, printing those that are present in the hashed **list** file.

Note: The hash tables created by **spellin** are binary files and are machine-dependant. If you are running **spell** in a heterogeneous TCF cluster, you must run **spell** and **spellout** on the same type of machine on which you ran **spellin**.

Three routines help maintain and check the hash lists used by **spell**.

/usr/lib/spell/hashmake	Reads a list of words from standard input and writes the corresponding nine-digit hash code to standard output.
/usr/lib/spell/spellin *num*	Reads *num* hash codes from standard input and writes a compressed spelling list to standard output.
/usr/lib/spell/hashcheck spellinglist	Reads a compressed **spellinglist** and recreates the nine-digit hash codes for all the words in it; it writes these codes to standard output.

Flags

-b	Checks British spelling.
-d hlist	Specify filename **hlist** as the alternate spelling list. The default is /usr/lib/spell/hlist [ab].
-hspellhist	Specify filename **spellhist** as the alternate history list which is used to accumlate all output. The default is /usr/lib/spell/spellhist.
-i	Suppresses processing of include files.
-l	Follows the chain of all included files (**.so** and **.nx** formatting commands). Without this flag, **spell** follows chains of all included files except for those beginning with **/usr/lib**.
-s hstop	Specify filename **hstop** as the alternate stop list which is used to filter out misspellings (for example thy-y+ier) that would otherwise pass. The default is /usr/lib/spell/hstop.
-v	Displays all words not literally in the spelling list and indicates plausible derivations from the words.
-x	Displays every plausible word stem with an = (equal sign).
+**wordlist**	Checks **wordlist** for additional word spellings. **wordlist** is the name of a file you provide that contains a sorted list of words one per line. With this flag, you can specify a set of correctly spelled words (in addition to **spell's** own spelling list) for each job.

Examples

1. To check your spelling:

```
spell  chap1  >mistakes
```

   This creates a file named **mistakes** containing all the words found in

chap1 that are not in the system spelling dictionary. Some of these may be correctly spelled words that **spell** does not know about. It is a good idea to save the output of **spell** in a file because the word list may be long.

2. To check British spelling:

   ```
   spell  -b  chap1  >mistakes
   ```

   This checks **chap1** against the British dictionary and writes the questionable words in **mistakes**.

3. To see how **spell** derives words:

   ```
   spell  -v  chap1  >deriv
   ```

   This lists the words that are not found literally in the dictionary, but are derived forms of dictionary words. The prefixes and suffixes used to form the derivative are indicated for each word. Words that do not appear in the dictionary at all are also listed.

4. To check your spelling against an additional word list:

   ```
   spell  +newwords  chap1
   ```

   This checks the spelling of words in **chap1** against the system dictionary and against **newwords**. The file **newwords** lists words in alphabetical order, one per line. You can create this file with a text editor, such as **ed**, and alphabetize it with the **sort** command.

5. To add a word to your spelling list:

   ```
   echo hookey  |  spellout /usr/dict/hlista
   echo hookey  |  spellin /usr/disct/hslista > myhlist
   spell -d myhlist huckfinn
   ```

   This example verifies that **hookey** is not on the default spelling list, adds it to the user's private list, and then uses it with **spell**. An alternative way is to place **hookey** into the sorted file **newwords** as in Example 4.

Files

D_SPELL=/usr/lib/spell/hlist[ab]	Hashed spelling lists, American and British.
S_SPELL=/usr/lib/spell/hstop	Hashed stop list.
H_SPELL=/usr/lib/spell/spellhist	History file.
/usr/lib/compress	Executable shell program to compress the history file.
/usr/lib/spell/spellprog	Program.

Related Information

See the following commands: "deroff" in topic 1.1.118, "eqn, neqn, checkeq" in topic 1.1.152, "sed" in topic 1.1.415, "sort" in topic 1.1.432, "tbl" in topic 1.1.463, "tee" in topic 1.1.467, and "troff" in topic 1.1.302.2.

1.1.437 spline

Purpose
Interpolates a smooth curve.

Syntax

```
                                           +-------------------------+
                              +------------¦             +---------+ +--------
          +- -k 0 -n 100 -+   ¦            +- -x lowlim -¦         +-+
spline ---¦ +-----------+ +---¦                          +- uplim -+         -
          +-¦  -k num   +-+   ¦      +-- 1 --+   +- -x 0 ------------------+ ¦
            ¦  -p       ¦¦    +- -a -¦       +---¦             +---------+ +--
           ¦¦  -n num   ¦¦           +- num -+   +- -x lowlim -¦         +-+
           ¦+-----------+¦                                     +- uplim -+
           +-------------+
```

Description
The **spline** command reads from the standard input pairs of numbers that represent the coordinates of a point on an x,y axis. From this input, **spline** calculates the coordinates of points to form a smooth curve through the points in the input set. It then writes these points to standard output. The output points are approximately equally spaced and includes the points that you provided as input. The cubic spline output has two continuous derivatives, and enough points so that when plotted with the **graph** command it looks smooth.

When data is not strictly monotone in **x**, **spline** reproduces the input without interpolating extra points.

You can only use 1,000 input points.

Flags

-a num	Supplies abscissas automatically; spacing is given by the next parameter or is assumed to be 1 if the next parameter is not a number.
-k num	Uses the constant **num** in the boundary value calculation: y0 =ky1 , y =**num**y The default for **num** is zero.
-n num	Spaces output points so that approximately **num** intervals occur between the lower and upper **x** limits (set with the **-x** flag). The default **num** is 100.
-p	Makes output periodic, that is, matches derivatives at ends. First and last input values should normally agree.
-x lowlim[uplim]	Sets lower and upper **x** limits as **lowlim** and **uplim**. Normally, these limits are calculated from the data. Automatic abscissas start at lower limit, defaults to zero.

1.1.438 split

Purpose
Splits a file into pieces.

Syntax

```
        +- -1000 -+   +--------+   +--- x ----+
split ---¦         +---¦        +---¦          +---¦
        +- - num -+   +- file -+   +- prefix -+
```

Description
The **split** command reads **file** and writes it in **num**-line pieces (default 1000 lines) to a set of output files. The name of the first output file is **prefixaa**, the second is **prefixab**, and so on lexicographically, through **prefixzz** (a maximum of 676 files). **prefix** cannot be longer than 12 characters. If you do not specify an output name, **x** is assumed.

If you do not specify an input file, or if you specify **-** (minus) in place of **file**, **split** reads standard input.

Examples

1. To split a file into 1000-line segments:

   ```
   split  book
   ```

   This splits **book** into 1000-line segments named **xaa**, **xab**, **xac**, and so forth.

2. To split a file into 50-line segments and specify the file name prefix:

   ```
   split  -50  book  sect
   ```

   This splits **book** into 50-line segments named **sectaa**, **sectab**, **sectac**, and so forth.

Related Information

See the following commands: "bfs" in topic 1.1.39 and "csplit" in topic 1.1.101.

1.1.439 splp

Purpose
Changes or displays printer driver settings.

Syntax

```
        +----------------+   +--------------------------+   +- /dev/lp -+
splp ---¦ +------------+ +---¦ one of       one of      +---¦           +---¦
        +-¦ indent=num +-+   ¦ +---+ ¦ +--------------+ ¦   +- device --+
          ¦ width=num  ¦¦    +-¦ + +---¦ fontinit bs  +-+
         ¦¦ length=num ¦¦      ¦ - ¦   ¦ parenb   ff  ¦¦
         ¦¦ timer=num  ¦¦     ¦+---+   ¦ parodd   tb  ¦¦
         ¦¦ csn        ¦¦     ¦        ¦ sync     cr  ¦¦
         ¦¦ speed      ¦¦     ¦        ¦ cstopb   nl  ¦¦
         ¦+------------+¦     ¦        ¦ wrap     cap ¦¦
         +--------------+     ¦        ¦ plot     err ¦¦
                              ¦        +--------------+¦
                              +------------------------+
```

¦ Do not put a blank between these items.

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **splp** command changes or displays settings for a printer driver (**device**). The default **device** is **/dev/lp0**. If you do not specify any flags, **splp** reports the current settings for the specified **device**. Select flags to change the current settings. No other processing is done, and there is no other output.

The changes that **splp** makes remain in effect until the next time you restart the system or rerun **splp**. You can run **splp** from the **/etc/rc** command file to configure your printer each time you startup the system.

Note: When the **print** command is used with the backend **piobe**, **splp** is set as **+plot**. Any parameters set by the user are ignored. If a file is redirected using the **cat** command instead of the **print** command, the **splp** settings are active.

Flags

timer=num	Sets the time out period to **num** seconds, where **num** is an integer.
indent=num	Indents **num** columns, where **num** is an integer.
length=num	Prints **num** lines per page, where **num** is an integer.
width=num	Prints **num** columns, where **num** is an integer
+bs (lpr-bs)	Sends (does not send) backspaces to the printer.
+cap (lpr-cap)	Converts (does not convert) all lowercase characters to uppercase.

+cr (lpr-cr)	Sends carriage returns (translates carriage returns to carriage return - line feeds).
+cstopb (lpr-cstopb)	Selects 2 (1) stop bits per character.
cs5 cs6 cs7 cs8	Selects character size. See **termio** in *AIX Operating System Technical Reference* for additional information on character size.
+err (lpr-err)	Issues (does not issue) a signal (SIGIOINT) upon receiving a error and attempts to resume I/O.
+ff (lpr-ff)	Sends form feeds (simulates a form feed with line feeds or carriage returns).
+fontinit (lpr-fontinit)	Indicates that fonts are (are not) loaded. Use this flag to control the initialization of fonts for the 3812 Pageprinter.
+nl (lpr-nl)	Sends line-feeds (translates line feeds to line feed - carriage returns).
+parenb (lpr-parenb)	Enables (disables) parity generation and detection.
+parodd (lpr-parodd)	Selects odd (even) parity.
+plot	Sends all characters to the printer unmodified. This overrides other settings.
-plot	Translates characters according to the settings.
+sync (lpr-sync)	Does not (does) return immediately without waiting for all data to be sent out.
+tb (lpr-tb)	Expands (does not expand) tabs on eight position boundaries.
+wrap (lpr-wrap)	Wraps (truncates) characters beyond the specified **width** to the next line and (with **+wrap**), prints "..." before the new-line character.
50 75 110 134 150 300 600 1200 1800 2400 4800 9600 exta extb	Sets the **speed** to the specified number of bits per second (**exta** is 19200).

Examples

1. To display the current printer settings:

```
splp
```

2. To change the printer settings:

```
splp  width=80  +wrap  +cap
```

   This changes the settings of the **/dev/lp** printer for 80-column paper (**width=80**). It wraps each line that is more than 80 columns wide onto a second line (**+wrap**), and prints all alphabetic characters in uppercase (**+cap**).

Related Information

See the following command: "qdaemon, lp" in topic 1.1.347.

See the **lp** file in *AIX Operating System Technical Reference*.

1.1.440 spost

Purpose
Delivers a message.

Syntax

```
         +- -alias /usr/lib/mh/MailAlias -+   +--- -format ---+
spost ---¦ +---- -noalias -----+        +---¦               +---
         +-¦                   +----------+   ¦    one of     ¦
           +--- -alias file ---+              ¦ +-----------+ ¦
                            ¦                 +-¦ -format   +-+
             +--------------+                   ¦ -noformat ¦
                                                +-----------+

    +--- -nofilter ----+   +- -width 72 --+   +-- -nowatch --+
 ---¦                  +---¦              +---¦              +---
    ¦      one of      ¦   +- -width num -+   ¦    one of    ¦
    ¦ +--------------+ ¦                      ¦ +----------+ ¦
    +-¦ -filter file +-+                      +-¦ -watch   +-+
      ¦ -nofilter    ¦                          ¦ -nowatch ¦
      +--------------+                          +----------+

    +--- -remove ---+   +-- -nobackup --+   +-- -noverbose --+
 ---¦               +---¦               +---¦                +--- file ---¦
    ¦    one of     ¦   ¦    one of     ¦   ¦     one of     ¦           ¦
    ¦ +-----------+ ¦   ¦ +-----------+ ¦   ¦ +------------+ ¦ +---------+
    +-¦ -remove   +-+   +-¦ -backup   +-+   +-¦ -verbose   +-+
      ¦ -noremove ¦       ¦ -nobackup ¦       ¦ -noverbose ¦
      +-----------+       +-----------+       +------------+

spost --- -help ---¦
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **spost** command is used to route messages to the proper destinations. The command is designed to be called by other programs rather than being run directly by users. The **spost** command is part of the Message Handling (MH) package.

The **spost** command searches all components of a message that specify a recipient's address and parses each address to check for proper format. The command puts proper addresses in the standard format and invokes the **sendmail** command. The **spost** command performs a similar function as the **post** command, but **spost** does less internal error checking than **post**.

Flags

-alias file	Searches the specified mail alias **file** for addresses. You can use this flag repetitively to specify multiple mail alias files. The **spost** command automatically searches the file **/usr/lib/mh/MailAliases**.
-backup	Renames the message file by placing a **,** (comma) before the file name after **spost** successfully posts the message.

-filter file	Uses the header components in the specified file for copies of the message sent to **bcc:** recipients.
-format	Puts all recipient addresses in a standard format for the delivery transport system. This flag is the default.
-help	Displays help information for the command.
-nofilter	Strips the **bcc:** header from the message and sends it to recipients specified in the **bcc:** component. This flag is the default.
-noalias	Does not use any alias files for delivering the message.
-nobackup	Does not rename the message after posting the file. This flag is the default.
-noformat	Does not alter the format of the recipient addresses.
-noremove	Does not remove the temporary message file after posting the message.
-noverbose	Does not display information during the delivery of the message to the **sendmail** command. This flag is the default.
-nowatch	Does not display information during delivery by the **sendmail** command. This flag is the default.
-remove	Removes the temporary message file after the successful completion of posting the message. This flag is the default.
-verbose	Displays information during the delivery of the message to the **sendmail** command. This information allows you to monitor the steps involved.
-watch	Displays information during the delivery of the message by the **sendmail** command. This information allows you to monitor the steps involved.
-width *num*	Sets the width of components that contain addresses. The default is 72 columns.

Files

$HOME/.mh_profile	The MH user profile.
/temp/pstnum	The temporary message file.

Related Information

See the following commands: "ali" in topic 1.1.17, "conflict" in topic 1.1.89, "mhmail" in topic 1.1.264, "post" in topic 1.1.320, "send" in topic 1.1.416, "sendmail, mailq, newaliases" in topic 1.1.417, and "whom" in topic 1.1.539.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See the *"Overview of the Message Handling Package"* in *Managing the AIX Operating System*.

1.1.441 spray

Purpose

Sends RPC packets to a host.

Syntax

```
              +-----------------+
              ¦ +-------------+ ¦
/etc/spray ---¦ ¦  -c count   ¦ +--- host ---¦
              +-¦  -d delay   +-+
                ¦  -i delay   ¦
                ¦  -l length  ¦
                +-------------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **spray** command sends a one-way stream of packets to **host** using RPC, and then reports how many were received by **host** and what the transfer rate was. The host name can be either a name or an internet address.

Flags

-c count
Specifies how many packets to send. The default value of **count** is the numbers of packets required to make the total stream size 100000 bytes.

-d delay
Specifies how many microseconds to pause between sending each packet. The default is 0.

-i Use ICMP echo packets rather than RPC. Since ICMP automatically echoes, this creates a two way stream.

-l length
The **length** parameter is the numbers of bytes in the ethernet packet that holds the RPC call message. Since the data is encoded using XDR, and XDR only deals with 32 bit quantities, not all values of **length** are possible, and **spray** rounds up to the nearest value. When **length** is greater than 1514, then the RPC call can no longer be encapsulated in one Ethernet packet, so the **length** field is no longer has a simple correspondence to Ethernet packet size. The default value of **length** is 86 bytes (the size of the RPC and UDP headers).

Related Information

See the following command: "rpc.sprayd" in topic 1.1.392.

See the **ping** command in the *TCP/IP User's Guide*.

1.1.442 stdhosts

Purpose
Converts a network hosts file into a standard format.

Syntax

```
            +------------+
stdhosts ---¦            +---¦
            +- filename -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **stdhosts** command converts a network hosts file such as **/etc/hosts** into a standard format. It does so by eliminating lines which may confuse certain older applications. Lines that are deleted include blank lines, comment lines and lines with no host names defined.

The input file can be specified on the command line (filename) or taken from standard input. The resulting file is written to standard output.

Related Information

See the **hosts** file format description in the *AIX TCP/IP Users Guide.*

1.1.443 store

Purpose
Changes file storage attribute on files in user-replicated file systems.

Syntax

```
         +-------+   +--- site ---+   +-----+   +------------+
store ---¦ +---+ +---¦          ¦ +---¦     +---¦            +---¦
         +-¦ + +-+   ¦ +-- , ---+ ¦ ¦ +- - -+   +--- file ---+
       ¦   ¦ - ¦     +--- all ----+ ¦                      ¦
       ¦   +---+                    ¦             +--------+
       +----------------------------+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **store** command sets the storage attribute for files in user-replicated filesystems. The **site** parameter is a site name or site number. The special name **all** refers to all present and future storage sites for the file. If a **-** precedes a **site** list, those sites are removed from the storage mask (the fstore field in the inode). If it is preceded by a **+**, those sites are added to the storage mask. If the **site** list is not preceded by a **+** or **-** modifier, the storage mask is set to exactly those sites. Additional **site** list arguments are combined, the effect being as if separate commands were done in the same order.

If the **file** list is empty or consists of a single **-**, the standard input is read for a list of file names. Otherwise, a single **-** argument can be used at the start of the file list to signal the end of the **site** lists. The **-** option is useful when file names begin with a **-** and might otherwise be mistaken as site list arguments.

Example

```
  store +alpha,beta myfile
```

This example causes the **myfile** to be replicated on sites named **alpha** and **beta**.

Diagnostics
If a given file is stored in a non-replicated or system-replicated filesystem, an error is given.

If the command is unable to change the storage mask on a file, an error message is displayed on the standard error and the command exits with a status of -1.

Files

/etc/site Site number/site name correspondences.

Related Information

See the following commands: "chfstore" in topic 1.1.64, "chmod" in topic 1.1.67, and "where" in topic 1.1.534.

See **fs** and **site** in the *AIX Operating System Technical Reference.*

1.1.444 strings

Purpose
Finds the printable strings in an object or other binary file.

Syntax

```
           +------+   +------+   +-----------+   +------------+
strings ---¦      +---¦      +---¦           +---¦            +---¦
           +- -a -+   +- -o -+   +- -number -+   +--- file ---+
                                                            ¦
                                                   +--------+
```

Note: This command does not have MBCS support.

Description

The **strings** command looks for strings in the specified binary files. It is useful for identifying random object files. To avoid unpredictable results, run **strings** on the same locale where the binary file on which you are running **strings** was compiled.

If no **file** is specified, standard input is read. A string is any sequence of four or more printable characters ending with a new-line or a null character. Unless the **-a** flag is given, **strings** looks only in the initialized data space of object files.

Flags

-a	Specifies that the entire object file is to be searched, rather than just the initialized data space.
-o	Precedes each string with its decimal offset from the beginning of the file.
-number	Specifies the minimum string length. The default is 4.

Related Information

See the following command: "od" in topic 1.1.304.

1.1.445 strip

Purpose
Removes symbol and line number information from a common object file.

Syntax

```
          +--------+   +------+
strip ---¦ one of +---¦      +--- file ---¦
          ¦ +----+ ¦   +- -l -+          ¦
          +-¦ -r +-+             +--------+
            ¦ -x ¦
            +----+
```

Description
The **strip** command removes the symbol table and line number information from common object files, including archive libraries. Once you use this command, symbolic debugging of the file is difficult; therefore, you should normally run **strip** only on production modules that you have debugged and tested. Using **strip** reduces the file storage overhead required by an object file.

For each object module, **strip** removes all symbol table information. For each archive, **strip** removes the library symbol table information. You can restore a stripped library symbol table to an archive or library file by using the **ar -s** command.

Flag

-l Removes only the line number information.

-r Does not remove static symbol, external symbol or relocation information.

Note: You must specify the **-r** flag if there are relocation entries in the object file.

-x Does not remove static or external symbol information.

Files

/usr/tmp/strp*

Related Information

See the following commands: "ar" in topic 1.1.23, "as" in topic 1.1.25, "cc" in topic 1.1.52, "dump" in topic 1.1.142, "ld" in topic 1.1.226, "nm" in topic 1.1.298, and "size" in topic 1.1.427.

See the **ar** and **a.out** files in *AIX Operating System Technical Reference*.

1.1.446 struct

Purpose
Structures FORTRAN programs.

Syntax

```
          +---------+   +--------+
struct ---¦ +-----+ +---¦        +---¦
          +-¦ -s  +-+   +- file -+
            ¦ -i  ¦¦
           ¦¦ -a  ¦¦
           ¦¦ -b  ¦¦
           ¦¦ -n  ¦¦
           ¦¦ -tn ¦¦
           ¦¦ -cn ¦¦
           ¦¦ -en ¦¦
           ¦+-----+¦
           +-------+
```

Description

The **struct** command translates the FORTRAN program specified by **file** (standard input default) into a Ratfor program. Wherever possible, Ratfor control constructs replace the original FORTRAN. Statement numbers appear only where still necessary. Cosmetic changes are made, including changing Hollerith strings into quoted strings and relational operators into symbols (for example ".GT." into ">"). The output is appropriately indented.

Notes:

1. The **struct** command knows FORTRAN 66 syntax, but not full FORTRAN 77.

2. If an input FORTRAN program contains identifiers which are reserved words in Ratfor, the structured version of the program is not a valid Ratfor program.

3. The labels generated cannot go above 32767.

4. If you get a goto without a target, try **-e**.

Flags

The following options may occur in any order.

-s Input is accepted in standard format, for example comments are specified by a c, C, or * in column 1, and continuation lines are specified by a nonzero, nonblank character in column 6. Normally input is in the form accepted by **f77**.

-i Do not turn computed goto statements into switches. (Ratfor does not turn switches back into computed goto statements.)

-a Turn sequences of else ifs into a non-Ratfor switch of the form:

```
switch
    {    case pred1: code
        case pred2: code
```

```
          case pred3: code
          default: code
      }
```

The case predicates are tested in order; the code appropriate to only one case is executed. This generalized form of switch statement does not occur in Ratfor.

-b Generate goto's instead of multilevel break statements.

-n Generate goto's instead of multilevel next statements.

-tn Make the nonzero integer **n** the lowest valued label in the output program (default 10).

-cn Increment successive labels in the output program by the nonzero integer **n** (default 1).

-en If **n** is 0 (default), place code within a loop only if it can lead to an iteration of the loop.

If **n** is nonzero, admit a small code segments to a loop if otherwise the loop would have exits to several places including the segment, and the segment can be reached only from the loop. 'Small' is close to, but not equal to, the number of statements in the code segment. Values of n under 10 are suggested.

Files

/tmp/struct*
/usr/lib/struct/*

1.1.447 stty, STTY

Purpose
Sets, resets, or reports work station operating parameters.

Syntax

```
        +------------------------+                 +------------+
stty ---¦ +--- specification ---+ +---¦      STTY ---¦   one of   +---¦
        +-¦                     ¦ +-+                 ¦ +--------+ ¦
          ¦ +-----------------+ ¦                   +-¦ -LCASE +-+
          ¦       one of        ¦                     ¦  LCASE ¦
          ¦       +----+        ¦                     +--------+
          +-------¦ -a +--------+
                  ¦ -g ¦
                  +----+
```

Description
The **stty** command sets certain work station I/O options for the device that is the current standard input (standard output if the **bsd** flag is used). If you run it without any **specification**s, **stty** writes to standard output information about the settings of certain options on your work station.

If you list any work station **specification**s, **stty** sets or resets the specified work station options.

The **STTY** command provides support for work stations with uppercase characters only. See the **LCASE** option described below.

You can find detailed information about the modes listed in the following groups in the discussion of the **termio** special facility in *AIX Operating System Technical Reference*.

Note: The **stty** command does not make compatibility checks on any parameter combinations.

Flags

-a	Writes the current state of all System V option settings to standard output. This is an extended version of the output generated when **stty** is run with no flags.
-g	Writes option settings to standard output in a form usable by another **stty** command.
all	Reports normally used option settings in **bsd** format.
bsd	Causes standard output to be used for settings; standard error is used for the report. **Note:** If the **bsd** flag is used, it must be the first flag.
everything	Prints everything **stty** knows in **bsd** format.
size	Prints the terminal (window) sizes on standard output, first rows and then columns.
speed	Prints only the terminal speed on standard output.

Specifications

The AIX terminal driver supports modes that are a superset of those provided by System V and by BSD. The **stty** command also supports output formats that are compatible with both systems. There are several different output formats generated by different flags.

The **stty** command with no flags reports on those control modes that differ from the system defaults.

The "**stty -a**" command reports on many modes in a System-V compatible format. Control modes added by BSD do not appear in the report. The interrupt, quit, erase, kill, eof and eol characters are displayed. The states of the modes listed below under the headings "Control Modes", "Input Modes", "Output Modes", "Local Modes" and "Paging Modes" are displayed. "Line = 0" indicates the old line discipline is in use, while "Line = 2" indicates the new line is in use. The terminal maps are also displayed.

The "**stty all**" command displays commonly-used terminal modes in a BSD-compatible format. The line discipline, line speed, tab settings, paging modes, and the state of all special characters are displayed.

The "**stty everything**" displays all terminal modes in a BSD-compatible format. The line discipline, line speed, all modes listed below under the heading "Job Control Modes", and all special characters are displayed.

Subtopics

1.1.447.1 Control Modes

The following options apply only when your work station connects to the system through an asynchronous line adapter. See **asy** in *AIX Operating System Technical Reference* for detailed information about this group.

parenb (-parenb)	Enables (disables) parity generation and detection.
parodd (-parodd)	Selects odd (even) parity.
cs5 cs6 cs7 cs8	Selects character size. See **termio** in *AIX Operating System Technical Reference* for additional information on character size.
0	Hangs up phone line immediately.
0 50 75 110 134 150 300 600 1200 1800 2400 4800 9600 19200 19.2 38400 38.4 exta extb	Sets the work station speed to the specified number of bits per second (**exta**, 19200, and 19.2 are synonyms; **extb**, 38400, and 38.4 are synonyms). Regardless of the baud rate, the software only works with terminals that generate the ASCII character set.
hupcl (-hupcl)	
hup (-hup)	Hangs up (does not hang up) dial-up connection on the last close.
cstopb (-cstopb)	Selects 2 (1) stop bits per character.

The next two options apply to all work stations, regardless of the line adapter:

cread (-cread)	Enables (disables) the receiver.
clocal (-clocal)	Assumes a line without (with) modem control.

1.1.447.2 Input Modes

ignbrk (-ignbrk)	Ignores (does not ignore) BREAK on input.
brkint (-brkint)	Signals (does not signal) INTR on break.
cbreak	Makes each character available to the **read** system call as received; no erase or kill processing, but all other processing (such as interrupt and suspend) is performed.
-cbreak	Makes characters available to **read** only when a newline is received.
ignpar (-ignpar)	Ignores (does not ignore) parity errors.
parmrk (-parmrk)	Marks (does not mark) parity errors.
inpck (-inpck)	Enables (disables) input parity checking.
istrip (-istrip)	Strips (does not strip) input characters to 7 bits.
inlcr (-inlcr)	Maps (does not map) NL to CR on input.
igncr (-igncr)	Ignores (does not ignore) CR on input.
icrnl (-icrnl)	Maps (does not map) CR to NL on input.
iuclc (-iuclc)	Maps (does not map) uppercase alphabetic characters to lowercase.
ixon (-ixon)	Enables (disables) START/STOP output control. Once START/STOP output control has been enabled, you can pause output to the work station by pressing **Ctrl-S** and resume output by pressing **Ctrl-Q**.
ixany (-ixany)	Allows any character (only **Ctrl-Q**) to restart output.
ixoff (-ixoff)	Sends (does not send) START/STOP characters when the input queue is nearly empty/full.

1.1.447.3 Output Modes

opost (-opost)	Processes output (does not process output; that is, it ignores all other output options).
olcuc (-olcuc)	Maps (does not map) lowercase alphabetic characters to uppercase on output.
onlcr (-onlcr)	Maps (does not map) NL characters to CR-NL characters.
ocrnl (-ocrnl)	Maps (does not map) CR-NL characters to NL characters.
onocr (-onocr)	Does not (does) output CR characters at column zero.
onlret (-onlret)	On the terminal, NL performs (does not perform) the CR function.
ofill (-ofill)	Uses fill characters (uses timing) for delays.
ofdel (-ofdel)	Uses DEL (NUL) characters for fill characters.
cr0 cr1 cr2 cr3	Selects style of delay for CR characters.
nl0 nl1	Selects style of delay for NL characters.
tab0 tab1 tab2 tab3	Selects style of delay for horizontal tabs.
bs0 bs1	Selects style of delay for backspaces.
ff0 ff1	Selects style of delay for form feeds.
vt0 vt1	Selects style of delay for vertical tabs.

1.1.447.4 Local Modes

isig (-isig)	Enables (disables) the checking of characters against the special control characters INTR and QUIT.
icanon (-icanon)	Enables (disables) **canonical input** (canonical input allows input-line editing with the ERASE and KILL characters).
xcase (-xcase)	Echoes (does not echo) uppercase characters on input, and displays uppercase characters on output with a preceding \ (backslash).
echo (-echo)	Echoes (does not echo) every character typed.
echoe (-echoe)	Echoes (does not echo) the ERASE character as the backspace-space-backspace string. **Note:** This mode does not keep track of column position, so you may get unexpected results when erasing tabs and escape sequences.
echok (-echok)	Echoes (does not echo) a NL character after a KILL character.
lfkc (-lfkc)	Functions the same as **echok**. This is an obsolete mode.
echonl (-echonl)	Echoes (does not echo) the NL character.
noflsh (-noflsh)	Does not clear (does clear) buffers after INTR or QUIT.
stwrap (-stwrap)	Disables (enables) truncation of lines longer than 79 characters on a synchronous line.
stflush (-stflush)	Enables (disables) flush on a synchronous line after every write.
stappl (-stappl)	Uses application mode (uses line mode) on a synchronous line.
xscan (-xscan)	Turns on or off scan code processing.

1.1.447.5 Control Assignments

Except for the **min** and **time** options, the following options allow you to specify which characters you must type to perform line editing and other special functions from your workstation. If **-c** is specified as **-u**, **undef** or -, the special function is disabled. A value of **x** (a two-character sequence) is interpreted as a control character. The two characters ^? can be used to represent DEL.

The **stty** command displays control characters as a ^ followed by another character. Depending on the output format selected, the character following the caret is chosen from a set of ASCII characters which includes the uppercase letters, or from a set of ASCII characters which includes the lowercase letters. For instance, Ctrl-A may be displayed as either ^a or ^A. The special case of having an undefined setting for a control function is displayed as a blank space or as either ^@ or ^'.

erase c	Sets erase character to **c**. (The default is **^H**.)
kill c	Sets kill character to **c**. (The default is **^U**.)
intr c	Sets interrupt character to **c**. (The default is **DEL** or **^?** (delete).
quit c	Sets quit character to **c**. (The default is ^V.)
start c	Sets start character to **c**. (The default is ^Q.)
stop c	Sets stop character to **c**. (The default is ^S.)
eof c	Sets end-of-file character to **c**. (The default is ^D.)
eol c	Sets end-of-line character to **c**. (The default is undefined.)
brk c	Sets break character to **c**. (The default is undefined). This character is an additional character causing wakeup.
nextpg c	If the **page** mode is enabled, resumes output when this character is typed after the screen has filled. The default value for this character is space. This character is recognized as special only when output has been stopped.
pgoff c	If the **page** mode is enabled, resumes output when this character is typed after the screen has filled, and temporarily disables paging mode until another input character is typed and read. The default values for this character is **Ctrl-M** (new line). Like **nextpg**, this character is recognized as special only when output has been stopped.
stat c	Displays the following information: (The default is **Ctrl-T**)

System Name

Topology	Indicates whether TCF is reconfiguring the network topology.

Uptime	Indicates how long the system has been up.
Load average	One-minute load average.
SPL	Percentage of time system is running at elevated priority level.
SVR	Percentage of time system is spending servicing TCF network messages.
Idle	Percentage of time system found no runnable processes.

susp c Sets suspend process character to **c**. The default is **Ctrl-Z**. Warning: If you are using the Bourne shell, entering the suspend process character causes you to log out.

dsusp c Sets delayed suspend process character to **c**. (The default is **Ctrl-Y**).

Warning:

The delayed suspend character is not interpreted by AIX.

rprnt c Sets reprint line character to **c**. (The default is **Ctrl-R**.)

flush c Sets flush output character to **c**. (The default is **Ctrl-O**.)

werase c Sets word erase character to **c**. (The default is **Ctrl-W**.)

lnext c Sets literal next character to **c**. (The default is **undefined**.)

min c Used with **-icanon**; a read request is not satisfied until at least **c** characters have been received or the timeout value specified by **time** has expired since the last character was received.

time c Used with **-icanon**; a read request is not satisfied until the timeout value specified by **c** has expired since the last character was received or at least **min** characters have been received.

1.1.447.6 Paging Options

page (-page)	Pauses (does not pause) during output after each screen displayed. Typing any character during the pause causes output to resume. Typing a carriage return during the pause causes output to continue uninterrupted until the next command is entered.
length n	Sets screen length to **n** lines, where **n** is an integer from 1 through 255. An automatic pause in output occurs after **n** lines if **page** is enabled.
row n	Records the terminal size as having **n** rows.
columns n	Records the terminal size as having **n** columns.
cols n	Is an alias for **columns**.
erasbell (-erasbell)	Rings (does not ring) the terminal bell whenever a character-erase, word-erase, or line-kill character is typed when there is no pending input to erase.
pgcbrk (-pgcbrk)	Stops (does not stop) output when the screen is full when in **cbreak** mode. This option does not affect **cooked** mode operation.
pgbell (-pgbell)	Rings (does not ring) the terminal bell when output stops on a full screen.

1.1.447.7 Combination Modes

evenp | parity	Enables **parenb** and **cs7**.
oddp	Enables **parenb**, **cs7**, and **parodd**.
-parity, **-evenp**, **-oddp**	Disables **parenb** and sets **cs8**.
raw (-raw | cooked)	Enables (disables) **raw** input and output (no ERASE, KILL, INTR, QUIT, EOT, or output processing).
nl (-nl)	Unsets (sets) **icrnl** and **onlcr**. Specifying **-nl** sets **icrnl** and **onlcr** and also unsets **inlcr**, **igncr**, **ocrnl**, and **onlret**.
lcase (-lcase)	
LCASE (-LCASE)	Sets **xcase**, **iuclc**, and **olcuc**. (Used for work stations with uppercase characters only.)
tabs (-tabs | tab3)	Preserve tabs (expand to spaces) when printing.
ek	Sets ERASE and KILL characters to # and @, respectively.
sane	Resets parameters to "reasonable" values.
term	Sets all parameters according to work station type **term**, where **term** is one of **tty33**, **tty37**, **vt05**, **tn300**, **ti700**, or **tek**.
even (-even)	Allows (disallows) even parity output.
odd(-odd)	Allows (disallows) odd parity output.
tandem(-tandem)	Enables (disables) flow control so that the system sends out the stop character when its internal queue is in danger of overflowing on input, and sends the start character when it is ready to accept further input.

1.1.447.8 Terminal Mapping

imap mapname	Loads **/etc/nls/termmap/mapname.in** as the terminal input map.
omap mapname	Loads **/etc/nls/termmap/mapname.out** as the terminal output map.

1.1.447.9 Job Control Modes

new	Switches to the BSD new **tty** driver line discipline (switching flushes typeahead).
old	Switches to the default **tty** driver line discipline (switching flushes typeahead).

The following options apply only when the new **tty** driver line discipline is being used:

crt	Sets options for a CRT (**crtbs**, **ctlecho**, and, if = 1200 baud, **crterase** and **crtkill**).
crtbs	Echoes backspaces on erase characters.
prterase	Echoes erased characters backwards within "\" and "/" for a printing terminal.
crterase (-crterase)	Wipes out (leaves visible) erased characters with backspace-space-backspace.
crtkill (-crtkill)	Wipes out (leaves visible) input on kill (**like crterase**).
ctlecho (-ctlecho)	Echoes (does not echo) control characters as **^x** and delete as **^?** (as themselves). Prints (does not print) two backspaces following the EOT character (**Ctrl-D**).
decctlq (-decctlq)	After output is suspended, only a start character (any character typed) will restart it. **Note:** **decctlq** is the same as **ixany**; **-decctlq** is the same as **-ixany**.
tostop (-tostop)	Stops background jobs (allows background jobs to proceed) if they attempt terminal output.
tilde (-tilde)	Converts (does not convert) a tilde (~) to a backslash-circumflex (\^) on output for Hazeltine terminals. (Not implemented.)
flusho (-flusho)	Discards (does not discard) output usually because **Ctrl-D** is pressed (internal state bit).
pendin (-pendin)	Input is (is not) pending after a switch from **cbreak** to **cooked** and will be re-input when a read becomes pending or more input arrives (internal state bit).
mdmbuf (-mdmbuf)	Starts (stops) output on carrier transitions (returns error is write attempted after carrier drops). (Not implemented.)
litout (-litout)	Sends ouput characters without any processing (does normal output processing).

nohang (-nohang)	Does not send (does send) hangup signal to the control process group when carrier drops.
etxack (-etxack)	Enables (disables) Diablo type **etx/ack** handshaking. (Not implemented.)
pass8 (-pass8)	Passes (does not pass) all 8 bits on input, in any mode. If **-pass8** is used, the 0200 bit is stripped on input except in raw mode.

Examples

1. To display a short listing of your work station configuration:

   ```
   stty
   ```

   This lists settings that differ from the defaults.

2. To display a full listing of your work station configuration:

   ```
   stty  -a
   ```

3. To enable a key sequence that stops listings from scrolling off the screen:

   ```
   stty  ixon  ixany
   ```

   This sets **ixon** mode, which lets you stop runaway listings by pressing **Ctrl-S**. The **ixany** parameter allows you to resume the listing by pressing any key. The normal work station configuration includes **ixon** and **-ixany**, which allows you to stop a listing with **Ctrl-S**, but only **Ctrl-Q** will restart it.

4. To prevent all listings from scrolling off the screen:

   ```
   stty  page  length  24
   ```

   This sets **page** mode with a page (screen) length of **24** lines. When a listing is more than 24 lines long, the system pauses after each page. It beeps, reminding you to press any key (except the carriage return) to view the next page. Press the carriage return to let the rest of the listing scroll off the screen and get to the end. Paging then resumes with the next listing.

5. To reset the configuration after it has been altered:

   Ctrl-J stty sane echoe -tabs **Ctrl-J**

   Sometimes the information displayed on the screen may look strange, or the system will not respond when you press the **Enter** key. This can happen when you use **stty** with parameters that are incompatible or that do things you do not understand. It can also happen when a screen-oriented text editor ends abnormally and does not have a chance to reset the work station configuration.

   Entering **stty sane** sets a reasonable configuration, but it may differ slightly from your normal configuration. That is why this example also includes two commonly used parameters, **echoe** (erases characters as you backspace over them) and **-tabs** (expand tab characters to spaces on the display screen).

Press **Ctrl-J** before and after the command instead of **Enter**. The system usually recognizes **Ctrl-J** when the parameters that control the **Enter** key processing have been confused.

6. To save and restore the work station's configuration:

```
OLDCONFIG=`stty  -g`          #  save  configuration
stty  -echo                #  do not  display  password
echo  "Enter password:  \c"
read  PASSWD               #  get  the  password
stty  $OLDCONFIG           #  restore  configuration
```

   This saves the work station's configuration, turns off echoing, reads a password, and restores the original configuration. The **`...`** (grave accents) in the first command tell the shell to insert the standard output of **stty -g** into the **OLDCONFIG=...** command. This is called **command substitution**. For more information, see "Command Substitution" in topic 1.1.420.5.

   The **stty -echo** turns off echoing, which means that the password does not appear on the screen when you type it at the keyboard. This has nothing to do with the **echo** command, which displays a message on the screen.

Related Information

See the following command: "tabs" in topic 1.1.459.

See the **ioctl** system call and the **termio** special file in *AIX Operating System Technical Reference*.

See the discussion of **stty** and the "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.448 style

Purpose
Analyzes the surface characteristics of a document.

Syntax

```
         +-----------------+
style ---¦ +-------------+ +--- file ---¦
         +-¦ -ml  -l num +-+
           ¦ -mm  -r num ¦
           ¦ -a   -P     ¦
           ¦ -e   -p     ¦
           +-------------+
```

Note: This command does not have MBCS support.

Description

The **style** command analyzes the surface characteristics of the writing style of a document. It reports on readability, sentence length and structure, word length and usage, verb type, and sentence openers. Because **style** runs **deroff** before looking at the text, formatting header files should be included as part of the input. The default macro package **-ms** may be overridden with the flag **-mm**. The flag **-ml** , which causes **deroff** to skip lists, should be used if the document contains many lists of non-sentences. The other options are used to locate sentences with certain characteristics.

Notes:

1. This command does not have National Language Support.

2. Use of nonstandard formatting macros can cause incorrect sentence breaks.

Flags

-a	print all sentences with their length and readability index.
-e	print all sentences that begin with an expletive.
-p	print all sentences that contain a passive verb.
-l num	print all sentences longer than **num**.
-r num	print all sentences whose readability index is greater than **num**.
-P	print parts of speech of the words in the document.

Related Information

See the following commands: "deroff" in topic 1.1.118 and "diction, explain" in topic 1.1.123.

1.1.449 su

Purpose
Obtains the privileges of another user, including superuser authority.

Syntax

```
     +-----+   +------+   +- root -+   +------------------+
su ---¦    +---¦     +---¦        +---¦                  +---¦
     +- - -+   +- -f -+   +- user -+   +- -c "cmdstring" -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **su** command runs a shell and lets you operate in it with the privileges of the specified **user** (by default, **root**).

If you use **su** to become the superuser (the **user** argument is **root**), the **su** command sets the PATH variable to **/bin:/usr/bin:/usr/ucb:/etc**, sets the $HOME variable to **/**, and changes the prompt to # (number sign). The PATH variable does not include the current directory. If you are not already operating with superuser authority, the **su** command prompts for the password associated with **user** before granting these privileges.

To restore your normal privileges, use the standard procedure for exiting from the shell. This often is pressing END OF FILE (**Ctrl-D**). This action ends the shell called by **su** and returns you to the previous shell and ID.

If you need to run only one command as **user**, you can run the desired command by including it (along with any of its associated flags) on the command line as an argument to the shell **-c** flag (see "sh, Rsh" in topic 1.1.420 for a description of this flag). In this case, **su** calls **sh** to run the command and then exits automatically.

Each time someone uses **su** to become the superuser, **su** writes a record in the file **/usr/adm/sulog**, creating this file if necessary. This record is written regardless of whether **su** is successful.

Note: If the **-c** option is not specified, **su** execs the shell listed in the shell field of the **/etc/passwd** file. If the **-c** flag is specified, **su** ignores the **passwd** file entry and runs **/bin/sh**. All exported environment variables are available unless you use the **-** flag when you call **su**.

Flags

- Creates the same environment for the new shell as the login shell of **user**. This is done by calling the new shell as a login shell (see "sh, Rsh" in topic 1.1.420), so it reads the system **profile** file and the **user**'s **$HOME/.login** or **$HOME/.profile** file, depending on the shell. The environment variables **LANG** and **LC_TIME** control the appearance of the date and time. The TERM and TZ variables are an exception. They are preserved at their current values. These variables are normally set by **init** or **getty** prior to login; so **su** handles them differently.

Note: This flag modifies the environment of the current shell only if the optional program named in the shell field of the **passwd** file is a program like **sh** that expects to be called as a login shell.

-c "cmdstring" Runs the **/bin/sh** shell, processes the specified **command**, and then exits the shell. This flag causes **su** to ignore the shell specified in the **passwd** file.

-f Prevents sourcing of the **.cshrc** or **.kshrc** file for the user being substituted, hence making start-up faster.

Examples

1. To obtain superuser authority:

   ```
   su
   ```

   This runs a subshell with the effective user ID and privileges of user **root**. The **su** command asks for a password, as if you were logging in as **root**. Now the commands you run have superuser authority. Press END OF FILE (**Ctrl-D**) to end the subshell and return to your original shell session and privileges.

2. To obtain **ann**'s privileges:

   ```
   su  ann
   ```

   This runs a subshell with the effective user ID and privileges of **ann**.

3. To set up the environment as if you had logged in as **ann**:

   ```
   su  -  ann
   ```

   This runs a subshell with the effective user ID and privileges of **ann**. The **-** causes the shell variable LOGNAME to be set to **ann**, HOME to be set to the path name of **ann**'s home directory, and **ann**'s **$HOME/.profile** shell procedure file to be run before prompting for the first shell command.

4. To run a single command with superuser authority:

   ```
   su  root  -c "backup  -9  -u"
   ```

   This runs the shell command **backup -9 -u** with superuser authority (if you know the password assigned to **root**).

Related Information

See the following commands: "sh, Rsh" in topic 1.1.420.

1.1.450 sum

Purpose
Displays the checksum and block count of a file.

Syntax

```
       +------+
sum ---¦+----++-- file --¦
       +¦ -r ++         ¦
        ¦ -o ¦ +--------+
        +----+
```

Description
The **sum** command reads **file** and calculates a 16-bit checksum and the number of blocks (in units of 512 bytes) in the file (or in standard input if no **files** are specified). The checksum and number of blocks are written to standard output. When you specify more than one **file**, **sum** also displays the name of each file. The **sum** command is generally used to determine if a file that has been copied or communicated over transmission lines is an exact copy of the original.

Flags

-r Compute byte-by-byte checksum. This is the default action.

-o Compute short-by-short checksum.

Example

To display the checksum of, and the number of blocks in **datafile**:

```
  sum  -r datafile
```

If the checksum of **datafile** is **1605** and if the file contains **3** blocks, **sum** displays:

```
  1605     3
```

Related Information

See the following command: "wc" in topic 1.1.530.

1.1.451 swapon

Purpose
Specifies additional devices for paging and swapping.

Syntax

swapon --- -a ---¦

swapon --- **device_name** ---¦

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **swapon** command is used to specify additional devices on which paging and swapping are to take place. The system begins by swapping and paging on only a single device so that only one disk is required at bootstrap time. Calls to **swapon** normally occur in the system multi-user initialization file **/etc/rc** making all swap devices available, so that the paging and swapping activity is interleaved across several devices.

The second form gives individual block devices as given in the system swap configuration table. The call makes only this space available to the system for swap allocation.

Normally, the **-a** argument is given, causing all swap or paging devices (as determined by the "swap" attribute being true) in **/etc/filesystems** to be made available.

Files

/etc/filesystem	Descriptions of mountable file systems
/dev/[r]hd?	Possible paging devices.

Related Information

See the following commands: "swapon" and "rc" in topic 1.1.354.

1.1.452 symorder

Purpose
Rearranges a name list.

Syntax

```
symorder --- orderlist --- symbolfile ---¦
```

Description

The **orderlist** file contains symbols to be found in **symbolfile**, 1 symbol per line.

The **symbolfile** is updated in place to put the requested symbols first in the symbol table, in the order specified. This is done by swapping the old symbols in the required spots with the new ones. If all of the order symbols are not found, an error is generated.

This program was specifically designed to cut down on the overhead of getting symbols from the kernel.

Related Information

See the following commands: "ld" in topic 1.1.226 and "nm" in topic 1.1.298.

1.1.453 sync

Purpose
Updates the superblock and writes buffered files to the fixed disk.

Syntax

```
sync ---¦
```

Description
The **sync** command runs the **sync** system call. If you stop the system without using the **shutdown** command, you must run **sync** to ensure file system integrity. **sync** writes all unwritten system buffers to disk. This includes modified superblocks, modified inodes and delayed block I/O. The **sync** command does not write out inodes for files opened with **O_DEFERC**.

Note: The writing, although scheduled, is not necessarily complete upon return from the **sync** system call. Therefore, you might want to call **sync** several times.

Related Information

See the **sync** system call in *AIX Operating System Technical Reference*.

1.1.454 syncfsmap

Purpose
Cleans up and updates **/etc/fsmap** stanzas for a site.

Syntax

```
              +------+   +------+   +-----------+
syncfsmap ---¦      +---¦      +---¦           +--- site ---¦
              +- -v -+   +- -d -+   +- -f file -+
```

Description

The **syncfsmap** command removes all file system stanzas for site from **/etc/fsmap** and (if **-d** flag is not set) appends the site's file systems file to **/etc/fsmap**.

Flags

-v	verbose mode.
-f file	uses the specified **file** instead of the default <LOCAL> file system.
-d	deletes the site from the **/etc/fsmap**.

If successful, the **syncfsmap** command exits with status 0. On failure, it prints out a diagnostic message to standard error and exits with status **-1**.

Related Information

See **filesystems** in the *Technical Reference Vol II*.

1.1.455 sysdump

Purpose

sysdump - initiates kernel core dump

Syntax

```
           +------+
sysdump ---¦      +---¦
           +- -p -+
```

Note: This command is for the System/370 only.

Note: This command does not have MBCS support.

Description

The **sysdump** command allows a system administrator to initiate a kernel core dump of an AIX/370 system. The core dump is written to **/dev/dump** and may later be extracted with the **savecore** command and analyzed with the **crash** command.

Sysdump allows the system administrator to specify whether the AIX/370 system should continue running after the kernel core dump, or whether the system should be halted with a panic.

Running this command will likely cause an interruption of service for all users accessing this AIX/370 site while the core dump is being written. Even if the system administrator elects to allow the system to continue running after writing the core dump, this interruption in service may cause other TCF cluster sites to believe that this AIX/370 system has gone down and then returned. Thus, this command should be run only after users on all sites in the cluster have been notified with the **wall** command, or after the problem site has been removed from the cluster using the **clusterstop** command.

Flag

-p	Force a system panic. Without this flag, the system will resume operation.

Related Information

See the following commands: "crash" in topic 1.1.96, "savecore" in topic 1.1.408, "wall" in topic 1.1.529, "clusterstart, clusterstop" in topic 1.1.75.

1.1.456 sysline

Purpose
Displays system status on the status line of a terminal.

Syntax

```
           +-----------+   +-------------+   +-------+
sysline ---¦ +-------+ +---¦             +---¦       +---¦
           +-¦ -b -i +-+   +- -H remote -+   +- + N -+
             ¦ -c -l ¦
            ¦¦ -d -m ¦¦
            ¦¦ -D -p ¦¦
            ¦¦ -e -q ¦¦
            ¦¦ -w -r ¦¦
            ¦¦ -h -s ¦¦
            ¦¦ -j    ¦¦
            ¦+-------+¦
            +---------+
```

Description

The **sysline** command runs in the background and periodically displays system status information on the status line of the terminal. Not all terminals contain a status line. If no flags are given, the **sysline** command displays the time of day, the current load average, the change in load average in the last 5 minutes, the number of users (followed by a "u"), the number of processes able to be run (followed by an "r"), the number of suspended processes (followed by an "s"), and the users who have logged on and off since the last status report. Finally, if new mail has arrived, a summary of it is printed. If there is unread mail in your mailbox, an asterisk will appear after the display of the number of users. The display is normally in reverse video (if your terminal supports this in the status line) and is right justified to reduce distraction. Every fifth display is done in normal video to give the screen a chance to rest.

If you have a file named .who in your home directory, then the contents of that file is printed first. One common use of this feature is, when using **csh** (c-shell) only, to alias chdir, pushd, and popd to place the current directory stack in **$HOME/.who** after it changes the new directory.

Flags

-b	Beep once every half hour and twice every hour.
-c	Clear the status line for 5 seconds before each re-display.
-d	Debug mode -- print status line data in human readable format.
-D	Print out the current day/date before the time.
-e	Print out only the information. Do not print out the control commands necessary to put the information on the bottom line.
-w	Window mode -- print the status on the current line of the terminal, suitable for use inside a one line window.
-H remote	Print the load average on the remote host **remote.** If the host is down, or is not sending out **rwhod** packets, then the down

time is printed instead. If the prefix "ucb" is present, then it is removed.

-h	Print out the host machine's name after the time.
-l	Don't print the names of people who log in and out.
-m	Don't check for mail.
-p	Don't report the number of process which are able to be run and suspended.
-r	Don't display in reverse video.
+N	Update the status line every N seconds. The default is 60 seconds.
-q	Don't print out diagnostic messages if something goes wrong when starting up.
-i	Print out the process ID of the **sysline** process onto standard output upon startup. With this information you can send the alarm signal to the **sysline** process to cause it to update immediately. **sysline** writes to the standard error, so you can redirect the standard output into a file to catch the process id.
-s	Print "short" form of line by left-justifying if escapes are not allowed in the status line. Some terminals (the Televideos and Freedom 100 for example) do not allow cursor movement (or other "intelligent" operations) in the status line. For these terminals, **sysline** normally uses blanks to cause right-justification. This flag will disable the adding of the blanks.
-j	Force the sysline output to be left justified even on terminals capable of cursor movement on the status line.

If you have a file .syslinelock in your home directory, then **sysline** will not update its statistics and write on your screen, it will just go to sleep for a minute. This is useful if you want to momentarily disable **sysline**. It may take a few seconds from the time the lock file is created until you are guaranteed that **sysline** will not write on the screen.

Files

/etc/utmp	Names of people who are logged in.
/dev/kmem	Contains process table.
/usr/spool/rwho/whod.*	Who/uptime information for remote hosts.
${HOME}/.who	Information to print on bottom line.
${HOME}/.syslinelock	When it exists, **sysline** will not print.

Related Information

See the **terminfo** file in *AIX Operating System Technical Reference.*

1.1.457 syslogd

Purpose
Logs system messages.

Syntax

```
          +-----------------+   +------------------+   +------+
syslogd ---¦                +---¦                  +---¦      +---¦
          +- -f configfile -+   +- -mmarkinterval -+   +- -d -+
```

Description
The **syslogd** command reads and logs messages into a set of files described by the configuration file **/etc/syslog.conf**. This daemon configures itself when it starts up and whenever it receives a hangup signal.

Each message read by **syslogd** is one line. A message can contain a priority code (marked by a number in **< >** brackets at the beginning of the line) and message text. Priorities are defined in **sys/syslog.h**. The **syslogd** command reads from the AIX domain socket **/dev/log** or from an Internet domain socket specified in **/etc/services**.

Each line in the **syslogd** configuration file must consist of two parts:

    A selector to determine the message priorities to which the lin
    applies
    An action

The two fields must be separated by one or more tabs. Here is an example of the line in a configuration file:

```
  mail.info;*.notice       /usr/spool/adm/syslog
```

The first part, the selector, is semicolon-separated list of priority specifiers. Each priority specifier consists of a facility describing the part of the system that generated the message, a **.** (period), and a level indicating the severity of the message. Symbolic names may be used and an ***** (asterisk) specifies all facilities. All messages of the specified level or higher (greater severity) are selected. In the previous example, **syslogd** selects the **mail** facility at the **info** level (or higher) and all facilities at the **notice** level (or higher).

More than one facility may be selected using commas to separate them. For example:

```
  *.emerg;mail,daemon.crit
```

selects all facilities at the **emerg** level (or higher) and the **mail** and **daemon** facilities at the **crit** level (or higher).

Known facilities and levels recognized by **syslogd** are those listed under **syslog** in the *AIX Operating System Technical Reference*. When you specify the name of a facility or level in a **syslogd** configuration file, omit the **LOG_** prefix used by **syslog** in the name. For example, **syslog** lists **LOG_DEBUG** as the lowest level. To specify this level in a **syslogd** configuration file, specify **debug**.

In addition to these facilities, there is a **mark** facility. This facility has messages at priority **info** sent to it every 20 minutes. You can change the mark time interval with the **-m** flag. The **mark** facility is not enabled by a facility field containing an asterisk; you must explicitly enable it. For example:

```
  kern,mark.debug
```

logs kernel messages and 20 minute marks of **debug** level (or higher).

The level none may be used to disable a particular facility. For example:

```
  *.debug;mail.none
```

logs all messages except mail messages.

The second part of each line, the action, describes where the message is to be logged if the line is selected. There are four forms:

A file name beginning with a leading **/** (Selected messages are appended to this file)
A host name preceded by a **@** (Selected messages are forwarded to **syslogd** on the named host)
A comma-separated list of users (Selected messages are written t those users, if they are logged in)
An ***** (Selected messages are written to all logged-in users).

For example:

```
  *.crit                 /usr/adm/critical
  kern.err               @nick
  *.alert                bobbi,kristi
  *.emerg                *
```

logs critical (or higher) messages into **/usr/adm/critical**, forwards kernel messages of error severity (or higher) to **syslogd** on the host **nick**, informs the users **bobbi** and **kristi** of any alert (or higher) messages, and informs all logged-in users of any emergency messages.

Blank lines and lines beginning with # are ignored.

The **syslogd** command creates the file **/etc/syslog.pid**, containing a single line with its process id. This file can be used to kill or reconfigure **syslogd**. To bring **syslogd** down, it should be sent a terminate signal. For example:

```
  kill `cat /etc/syslog.pid`
```

Flags

-d	Turns on debugging.
-f configfile	Specifies an alternate configuration file.
-m markinterval	Specifies the number of minutes between mark messages.

Examples

To start **syslogd** daemon and change the mark interval:

```
syslogd -m30
```

This command changes the mark interval to 30 minutes. If the configuration file contains:

```
  kern,mark.notice        /usr/adm/notice
  kern.err                @scott
  *.info;mail.none        /usr/spool/adm/syslog
  *.alert;auth.warning    darlene
```

syslogd logs kernel messages and 30 minute marks at **notice** level (or higher) in the file **/usr/adm/notice**, forwards kernel messages at **err** level (or higher) to **syslogd** on the host **scott**, logs messages at **info** level (or higher) except mail messages in the file **/usr/spool/adm/syslog**, and informs the user **darlene** of any warning message (or higher) from the authorization system.

Files

/etc/services	Contains definition of the internet domain socket.
/etc/syslog.conf	Contains the configuration file.
/etc/syslog.pid	Contains the process id.
/dev/log	Contains AIX domain datagram log socket.

Related Information

See the **syslog** system call in *AIX Operating System Technical Reference*.

1.1.458 tab, untab, expand, unexpand

Purpose
Changes spaces into tabs or tabs into spaces.

Syntax

```
       +------+   +--------+
tab ---¦      +---¦        +---¦
       +- -e -+   +- file -+
                          ¦
                   +------+


         +--------+
untab ---¦        +---¦
         +- file -+
                 ¦
          +------+
```

expand [-tabstop] [-tab1, tab2, . . ., tabn] [file . . .]
unexpand [-a] [file . . .]

Description
The **tab** command reads **file**s (standard input by default), replaces spaces in the input with tab characters wherever it can eliminate one or more spaces. It writes the resulting file back to **file** or, if the input was standard input, to standard output. **tab** assumes that the tab stops are set every eight columns starting with column nine.

The **untab** command reads **file**s or standard input, replaces tabs in the input with space characters and writes back to the original file or to standard output.

Expand processes the named files or the standard input writing the standard output with tabs changed to blanks. Backspace characters are preserved in the output and decrement the column count for tab calculations. **Expand** is useful for pre-processing character files (before sorting, looking at specific columns, etc.) that contain tabs.

Unexpand puts tabs back into the data from the standard input or the named files and writes the result on the standard output. By default, only leading blanks and tabs are reconverted to maximal strings of tabs.

Flag

For the tab command:

-e Replaces only those spaces at the beginning of a line up to the first non-space character.

For the **expand** command:

-tabstop
Sets tabs at **tabstop** spaces apart instead of the default (8).

-tab1, tab2, . . .,tabn
Sets tabs at specified columns.

For the **unexpand** command:

-a Inserts tabs wherever their presence compresses the resultant file by replacing two or more characters.

Related Information

See the following command: "newform" in topic 1.1.288.

1.1.459 tabs

Purpose
Sets tab stops on work stations.

Syntax

```
        +--- -8 ----+   +---- -T$Term -----+   +----- +m0 -------+
tabs ---¦           +---¦                  +---¦       +- 10 --+ +---¦
        +- tabspec -+   +- -T workstation -+   +- +m --¦¦      +-+
                                                       +- num -+


-----------------
¦ Do not put a space between these items.
```

Note: This command is for the PS/2 only.

Description
The **tabs** command clears up to 20 previous tabs and sets up to40 tabs on the work station according to the supplied **tabspec**. **tabspec** can be either a flag indicating an available code or column numbers. The available codes cover formats required by most structured programming languages.

When you use the **tabs** command, always see the leftmost column number as 1, even if your work station refers to it as zero (0).

If you do not specify a **tabspec**, the default value is **-8**.

Tabspecs

-a Sets the tabs to 1, 10, 16, 36, and 72 (IBM S/370 Assembler first format)

-a2 Sets the tabs to 1, 10, 16, 40, and 72 (IBM S/370 Assembler second format)

-c Sets the tabs to 1, 8, 12, 16, 20, and 55 (COBOL normal format)

-c2 Sets the tabs to 1, 6, 10, 14, and 49 (COBOL compact format, columns 1-6 omitted). With this code, the first column position corresponds to card column 7. One space gets you to column 8, and a tab reaches column 12. Files using this code should include a format specification of:

```
<:t-c2  m6  s66  d:>
```

For an explanation of format specifications, see the **fspec** file in *AIX Operating System Technical Reference*.

-c3 Sets the tabs to 1, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62, and 67 (COBOL compact format with more tabs than **-c2**. This is the recommended format for COBOL. Files using this code should include a format specification of:

```
<:t-c3  m6  s66  d:>
```

-f Sets the tabs to 1, 7, 11, 15, 19, and 23 (FORTRAN).

-p Sets the tabs to 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, and 61 (PL/I).

-s Sets the tabs to 1, 10, and 55 (SNOBOL).

-u Sets the tabs to 1, 12, 20, and 44.

In addition to the preset formats, three other types of **tabspec**s are available:

-num Sets regularly repeating tabs at every **num**th column. (**-8** is the standard AIX tab setting and the one required for use with the **nroff -h** flag.) Another special case is **-0**, which implies no tabs at all.

num[,num]... Sets tabs at the named column numbers (a comma-separated list in ascending order). You may specify up to 40 numbers. If any number except the first has a plus sign prefix, the prefixed number is added to the previous number for the next setting. Thus, the tab lists **1,10,20,30** and **1,10,+10,+10** provide the same tab settings.

--filep Reads the first line of the named **filep** for a format specification. If it finds one, it sets tabs the same way. If it does not find a format specification, it sets tabs to the system default (**-8**). Use this **tabspec** to make sure that a file has the same tab settings as those in a file already correctly formatted.

Flags

Note: If the same flag occurs more than once, only the last one takes effect.

-Tworkstation Identifies the work station so that **tabs** can set tabs and margins correctly.

300	DASI 300.
300-12	DASI in 12-pitch.
300s	DASI 300s.
300s-12	DASI 300s in 12-pitch
450	DASI 450.
450-12	DASI 450 in 12-pitch.
1620	Diablo 1620 (alias DASI 450).
1620-12	Diablo 1620 (alias DASI 450) in 12-pitch.
2621	Hewlett-Packard 2621, 2640, and 2645.
2640	Hewlett-Packard 2621, 2640, and 2645.
2645	Hewlett-Packard 2621, 2640, and 2645.
4014	Tektronix 4014
hp	Hewlett-Packard 2621, 2640, and 2645.
tek	Tektronix 4014.

If you do not provide a **-T** flag, **tabs** uses the shell variable **$TERM**. If the terminal type specified by **$TERM** is not supported, tabs will not work property. If no **workstation** can be found, **tabs** tries a general value that works for most work stations.

+mnum Moves all tabs to the right **num** columns, and makes column **num**+1 the left margin. If **m** is given without a value, 10 is assumed. The leftmost margin on most work stations is defined by **m0**.

Related Information

See the following commands: "nroff, troff" in topic 1.1.301.

See the discussion of **term** and **environ** in *AIX Operating System Technical Reference*.

1.1.460 tail

Purpose
Writes a file to standard output, beginning at a specified point.

Syntax

```
        +-------- -10 --------------------+   +------+   +--------+
tail ---¦ one of                         +---¦      +---¦        +---¦
        ¦ +---+   +- 10 --+   +--- l ---+ ¦   +- -f -+   +- file -+
        +-¦ + +---¦¦      +---¦¦ one of +-+
          ¦ - ¦   +- num -+   ¦  +---+  ¦
          +---+               +--¦ l +--+
                                 ¦ b ¦
                                 ¦ c ¦
                                 +---+
```

```
-----------------
¦ Do not put a blank between these items.
```

Description
The **tail** command writes the named **file** (standard input by default) to standard output, beginning at a point you specify. It begins reading at **+[num**] lines from the beginning of **file** or **-[num**] lines from the end of **file**. The default **num** is 10. **num** is counted in units of lines, blocks, or characters, according to the subflag appearing after **num** (see the following flags). The block size is 512 bytes.

Flags

-f Does not end after it copies the line of the input file if the input file is not read from a pipe, but enters an endless loop in which it sleeps for a second and then attempts to read and copy further records from the input file. Thus, it can be used to monitor the growth of a file being written by another process.

+[num]l

+[num]b

+[num]c Begins reading **num** lines (**l**, the default), blocks (**b**), or characters (**c**) from the beginning of the input.

-[num]l

-[num]b

-[num]c Begins reading **num** lines (**l**, the default), blocks (**b**), or characters (**c**) from the end of the input.

Examples

1. To display the last 10 lines of a file:

```
      tail  notes
```

2. To specify how far from the end to start:

```
tail  -20  notes
```

This displays the last **20** lines of **notes**.

3. To specify how far from the beginning to start:

```
tail  +200c  notes  |  pg
```

This displays **notes** a page at a time starting with the **200**th character from the beginning.

4. To follow the growth of a file:

```
tail  -1  -f  accounts
```

This displays the last line of **accounts**. Once a second, **tail** displays any lines that have been added to the file. This continues until stopped by pressing INTERRUPT (**Ctrl-C**).

Related Information

See the following commands: "dd" in topic 1.1.114 and "pg" in topic 1.1.315.

1.1.461 tapechk

Purpose
Performs consistency checking of the streaming tape device.

Syntax

```
                +- ? ---------------+
/etc/tapechk ---¦        +--------+ +---¦
                +- num1 -¦        +-+
                         +- num2 -+
```

Note: This command does not have MBCS support.

Description
The **tapechk** command performs rudimentary consistency checking on an attached streaming tape device. Some hardware malfunctions with a streaming tape drive can be detected by simply reading a tape. **tapechk** provides a way to perform tape reads on the file level.

Since the streaming tape drive cannot backspace over physical data blocks or files, **tapechk** rewinds the tape to its starting position prior to each check. You can specify numeric arguments to control the number of files checked or skipped. If you do not specify any arguments, **tapechk** rewinds the tape and checks only the first physical block.

Although you can use **tapechk** on any streaming tape cartridge, it is primarily designed for checking tapes written by the **backup** command.

Note: This command is for the PS/2 only.

Flags

num1	Checks data for the next **num1** files.
num2	Skips the next **num2** files from the beginning of the tape.
?	Explains the format of the **tapechk** command.

Note: If you specify this argument, it must be the first argument. The ? must be enclosed in quotes so that it is not treated as a special character.

1.1.462 tar

Purpose
Manipulates archives.

Syntax

```
       one of
       +----+  +------ -b20 ------+  +-----------------+  +------------------
tar ---¦ -c +--¦ +--------------+ +--¦     one of      +--¦        one of
       ¦ -r ¦  +-¦ -v -w -d -h  +-+  ¦ +-------------+ ¦  ¦ +----------------
       ¦ -t ¦    ¦ -m -l -      ¦¦   +-¦ -f file     +-+  +-¦ -S blocksb
       ¦ -u ¦   ¦¦ -b blocks    ¦¦     ¦ -f file-num ¦      ¦ -S feet
       ¦ -x ¦   ¦¦ - num        ¦¦     +-------------+      ¦ -S feet @ densi
       +----+   ¦¦ -L inputlist ¦¦                          +----------------
                ¦¦ -N blocks    ¦¦
                ¦¦ -s -I        ¦¦
                ¦¦ -o           ¦¦
                ¦¦ -p           ¦¦
                ¦¦ -B           ¦¦
                ¦¦ -F           ¦¦
                ¦+--------------+¦
                +----------------+
```

Description
The **tar** command writes files to or retrieves files from an archival storage medium. The **tar** command looks for archives on the default device (usually tape), unless you specify another device with the **-f** flag. File names must not be longer than 100 characters and must not contain blanks. Characters following the first blank are ignored.

Pathname/filename combinations must not be longer than 256 characters. No single directory component may be longer than 155 characters.

When writing to an archive, **tar** uses a temporary file (**/tmp/tar***) and maintains in memory a table of files with several links. You receive an error message if **tar** cannot create the temporary file, or if there is not enough memory available to hold the link tables.

Notes:

1. Some older versions of **tar** may incorrectly process archives containing pathname/filename combinations longer than 100 characters.

2. When the storage device is an ordinary file or a block special file, **-u** and **-r** flags backspace. However, raw magnetic tape devices do not support backspacing. So when the storage device is a raw magnetic tape, the **-u** and **-r** flags rewind the tape, open it, and then read it again.

3. Records are one block long on block magnetic tape, but they are typically less than half as dense on raw magnetic tape. As a result although a blocked raw tape must be read twice, the total amount of tape motion is less than it is when reading one-block records from a block magnetic tape once.

4. The structure of a streaming tape device does not support the addition of information at the end of a tape. Consequently when the storage device is a streaming tape, the **-u** and **-r** flags are not valid options.

An attempt to use these flags results in the error message **tar: Update and Replace options not valid for a streaming tape drive**.

5. There is no way to ask for any occurrence of a file other than the last.

6. There is no recovery from tape errors.

7. The file storage (**fstore**) value for replicated files is not saved in the **tar** archive. Files created by **tar** always have an **fstore** value of 0.

8. The device site number (**rdevsite**) is not maintained for block special and character special files in a **tar** archive. Extracted special files are created using the site on which the extracting **tar** is executing as the **rdevsite** number.

9. If no **-S** flag is used and tape devices are used, **tar** assumes tape density and length so that 73,840 blocks are written to the first tape. The system then prompts for a new tape.

10. The **-u** and **-r** options are not supported for internal tape backup units. The **-b** option must specify an even-number argument. Use the **-f** option to specify the PS/2 internal tape backup unit. For example:

```
tar -cvf/dev/rst0
```

Flags

You must supply one of the following five function flags to control the actions of **tar**:

-c Creates a new archive and writes the **file** at the beginning of the archive.

-r Writes the **file** at the end of the archive. Since the structure of a streaming tape device does not support the addition of information at the end of a tape, this option is not a valid flag when the archival storage device is a streaming tape.

-t Lists the files in the order in which they appear in the archive. Files may appear more than once.

-u Adds **file** to the end of the archive only if it is not in the archive already or if it has been modified since it was written to the archive. Since the structure of a streaming tape device does not support the addition of information at the end of a tape, this is not a valid flag when the archival storage device is a streaming tape.

-x Extracts **file** from the archive. If you specify a **directory**, **tar** extracts all files in that directory from the archive. If you do not specify a **file** or a **directory**, **tar** extracts all of the files from the archive. When an archive contains multiple copies of the same file, **tar** extracts only the last one and overwrites all earlier ones. If you have superuser authority (see "su" in topic 1.1.449), **tar** creates all files and directories with the same user and group IDs as on the tape. If you do not have superuser authority, the files and directories have your user and group IDs.

The other optional flags to **tar** are listed below. In all cases, a directory parameter refers to all the files and subdirectories, recursively, within that directory. Flags without corresponding parameters may appear separately or be grouped together. Flags that take parameters may have them adjacent to the flag letter or as the entire following argument.

-bblocks
: Specifies the number of 512-byte blocks per record. The default is 20, which is appropriate for tape records. Due to the size of inter-record gaps, tapes written with large blocking factors can hold much more data than tapes with only one block per record.

  The block size is determined automatically when tapes are read (function flags **-x** or **-t**). When archives are updated with the **-u** and **-r** functions, the existing record size is used. The **tar** command writes archives using the specified **blocks** value only when creating new archives with the **-c** flag.

-B
: Forces input and output blocking to the number of blocks specified with the **-b** option or to 20 block per record if **-b** is not specified. Certain devices, such as disk devices, write data in record sizes independent of the blocking factor. On such a device, the blocking factor cannot be determined when the device is later read. On input, the blocking factor is the default blocking factor unless an explicit blocking factor is specified.

-C
: If a filename is preceded by **-C,** then **tar** performs a **chdir**(2) to that filename. This allows multiple directories not related by a close common parent to be archived using short relative path names. For example, to archive files from /usr/include and /etc, one might use the following command:

```
tar c -C /usr include -C / etc
```

-d
: Makes separate entries for directories, blocks and character special files, and FIFOs. Normally, **tar** writes only ordinary files to an archive, and extracts only ordinary files and the directories required to contain them as determined by the path names in the archive. When writing to an archive with the **-d** flag, **tar** makes it possible to preserve the directory permission codes and to restore empty directories, special files, and FIFOs with the **-x** flag.

  Note: Although anyone can archive special files, only a user with superuser authority can extract them from an archive.

-ffile[-*num*]
: Uses **file** as the archive to be read or written. When this flag is not specified, **tar** uses a

system-dependent default file name of the form **/dev/rmt?**. If the **file** specified is **-** (minus), **tar** writes to standard output or reads from standard input. If you write to standard output, the **-c** flag must be used. See the Examples section below for more information.

If you specify *num*, **tar** provides automatic spillover from one archive storage unit to another. This feature allows the operator of a system with multiple tape drives to use multitape archives without having to change tapes. For example, **-f/dev/rmt0-2** writes or reads **/dev/rmt0**, followed by **/dev/rmt1**, and then **/dev/rmt2** before requesting that additional volumes be mounted.

-F Checks the file type before archiving. Files or directories named "SCCS" or "RCS" are identified and skipped. Core, error files, filenames ending in ".o" and a.out files is archived.

-h Force tar to follow symbolic links as if they were normal files or directories. Normally, tar does not follow symbolic links.

-i Ignores directory checksum errors.

-I Flattens hidden directories.

Some newer implementations of **tar** may not be able to extract a hidden directory file type. For greater portability, the **-I** flag should be used to archive hidden directory component names which match the machine type on which the **tar** command is executed. For example, hidden directory and component **/bin/foo@/i386** would be extracted as regular file **/bin/foo**.

If the **-I** flag is not used during archive, then on most non-AIX implementations which follow System V, Release 2 or earlier **tar** conventions, an archived hidden directory is extracted as a regular directory with '@' as the last character of the directory pathname. For example, **/bin/foo@/i386** archived under AIX would be extracted as regular directory **'/bin/foo@'** containing file '**i386**'.

-l Writes error messages to standard output if **tar** cannot resolve all of the links to the files archived. When you do not specify this flag, the system does not display these messages.

Linputlist Writes the files listed in the **inputlist** file to archive. The **inputlist** should contain one file name per line. Files from **inputlist** are not treated recursively. If you include the name of a directory in **inputlist**, **tar** does not write that directory's subdirectories to the tape, only that directory's files. If you also list files or directories on the command line, the contents of **inputlist** are included after **tar** has written all the files or the directories and their subdirectories to the archive.

-m	Uses the time of extraction as the modification time. The default is to preserve the modification time of the files.
-Nblocks	Allows **tar** to use very large clusters of blocks when it deals with streaming tape archives. However, on input, **tar** cannot automatically determine the block size of tapes with very long block sizes created with this flag. In the absence of a **-Nblocks** argument, the largest block size that **tar** can automatically determine is 20 blocks.
-o	On output, **tar** normally places information specifying owner and modes of directories in the archive. Former versions of **tar** when encountering this information gives error messages of the form: "<name> / : cannot create"
-p	This modifier says to restore files to their original modes, ignoring the present **umask**(2). Setuid and sticky information is also restored to the superuser.
-s	If **tar** fails in its attempt to link (regular link) two files, with the **-s** option it tries to create a symbolic link instead.
-S blocksb	
-S feet	
-S feet@density	Specifies the number of 512-byte **blocks** per volume (first format), independent of the tape blocking factor. You can also specify the size of the tape in feet by using the second form, and **tar** assumes a default **density**. The third form allows you to specify both tape length and density. Feet are assumed to be 11 inches long to be conservative. This flag lets you deal more easily with multivolume tape archives, where **tar** must be able to determine how many blocks fit on each volume.
-v	Lists the name of each file as it is processed. With the **-t** flag, **-v** gives more information about the tape entries, including file sizes, times of last modification, UID, and GID, and permissions.
-w	Displays the action to be taken followed by the file name, then waits for user confirmation. If the response begins with **y** or **Y**, the action is performed; otherwise, the file is ignored.
-*num*	Uses **/dev/rmt***num* instead of the default. For example, **-2** is the same as **-f/dev/rmt2**. In AIX systems with multidensity tape drives, this flag allows selecting a particular density. The default

unit is system dependent and is chosen to match the default density, as described under the **-S** flag.

Examples

1. To write **file1** and **file2** to a new archive on the default tape drive:

```
tar -c file1 file2
```

2. To extract all files that are in the **/tmp** directory from the archive file on the tape device **/dev/rmt2** and use the time of extraction as the modification time:

```
tar -xm -f/dev/rmt2 /tmp
```

3. To create a new archive file that contains **file1** and pass the archive file to the **dd** command to be written to the device **/dev/rmt1**:

```
tar -cvf - file1 ¦ dd of=/dev/rmt1
```

4. To display the names of the files in the disk archive file **out.tar** on the current directory:

```
tar -vtf out.tar
```

5. To expand the compressed archive file **fil.tar.z**, pass the file to the **tar** command, and extract all files from the expanded archive file:

```
pcat fil.tar.z ¦ tar -xvf -
```

6. To create a tape archive on the default tape drive having the greatest potential of portability to non-AIX systems, use **-I**, **-o** (and **-h** if the target system does not support symbolic links):

```
tar -cvIho ship_directory
```

Files

/dev/rmt?
/tmp/tar*

Compatibility Note

An archive created with a non-AIX **tar** which contains a directory ending with '@' is extracted as a regular directory and follows the TCF conventions for directories ending in '@'. For example, directory **foo@** archived from a non-AIX system is accessible as regular directory **foo** or **foo@** when extracted under AIX.

1.1.463 tbl

Purpose
Formats tables for the **nroff** and **troff** commands.

Syntax

```
       +---------+   +--------+
tbl ---¦ +-----+ +---¦        +---¦
       +-¦ -TX +-+   +- file -+
         +-----+             ¦
                      +------+
```

Note: This command does not have MBCS support.

Description
The **tbl** command is a preprocessor that formats tables for **nroff** and **troff**. The command reads the specified **file** or, if you do not specify any file names or you specify a - (minus) as a file name, it reads standard input. Except for text between lines containing **.TS** and **.TE**, the input is copied unchanged to standard output. Text between **.TS** and **.TE** describes tables and is reformatted by **tbl**. The **.TS** and **.TE** lines are not altered by **tbl**. For more detailed information on how to format text for **tbl**,see *Text Formatting Guide*.

Note: When **tbl** is used with **eqn** or **neqn**, **tbl** should come first to minimize the volume of data passed through pipelines.

The format of the text processed by the **tbl** command can be either of the following:

```
  .TS
  options;
  format.
  data
  .TE
```

or

```
  .TS
  options;
  format.
  data
  .T&
  more format.
  more data
  .TE
```

At the start of **tbl** text, you should include a line containing **.TS**. You can follow this with a line containing global options. The available global options are:

center	Centers the table (the default is left-adjusted).
expand	Makes the table as wide as the current line length.
box	Encloses the table in a box.
doublebox	Encloses the table in a double box.
allbox	Encloses each item of the table in a box.
tab (x)	Uses the character **x** instead of a tab to separate items in a line of input data.

linesize (**n**)	Sets lines and rules for boxes in point size **n**.
delim (**x**,**y**)	Sets **x** and **y** as the **eqn** and **neqn** text delimiters.

End the list of global options with a ; (semicolon).

After the global options, enter lines describing the format of each row in the table. Each format line (except the last) describes one row of the table. The last format line one describes all remaining rows of the table. This must end with a period to indicate that it is the end of the format specification. Each column of the table is described by a single keyletter. The available keyletters are:

c	Centers the item in the column.
r	Right-adjusts the item in the column.
l	Left-adjusts the item in the column.
n	Adjusts the line to \&, if it exists. If the line does not contain \&, adjusts line to the decimal point. If the line does not contain a decimal point but does have digits, adjusts to the last digit. If none of these characters are present, centers the line.
s	Allows the previous item on the left to spill over into this column if the item is too wide for its column.
a	Centers the longest line in this column and then left-adjusts all other lines in it with respect to the centered line.
PS/2	Allows the item above to spill over into this column if the item is too large.

After the keyletter, you can enter specifiers that determine where vertical lines appear between columns, column width, inter-column spacing, and the font and point size of the item. The specifiers which control columns and fonts are as follows:

Specifier	**Meaning**
e or **E**	Equal width columns
f or **F**	Font change
nnn	Column separation
p or **P**	Point size change
t or **T**	Vertical spanning at top
u or **U**	Half line elevation
v or **V**	Vertical spacing change
w or **W**	Minimum width column
z or **Z**	Zero width column
|	Vertical line
||	Double vertical line

The format lines are followed by lines containing data for the table. The last line consists of **.TE**. Within the data lines, data items are separated by tab characters unless the global option **tab** is used. The specifiers which control data entries are as follows:

Specifier	**Meaning**
T{...T}	Text block
\^	Vertical span
_	Short horizontal line
\Rx	Repeat character

The specifiers which control the lines between data entries are as follows:

Specifier	**Meaning**
.xx	Included **troff** request

_	Single line
=	Double line

If a data line consists of only _ (underscore) or = (equal sign), a single or double line is drawn across the table at that point. If an entry in a data line consists of only _ or =, that item is replaced by a single or double line.

Flag

-TX Uses only full vertical line motions, making the output suitable for line printers and other devices that do not have partial vertical line motions.

Related Information

See the following commands: "cw, checkcw" in topic 1.1.108, "eqn, neqn, checkeq" in topic 1.1.152, "mm, checkmm" in topic 1.1.274, "mmt, mant, mvt" in topic 1.1.275, "nroff, troff" in topic 1.1.301, and "troff" in topic 1.1.302.2.

See the **mm** and **mv** miscellaneous facilities in *AIX Operating System Technical Reference*.

See the discussion of **tbl** in *Text Formatting Guide*.

1.1.464 tc

Purpose
Simulates typesetter output for a Tektronix 4014.

Syntax

```
     +------+   +----------------------+   +--------+
tc ---¦      +---¦          +--- P ---+ +---¦        +---¦
     +- -t -+   +- -p num -¦¦ one of +-¦   +- file -+
                ¦          ¦ +-----+ ¦ ¦
                ¦          +-¦ i c +-+ ¦
                ¦            ¦ P p ¦   ¦
                ¦            +-----+   ¦
                +------- -s num -------+
```

```
----------------
¦ Do not put a blank between these items.
```

Description
The **tc** command interprets its input, either a **file** or standard input, as a **troff** document. It then simulates the typesetter output for a Tektronix 4014 work station with ASCII and APL character sets and writes the results to standard output (usually the work station display). The 16 typesetter sizes are mapped into the 4014's four sizes; the entire **troff** character set is drawn using the 4014's character generator, with overstruck combinations where necessary.

At the end of each page, **tc** waits for a new-line character from the keyboard before continuing to the next page. While it is waiting, the command **e** suppresses the screen erase before the next page. !**AIX-cmd** sends **AIX-cmd** to the shell.

There are no font distinctions in the display.

Flags

-pnum letter	Sets page length to **num** and scale to **letter**. **letter** may include the scale factors **p** (points), **i** (inches), **c** (centimeters), and **P** (picas). The default is picas. Do not put a space between **num** and **letter**.
-t	Does not wait between pages (use in a pipeline).
-s num	Number of pages to skip.

Example
To use **tc** in a pipeline with **troff**:

```
  troff  -t  chapter1  |  tc
```

Related Information

See the following commands: "sh, Rsh" in topic 1.1.420 and "troff" in topic 1.1.302.2.

1.1.465 tcopy

Purpose
Copies a magnetic tape.

Syntax

```
tcopy --- src --- dest ---¦
```

Description

The **tcopy** command is designed to copy magnetic tapes. Source and target file names are specified by **src** and **dest**.

Related Information

See the following command: "rmt" in topic 1.1.381.

1.1.466 tctl, mt

Purpose
Gives commands to streaming tape.

Syntax

```
        +--- -f$TAPE ---+             +--- 1 ---+
tctl ---¦               +-- subcmd --¦         +---¦
        +- -f tapename -+             +- count -+
```

Description
The **tctl** command gives subcommands to a streaming tape device. If you do not specify the **-f** flag with **tapename**, the environment variable **TAPE** is used. If the environment variable does not exist, **tctl** uses the device **/dev/rmt4**. The **tapename** parameter must be a raw (not block) tape device. You can specify more than one operation with **count**. The **mt** command is a synonym for the **tctl** command.

Note: To specify a PS/2 internal tape backup unit with the **tctl** command, use the **-f** option and specify the **norewind** device. Superuser authority is required to use the **erase**, **eof**, or **weof tctl** operations. The **ras1** and **ras2** subcommands are not supported on the internal tape backup unit. The **erase** subcommand erases only the first 29K of user data and filemark map. The following example rewinds a PS/2 internal tape backup unit:

```
tctl -f /dev/rst4 rewind
```

Subcommands

eof	Writes **count** end-of-file markers at the current position on the tape.
weof	Writes **count** end-of-file markers at the current position on the tape.
fsf	Moves the tape forward **count** files.
fsr	Moves the tape forward **count** records.
bsf	Moves the tape backward **count** files if the device has backspace capability.
bsr	Moves the tape backward **count** records if the device has backspace capability.
rewind	Rewinds the tape. The **count** parameter is ignored. **Note:** It is sometimes necessary to issue a **reset** before issuing a **rewind** subcommand.
offline	Places the tape drive off-line. The **count** parameter is ignored.
rewoffl	Places the tape drive off-line. The **count** parameter is ignored.
reset	Makes the tape drive ready for either read or write access.

The count parameter is ignored.

erase	Erases all contents on the tape and rewinds it.
retension	Moves the tape to the beginning, the end, and back to the beginning of the tape. If you have excessive read errors during a restore operation, you should run the **retension** subcommand. If the tape has been exposed to environmental extremes, you should run the **retension** subcommand before the save operation. **Note:** This subcommand is available on a PS/2 only.
ras1	Performs a checksum on the tape drive.
ras2	Checks the capstan speed, verifies the operations of the BOT, EOT, and SAFE sensors, and writes a worst case pattern on the tape and attempts to verify the pattern.
status	Prints status information about the tape unit.

Files

/dev/rmt??	The raw streaming tape interface.

Related Information

See the following command: "dd" in topic 1.1.114.

See the **ioctl** system call and the **tape** and **environ** files in *AIX Operating System Technical Reference*.

1.1.467 tee

Purpose
Displays the output of a program and copies it into a file.

Syntax

```
      +--------+   +--------+
tee ---¦ +----+ +---¦        +---¦
      +-¦ -i +-+   +- file -+
        ¦ -a ¦¦          ¦
       ¦+----+¦     +------+
       +------+
```

Description
The **tee** command reads standard input and writes the output of a program to standard output and copies it into **file** at the same time.

Flags

-a Adds the output to the end of **file** instead of writing over it.

-i Ignores interrupts.

Note: If you specify both flags, each must appear separately on the command line, preceded by a -. (minus).

Examples

1. To view and save the output from a command at the same time:

   ```
   lint  program.c  |  tee  program.lint
   ```

   This displays the standard output of the command **lint program.c** at the work station, and at the same time saves a copy of it in the file **program.lint**. If **program.lint** already exists, it is deleted and replaced.

2. To display and append to a file:

   ```
   lint  program.c  |  tee  -a   program.lint
   ```

   This displays the standard output of **lint program.c** at the work station and at the same time appends a copy of it to the end of **program.lint**. If the file **program.lint** does not exist, it is created.

1.1.468 termdef

Purpose
Queries terminal characteristics.

Syntax

```
           +-- -t --+
termdef ---¦ one of +---¦
           ¦ +----+ ¦
           +-¦ -c +-+
             ¦ -l ¦
             ¦ -t ¦
             +----+
```

Description
The **termdef** command identifies the current display model, the active lines setting, or the current columns setting for the display being used. Since the **$TERM** environment variable does not automatically reflect the display being used, the **termdef** command can be used to set it to the correct value. If you are using a display other than an **ibm8513**, you must explicitly reset **$TERM** variable to access the **terminfo** correctly.

Flags

-c Returns the current column value.

-l Returns the current lines value.

-t Returns the name of the current display (this is the default action).

Example
The **termdef** command queries the operating system as to which display model it is currently running on. To set the **$TERM** environment variable to this same value, the following commands can be issued:

If you are running the Bourne shell **sh**)

```
TERM=`termdef`
export TERM
```

If you are running the C shell **csh**)

```
set term=`termdef`
```

These commands can be placed in your **.profile**, if running under the Borne shell, or in your **.cshrc** if running under the 'c' shell.

Related Information

See the following command: "display" in topic 1.1.130.

See the **terminfo** file and the **hft** special file in *AIX Operating System Technical Reference*.

1.1.469 test

Purpose
Evaluates conditional expressions.

Syntax

test -- **expression** --¦

[-- expression --] --¦

Description
The **test** command evaluates **expression** and, if its value is true, returns a zero (true) exit value; otherwise it returns a nonzero (false) exit value; **test** also returns a nonzero exit value if there are no parameters.

Note: In the second form of the command, that is the one that uses square brackets (**[ ]**), rather than the word **test**, the brackets must be surrounded by blanks.

Functions

All the functions and operators are separate parameters to **test**. The following functions are used to construct **expression**:

-r file	True if **file** exists and has read permission.
-w file	True if **file** exists and has write permission.
-x file	True if **file** exists and has execute permission. If the user has superuser authority, **test -x file** evaluates to true for any file that exists whether to not the execute permission bits are set.
-f file	True if **file** exists and is a regular file.
-d file	True if **file** exists and is a directory.
-h file	True if **file** exists and is a hidden directory.
-l file	True if **file** exists and is a symbolic link.
-L file	True if **file** exists and is a local file. If **file** resides in a replicated file system true only if the local copy is the primary copy.
-c file	True if **file** exists and is a character special file.
-b file	True if **file** exists and is a block special file.
-p file	True if **file** exists and is a named pipe (FIFO).
-u file	True if **file** exists and its set-user-ID bit is set.
-g file	True if **file** exists and its set-group-ID bit is set.
-k file	True if **file** exists and its **sticky bit** is set.
-s file	True if **file** exists and has a size greater than zero.

-t [filedescr]	True if the open file with file descriptor number **filedescr** (1 by default) is associated with a work station device.
-z s1	True if the length of string **s1** is zero.
-n s1	True if the length of the string **s1** is nonzero.
s1 = s2	True if strings **s1** and **s2** are identical.
s1 != s2	True if strings **s1** and **s2** are not identical.
s1	True if **s1** is not the null string.
n1 -eq n2	True if the integers **n1** and **n2** are algebraically equal. Any of the comparisons **-ne**, **-gt**, **-ge**, **-lt**, and **-le** can be used in place of **-eq**.

These functions can be combined with the following operators:

!	Unary negation operator.
-a	Binary AND operator.
-o	Binary OR operator (**-a** has higher precedence than **-o**).
\(expression \)	Parentheses for grouping.

Examples

1. To test whether a file exists and is not empty:

```
if test ! -s "$1"
then
   echo $1 does not exist or is empty.
fi
```

   If the file specified by the first positional parameter to the shell procedure does not exist or is empty, this displays an error message. If **$1** does exist and is not empty, it displays nothing. There must be a space between **-s** and the file name.

   The double quotes around **$1** ensure that the test works properly even if the value of **$1** is the empty string. If the double quotes are omitted and **$1** is the empty string, **test** displays the error message **test: parameter expected**.

2. To do a complex comparison:

```
if [ $# -lt 2  -o  ! -s "$1" ]
then
   exit
fi
```

   If the shell procedure was given fewer than two positional parameters or the file specified by **$1** does not exist, this exits the shell procedure. The special shell variable **$#** represents the number of positional parameters entered on the command line that started this shell procedure. For more details, see "Substitutions" in topic 1.1.420.6.

Related Information

See the following commands: "find" in topic 1.1.165 and "sh, Rsh" in topic 1.1.420.

1.1.470 tic

Purpose
Translates **terminfo** files from source to compiled format.

Syntax

```
       +----------------+
tic ---¦      +-------+ +--- file ---¦
       +- -v -¦¦      +-+          ¦
              +- num -+   +--------+
```

```
-----------------
¦ Do not put a blank between these items.
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The **tic** command translates/compiles **terminfo** files from the source format into the compiled format. **tic** places the results in the directory **/usr/lib/terminfo**. If the environment variable **TERMINFO** is set, the results are placed there instead of in **/usr/lib/terminfo**.

The **tic** command compiles all terminfo descriptions in **file**s. When **tic** finds a **use=** field, it searches first the current file, then the master file, **./terminfo.src**.

The total compiled entries cannot exceed 4096 bytes and the name field cannot exceed 128 bytes.

Flag

-vnum	Writes trace information on the progress of **tic**. **num** is an integer that increases the level of the verbosity.

Files

/usr/lib/terminfo/?/*	Compiled terminal capability data base.

Related Information

See the **curses** subroutine and the **terminfo** file in *AIX Operating System Technical Reference*.

1.1.471 time

Purpose
Times the execution of a command.

Syntax

time -- **command** --¦

Description
The **time** command times the execution of the named **command**. The **time** command writes to standard error the elapsed time of the command, the system time used, and the execution time, in seconds.

Examples

1. To measure the time required to run a program:

   ```
   time  a.out
   ```

   This runs the program **a.out** and writes to the standard error output the amount of real, system, and user time that it uses:

   ```
   real 8.6
   user 7.7
   sys 0.5
   ```

   where **real** is the elapsed clock time in seconds, **user** is the user process time in seconds, and **sys** is the CPU time in seconds.

2. To save a record of the **time** information in a file using the Bourne shell.

   ```
   time  a.out  2> a.time
   ```

Related Information

See the following commands: "csh" in topic 1.1.100 and "timex" in topic 1.1.472.

Note: The **csh** command contains a built-in subcommand named **time**. The command and subcommand do not necessarily work the same way. For information on the subcommand, see the **csh** command.

See the **times** system call in *AIX Operating System Technical Reference*.

1.1.472 timex

Purpose
Times a command and reports process data and system activity.

Syntax

```
         +--------+   +-------------------+
timex ---¦ +----+ +---¦      ¦+-------+   +-- command --¦
         +-¦ -o +-+   +- -p --¦ f h k +---+
           ¦ -s ¦            ¦ m r   ¦ ¦
          ¦+----+¦          ¦ +-------+ ¦
          +------+          +-----------+


-----------------
¦ Do not put a blank between these items.
```

Note: This command does not have MBCS support.

Description
The **timex** command reports, in seconds, the elapsed time, user time, and system execution time for **command** where **command** is a local command. With flags specified, **timex** can list or summarize process accounting data for **command** and all of its children, and report total system activity during the execution interval. Output is written to standard error. The system uses the **/usr/adm/pacct** to select process records associated with **command** and includes background processes having the same user ID, work station ID, and execution time window.

Flags

-o Reports the total number of blocks read or written and total characters transferred by **command** and all its children.

-p Lists process accounting records for **command** and all its children. The number of blocks read or written and the number of characters transferred are always reported. The **f**, **h**, **k**, **m**, **r**, and **t** arguments, defined in the **acctcom** command, modify the other data items reported.

-s Reports total system activity that occurred during the execution of **command**. All the data items listed in the **sar** command are reported.

Related Information

See the following commands: "acctcom" in topic 1.1.8 and "sar" in topic 1.1.407.

1.1.473 tip

Purpose
Connects to a remote system.

Syntax

```
       +------+   +----------+
tip ---¦      +---¦          +--- system-name ---¦
       +- -v -+   +- -speed -+
       +------+   +----------+
tip ---¦      +---¦          +--- phone-number ---¦
       +- -v -+   +- -speed -+
```

Note: This command does not have MBCS support.

Description

The **tip** command establishes a full-duplex connection to another machine.

Typed characters are normally transmitted directly to the remote machine (which does the echoing as well). A tilde appearing as the first character of a line is an escape signal; the following are recognized:

~^D ~.	Drop the connection and exit (you may still be logged in on the remote machine).
~c [**name**]	Change directory to name (no argument implies change to your home directory).
~!	Escape to a shell (exiting the shell returns you to tip).
~>	Copy file from local to remote. The **tip** command prompts for the name of a local file to transmit.
~<	Copy file from remote to local. The **tip** command prompts first for the name of the file to be sent, then for a command to be executed on the remote machine.
~p from [**to**]	Send a file to a remote UNIX host. The put command causes the remote UNIX system to run the command string "cat > **to**", while **tip** sends it the **from** file. If the **to** file isn't specified the **from** file name is used. This command is actually a UNIX-specific version of the ~> command.
~t from [**to**]	Take a file from a remote UNIX host. As in the put command the **to** file defaults to the **from** file name if it isn't specified. The remote host executes the command string "cat 'from';echo ~A" to send the file to **tip**.
~|	Pipe the output from a remote command to a local UNIX process. The command string sent to the local UNIX system is processed by the shell.
~$	Pipe the output from a local UNIX process to the

remote host. The command string sent to the local UNIX system is processed by the shell.

~#	Send a BREAK to the remote system. For systems which don't support the necessary **ioctl** call the break is simulated by a sequence of line speed changes and DEL characters.
~s	Set a variable (see the discussion below).
~^Z	Stop the **tip** command (only available with job control).
~^Y	Stop only the "local side" of **tip** (only available with job control); the "remote side" of **tip**, the side that displays output from the remote host, is left running.
~?	Get a summary of the tilde escapes

The **tip** command uses the file **/etc/remote** to find how to reach a particular system and to find out how it should operate while talking to the system; refer to **remote** for a full description. Each system has a default baud rate with which to establish a connection. If this value is not suitable, the baud rate to be used may be specified on the command line, for example "tip -300 mds".

When **tip** establishes a connection it sends out a connection message to the remote system; the default value, if any, is defined in /etc/remote.

When **tip** prompts for an argument (for example during setup of a file transfer) the line typed may be edited with the standard erase and kill characters. A null line in response to a prompt or interrupt aborts the dialogue and returns you to the remote machine.

The **tip** command guards against multiple users connecting to a remote system by opening modems and terminal lines with exclusive access, and by honoring the locking protocol used by **uucp**.

During file transfers **tip** provides a running count of the number of lines transferred. When using the **~>** and **~<** commands, the **eofread** and **eofwrite** variables are used to recognize end-of-file when reading, and specify end-of-file when writing (see below). File transfers normally depend on tandem mode for flow control. If the remote system does not support tandem mode, **echocheck** may be set to indicate **tip** should synchronize with the remote system on the echo of each transmitted character.

When **tip** must dial a phone number to connect to a system, it prints various messages indicating its actions. **tip** supports the DEC DN-11 and Racal-Vadic 831 auto-call-units; the DEC DF02 and DF03, Ventel 212+, Racal-Vadic 3451, and Bizcomp 1031 and 1032 integral call unit/modems.

Variables

The **tip** command maintains a set of **variables** which control its operation. Some of these variable are read-only to normal users (root is allowed to change anything of interest). Variables may be displayed and set through the "s" escape. The syntax for variables is patterned after **vi** and **mail**. Supplying **all** as an argument to the set command displays all variables readable by the user. Alternatively, the user may request display of a particular variable by attaching a "?" to the end. For example, **escape?** displays the current escape character.

Variables are numeric, string, character, or Boolean values. Boolean variables are set merely by specifying their name; they may be reset by prepending a "!" to the name. Other variable types are set by concatenating an **=** and the value. The entire assignment must not have any blanks in it. A single set command may be used to interrogate as well as set a number of variables. Variables may be initialized at run time by placing set commands (without the **~s** prefix in a file **.tiprc** in one's home directory). The **-v** option causes **tip** to display the sets as they are made. Certain common variables have abbreviations. The following is a list of common variables, their abbreviations, and their default values.

beautify
(bool) Discard unprintable characters when a session is being scripted; abbreviated **be**.

baudrate
(num) The baud rate at which the connection was established; abbreviated **ba**.

dialtimeout
(num) When dialing a phone number, the time (in seconds) to wait for a connection to be established; abbreviated **dial**.

echocheck
(bool) Synchronize with the remote host during file transfer by waiting for the echo of the last character transmitted; default is **off**.

eofread
(str) The set of characters which signify and end-of-tranmission during a ~< file transfer command; abbreviated **eofr**.

eofwrite
(str) The string sent to indicate end-of-transmission during a ~> file transfer command; abbreviated **eofw**.

eol
(str) The set of characters which indicate an end-of-line. **tip** recognizes escape characters only after an end-of-line.

escape
(char) The command prefix (escape) character; abbreviated **es**; default value is **~**.

exceptions
(str) The set of characters which should not be discarded due to the beautification switch; abbreviated **ex**; default value is **/et/en/ef/eb**.

force
(char) The character used to force literal data transmission; abbreviated **fo**; default value is "^P".

framesize
(num) The amount of data (in bytes) to buffer between file system writes when receiving files; abbreviated **fr**.

host

(str) The name of the host to which you are connected; abbreviated **ho**.

prompt
(char) The character which indicates an end-of-line on the remote host; abbreviated **pr**; default value is "\n". This value is used to synchronize during data transfers. The count of lines transferred during a file transfer command is based on receipt of this character.

raise
(bool) Uppercase mapping mode; abbreviated **ra**; default value is **off**. When this mode is enabled, all lowercase letters are mapped to uppercase by **tip** for transmission to the remote machine.

raisechar
(char) The input character used to toggle uppercase mapping mode; abbreviated **rc**; default value is "^A".

record
(str) The name of the file in which a session script is recorded; abbreviated **rec**; default value is "tip.record".

script
(bool) Session scripting mode; abbreviated **sc**; default is **off**. When **script** is **true**, **tip** records everything transmitted by the remote machine in the script record file specified in **record**. If the **beautify** switch is on, only printable ASCII characters are included in the script file (those characters between 040 and 0177). The variable **exceptions** is used to indicate characters which are an exception to the normal beautification rules.

tabexpand
(bool) Expand tabs to spaces during file transfers; abbreviated **tab**; default value is **false**. Each tab is expanded to 8 spaces.

verbose
(bool) Verbose mode; abbreviated **verb**; default is **true**. When verbose mode is enabled, **tip** prints messages while dialing, shows the current number of lines transferred during a file transfer operations, and more.

SHELL
(str) The name of the shell to use for the ~! command; default value is **/bin/sh**, or taken from the environment.

HOME
(str) The home directory to use for the ~c command; default value is taken from the environment.

Files

/etc/remote	Global system descriptions.
/etc/phones	Global phone number data base.
$REMOTE	Private system descriptions.
$PHONES	Private phone numbers.
$HOME/.tiprc	Initialization file.
	Lock file to avoid conflicts with **uucp**.

Related Information

See the following commands: "cu" in topic 1.1.105 and "connect" in topic 1.1.90.

1.1.474 tlog

Purpose
Starts and stops sending of I/O data.

Syntax

```
             +- on --+
/etc/tlog ---¦       +---¦
             +- off -+
```

Description
The **tlog** command stops and restarts the sending of terminal I/O data to the message queue of the terminal-logging daemon, **tlogger**. The **tlog off** command stops I/O from going to the daemon. The **tlog on** command restarts the sending of the I/O.

The **tlog** command uses the **TCLOG ioctl ()** system call to stop and restart the flow of I/O to the daemon.

Files

/etc/tlogger	Daemon writes terminal data to a log file.

Related Information

See the following commands: "tlogger" in topic 1.1.475.

1.1.475 tlogger

Purpose
Writes data to a log file.

Syntax

```
                +- -c /usr/adm/ras/tlogfile -+
/etc/tlogger ---¦                            +---
                +--------- -c file ----------+

    +- -b /usr/adm/ras/tlogfile.bk -+
 ---¦                               +--- & ---¦
    +----------- -b file -----------+
```

Description
The terminal-logging daemon **tlogger** collects data read or written to its associated terminal and writes that data to a log file. Each time the daemon is started, the contents of the current log file replace the backup log file. A new current log file is created with permissions set to allow read and write by the owner.

The associated terminal is identified in the following manner: Standard error is assumed to be the correct terminal if it is a terminal device (**isatty()** returns true). Otherwise, the process's **usrinfo** is used to identify the login terminal, and that device is used.

The **tlogger** daemon creates a message queue, and passes that queue id to the associated terminal using the **ioctl TCLOG** system call. The daemon then loops waiting on message queue data; it writes any message queue data it receives to the end of the current log file. The terminal log daemon catches all signals (except **SIGKILL**). On receipt of a signal, the daemon issues an **ioctl TCLOG** to its associated terminal to turn off logging. This causes the terminal to stop sending log messages. The daemon then removes the message queue and exits. The daemon also terminates if it can no longer write to the log file due to file size constraints. In this case, an error message is written to standard error.

The **tlogger** daemon should be started in the background either from **/etc/rc**, or from the command line. This starts the terminal sending its I/O data to the daemon. The **tlog** command can then be used to stop or restart the sending of terminal I/O to the daemon. The daemon itself may be terminated with the **kill** command, but would ordinarily continue to run until shutdown occurs.

Notes:

1. **SIGKILL** should not be used to stop the daemon, since cleanup of system resources cannot be done in that case.

2. It may be necessary to prevent passwords from showing up in the terminal logs. You can prevent the system from logging passwords by having the **getpass()** subroutine turn off terminal logging while it is reading the password. The **login**, **adduser**, **newgrp**, and **passwd** commands use this subroutine.

Flags

-b file	Specifies a file to be used as the backup log file.

	The default backup file is **/usr/adm/ras/tlogfile.bk**.
-c file	Specifies a file to be used as the current log file. The default current file is **/usr/adm/ras/tlogfile**.

Files

/etc/rc	System startup file.
/usr/adm/ras/tlogfile	Default current log file.
/usr/adm/ras/tlogfile.bk	Default backup log file.

Related Information

See the following commands: "shutdown" in topic 1.1.425 and "kill" in topic 1.1.221.

See the **ioctl** system call and the **getpass** subroutine in *AIX Operating System Technical Reference*.

1.1.476 touch

Purpose
Updates the access and modification times of a file.

Syntax

```
         +- -a -m -+   +------------------+   +- directory -+
touch ---¦ +-----+ +---¦              ¦   +---¦              +---¦
         +-¦ -a  +-+   ¦ +-- mmddhhmm --+ ¦   +--- file ----+ ¦
           ¦ -   ¦¦    +-¦              +-+ +-----------------+
          ¦¦ -f  ¦¦      +- mmddhhmmyy -+
          ¦¦ -m  ¦¦
          ¦+-----+¦
          +-------+
```

¦ The current year is the default year.

Description
The **touch** command updates the access and modification times of each **file** or **directory** named to the one specified on the command line. If you do not specify a time, **touch** uses the current time. If you specify a file that does not exist, **touch** creates a file with that name unless you request otherwise with the **-c** flag.

To touch an existing file, you must be the owner of the file or have write permission to the file.

The environment variables **LANG** and **LC_TIME**, if defined, specify the order of month and day in the date specification and of hour and minute in the time specification. Otherwise, these orders default to **mmdd** and **hhmm**.

The return code from **touch** is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

Flags

-a Changes only the access time.

-c Does not create the file if it does not already exist.

-f Attempts to force the touch in spite of write permissions on a file. This option is unnecessary in AIX but is provided for compatibility with other systems.

-m Changes only the modification time.

Examples

1. To update the access and modification times of a file:

   ```
   touch  program.c
   ```

   This sets the last access and last modification times of **program.c** to the current date and time. If **program.c** does not exist, **touch** creates an empty file with that name.

2. To avoid creating a new file:

```
touch  -c  program.c
```

3. To update only the modification time:

```
touch  -m  *.o
```

   This updates only the last modification times of the files in the current directory that end with **.o**. **touch** is often used in this way to alter the results of the **make** command.

4. To explicitly set the access and modification times:

```
touch  -c  02171425  program.c
```

   This sets the access and modification dates to 14:25 (2:25 p.m.) February 17 of the current year.

Related Information

See the following command: "date" in topic 1.1.110.

See the **utime** system call in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.477 tplot

Purpose
Produces plotting instruction for a particular work station.

Syntax

```
         +----- -T$TERM ------+   +--------+
tplot ---¦ +----------------+ +---¦        +---¦
         +-¦ -T workstation +-+   +- file -+
           +----------------+
```

Description
The **tplot** command reads plotting instructions from standard input or from **file**, if specified. (For more information about plotting instructions, see the **plot** file format *AIX Operating System Technical Reference*). **tplot** writes instructions suitable for the specified **work station** to standard output. If **work station** is not specified, the environment variable **TERM** is used. (For more information about environment variables, see the **environ** file in *AIX Operating System Technical Reference*.)

Flags

-Twork station	Uses the plotting instructions for **work station**. The known **work station** is:
	lp 4014 Tek 4014 terminal

Files

/usr/lib/tdumb

Related Information

See the following commands: "graph" in topic 1.1.191 and "splp" in topic 1.1.439.

See the **plot** subroutine and the **plot** file in *AIX Operating System Technical Reference*.

1.1.478 tput

Purpose

Queries the **terminfo** file.

Syntax

```
        +- -T$TERM -+   +-------+   +-----------+
tput ---¦           +---¦       +---¦           +---¦
        +- -Ttype --+   +- cmd -+   +- capname -+
```

Description

The **tput** command uses the **terminfo** file to make terminal-dependent information available to the shell. The output of the **tput** command is a string if the attribute **capname** (for capability name) is of type string or an integer if the attribute is of type integer. If the attribute is of type Boolean, the **tput** command simply sets the exit value (0 for TRUE, 1 for FALSE), and produces no other output.

Flags

-Ttype	Indicates the type of work station. Normally, the value of **type** is supplied by the environment variable **$TERM**.
capname	Indicates the attribute from the **terminfo** file. For more information, see the **terminfo** file in *AIX Operating System Technical Reference*.

Examples

1. To echo the clear-screen sequence for the current work station:

   ```
   tput  clear
   ```

2. To display the number of columns for the current work station:

   ```
   tput  cols
   ```

3. To display the number of columns for the 450 work station:

   ```
   tput  -T450  cols
   ```

4. To set the shell variable **bold** to the highlight mode sequence for the current work station:

   ```
   bold=`tput  smso`
   ```

   This might be followed by a prompt:

   ```
   echo  "${bold}Please  type  in  your  name:  \c"
   ```

5. To set the exit value to indicate if the current work station is a hardcopy terminal:

   ```
   tput  hc
   ```

Files

/usr/lib/terminfo/?/*	Terminal descriptor files.

/usr/include/term.h **/usr/include/curses.h**	Definition files.

Related Information

See the following command: "stty, STTY" in topic 1.1.447.

See the **terminfo** file in *AIX Operating System Technical Reference*.

1.1.479 tr

Purpose
Translates characters.

Syntax

```
           +--------+   +-----------+   +-----------+
/usg/tr ---¦ +----+ +---¦           +---¦           +---¦
           +-¦ -A +-+   +- string1 -+   +- string2 -+
               ¦ -c ¦
            ¦¦ -d ¦¦
            ¦¦ -s ¦¦
            ¦+----+¦
            +------+
```

```
               +--------+   +-----------+   +-----------+
/usr/ucb/tr ---¦ +----+ +---¦           +---¦           +---¦
               +-¦ -A +-+   +- string1 -+   +- string2 -+
                   ¦ -c ¦
                ¦¦ -d ¦¦
                ¦¦ -s ¦¦
                ¦+----+¦
                +------+
```

Note: This command does not have MBCS support.

Description
The **tr** command copies characters from the standard input to the standard output with substitution or deletion of selected characters. Input characters from **string1** are replaced with the corresponding characters in **string2**. **tr** cannot handle an ASCII NUL (\000) in **string1** or **string2**; it always deletes NUL from the input.

Note: AIX is shipped with **/usr/bin/tr** linked to **/usg/tr**. Unless this link is modified, entering **tr** executes **/usg/tr** (that is, you do not have to use the full path name).

In the AIX version abbreviations that can be used to introduce ranges of characters or repeated characters are:

[a-z] Stands for a string of characters whose ASCII codes run from character **a** to character **z**, inclusive.

[a*num] Stands for **num** repetitions of **a**. The value of **num** is considered to be in decimal unless the first digit of **num** is **0**, in which case it is considered to be in octal. When **num** is 0 or is omitted, it is treated as a number large enough to pad **string2** with **a**.

In the BSD version, the abbreviation that can be used to introduce ranges of characters is:

a-z Stands for a string of characters whose ASCII codes run from character **a** to character **z** inclusive. Brackets are not special characters and the hyphen (-) is a special character.

Use the escape character \ (back slash) to remove special meaning from any character in a string. Use the \ followed by 1, 2, or 3 octal digits for the ASCII code of a character.

Flags

-A Translates on a byte-by-byte basis. When you specify this flag, **tr** does not support extended characters.

-c Complements (inverts) the set of characters in **string1** with respect to the universe of characters whose ASCII codes are 001 through 377 octal, if you specify **-A**, and all non-null characters, if you do not specify **-A**.

-d Deletes all input characters in **string1**.

-s Changes characters that are repeated output characters in **string2** into single characters.

Examples

1. To translate braces into parentheses:

   ```
   /usg/tr  '{}'  '()'  <textfile  >newfile
   ```

   This translates each "**{**" to "**(**" and each "**}**" to "**)**". All other characters remain unchanged.

2. To translate lowercase characters to uppercase:

   ```
   /usg/tr  '[a-z]'  '[A-Z]'  <textfile  >newfile
   ```

   or

   ```
   /usr/ucb/tr  a-z  A-Z  <textfile  >newfile
   ```

3. In the AIX version, this is what happens if the strings are not the same length:

   ```
   /usg/tr '[0-9]'  '#'  <textfile  >newfile
   ```

   This translates each "**0**" to a # (number sign).

   Note: If the two character strings are not the same length, the extra characters in the longer one are ignored.

4. In the BSD version, if the strings are not the same length:

   ```
   /usr/ucb/tr  0-9  '#'  <textfile  >newfile
   ```

   This translates each digit to a #.

   Note: If **string2** is too short, it is padded to the length of **string1** by duplicating its last character.

5. In the AIX version, to translate each digit to a #:

   ```
   /usg/tr '[0-9]'  '[#*]'   <textfile  >newfile
   ```

   The * tells **tr** to repeat the # enough times to make the second string as long as the first one.

6. To translate each string of digits to a single #:

```
/usg/tr  -s  '[0-9]'  '[#*]'   <textfile  >newfile
```

   or, with the BSD version:

```
/usr/ucb/tr  -s  0-9  '#'   <textfile  >newfile
```

7. To translate all ASCII characters that are **not** specified:

```
/usg/tr -c '[ -~]' '[A-_]?' <textfile >newfile
```

   This translates each nonprinting ASCII character to the corresponding control key letter (**"\001"** translates to **"A"**, **"\002"** to **"B"**, etc.). ASCII DEL (**"\177"**), the character that follows **"~"** (tilde), translates to ?.

   Or, with the BSD version:

```
/usr/ucb/tr -c ' -~' 'A-_?' <textfile >newfile
```

8. To create a list of the words in a file:

```
/usg/tr  -cs  '[a-z][A-Z]'  '[\012*]'    <textfile  >newfile
```

   or, with the BSD version:

```
/usr/ucb/tr  -cs 'a-zA-Z'  '\012'  <textfile  >newfile
```

Related Information

See the following commands: "ed, red" in topic 1.1.147 and "sh, Rsh" in topic 1.1.420.

See the **ascii** file in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.480 trace

Purpose
Starts the trace function.

Syntax

```
        +------------+   +- /etc/trcprofile -+
trace ---¦ +---------+ +---¦                   +---¦
        +-¦ -o name +-+   +----- profile -----+
          +---------+¦
         +-----------+
```

Description
The **trace** command starts the trace function in the background. This trace function provides a base for debugging the system. The **trace** command monitors the occurrence of selected events in the system and records on disk important data specific to each of these events. You can format this output with the **trcrpt** command.

Any user or program that needs the trace process enabled for debugging or error determination can start the **trace** command. When starting **trace**, you must provide a **profile**. This allows you to tailor the output of the trace session to individual needs. The default **profile** is **/etc/trcprofile**.

There may be more than one trace profile in the file system at a time. The trace profile contains the classes of events that you can select to trace, listed by event class and by a descriptive label. See "Example" for a sample profile. You may keep different profiles to trace different combinations of event classes. The **trace** command also takes additional information about the trace session from the configuration file **/etc/rasconf** (see *AIX Operating System Technical Reference* for a discussion of this file). You set the name and size of the output file in this configuration file.

In a multiuser environment, the **trace** command records all system events, not just events at one virtual terminal.

Flags

-o name	Specifies the name of the log file into which the **trace** demon stores the trace data.

Example

```
****************************************************************************
* SYSTEM TRACE PROFILE
****************************************************************************
* To set trace on for an event class, remove the comment mark (*) from the
* first column of the line containing the event you wish to trace.
* Add a comment mark (*) in the first column of lines containing event types
* you wish to stop tracing.

***** Event
*     Type    Description

*****         Applications
```

```
*****         AIX Extensions
*       36      Config

*****         AIX System Calls
*       60      Shared Memory
*       61      Messages
*       62      Semaphores
*       63      Signals
*       64      Time
*       65      File system
*       66      File Handling
*       67      Directory Handling
*       68      Process
```

Files

/etc/trcprofile	Default profile.
/usr/adm/ras/trcfile	Output file defined in **/etc/rasconf**.
/etc/rasconf	Configuration file.

Related Information

See the following commands: "trcstop" in topic 1.1.482 and "trcrpt" in topic 1.1.481.

See the **rasconf** configuration file in *AIX Operating System Technical Reference*.

See the discussion of **trace** in *AIX Operating System Programming Tools and Interfaces*.

1.1.481 trcrpt

Purpose
Formats a report from the trace log file.

Syntax

```
         +-------------+   +- /usr/adm/ras/trcfile -+
trcrpt ---¦ +---------+ +---¦                        +---¦
         +-¦ -s date +-+   +--------- file ---------+
           ¦ -e date ¦¦              ¦
          ¦+---------+¦           +------+
          +-----------+
```

Note: This command does not have MBCS support.

Description
The **trcrpt** command writes to standard output a chronological listing in readable format of the trace log **file** or **file**s specified. You can specify a maximum of 10 log files. If you do not specify any files, **trcrpt** reads **/etc/rasconf** for a file name. This name is usually **/usr/adm/ras/trcfile**.

Flags

-e date	Ends the report time with entries on or before **date**. The format of **date** is the same as the **date** command, **MMddhhmmyy**.
-s date	Starts the report with entries on or later than **date**. The format of **date** is the same as the **date** command, **MMddhhmmyy**. If you do not specify this flag, **trcrpt** formats the entire log file.

Example

To format a trace log file:

```
  trcrpt -s0109100384 -e0109100584 /u/dave/trc_log  |  print
```

This formats the log file **/u/dave/trc_log**, starting with entries from January 09, 1984 at 10:03 and ending at 10:05. It pipes the formatted output to the print queue.

Files

/usr/adm/ras/trcfile	Default log file.
/etc/trcfmt	Trace format file.
/usr/adm/ras/.trcevents	Trace event types table.

Related Information

See the following commands: "trace" in topic 1.1.480 and "trcstop" in topic 1.1.482.

See the **rasconf** file in *AIX Operating System Technical Reference*.

See the discussion of **trcrpt** *AIX Operating System Programming Tools and Interfaces*.

1.1.482 trcstop

Purpose
Stops the trace function.

Syntax

```
trcstop ---¦
```

Description
The **trcstop** command sends a Software Terminate signal to the **trace** background process. This gracefully ends the **trace** command and forces cleanup.

Files
/tmp/trc_PIDs

Related Information

See the following commands: "trace" in topic 1.1.480 and "trcrpt" in topic 1.1.481.

See the discussion of **trcstop** in *AIX Operating System Programming Tools and Interfaces*.

1.1.483 trcupdate

Purpose
Updates trace format templates.

Syntax

```
                    +------+
trcupdate -- file --¦      +---¦
                    +- -o -+
```

Description

The **trcupdate** command adds, replaces, or deletes trace report format templates in the files **/etc/trcfmt** and **/usr/adm/ras/.trcevents** and event types in the file **/etc/trcprofile**. **trcupdate** creates three undo files in the current directory named **file.undo.trc**, **.trcevents.undo.evt**, and **file.undo.pro**. These undo files can be used as input to **trcupdate** with the **-o** (override) flag to undo the changes the **trcupdate** command has just made.

The **trcupdate** command reads three files named **file.trc**, **file.evt**, and **file.pro**. The **trc** file contains trace format templates; the **evt** file contains trace event types and their corresponding hook IDs; the **pro** file contains the event type line for the trace profile.

The first field of each template contains an operator:

+ To add or replace a template

- To delete a template.

If the operation is +, the following fields contain the template to be replaced. The hook ID of the template is also added to the **/usr/adm/ras/.trcevents** file, and the event type line is added to the trace profile **/etc/trcprofile**. If the operation is a -, the second field contains the hook ID of the template to delete. That hook ID is also deleted in **/usr/adm/ras/.trcevents**, and the event type line is deleted from **/etc/trcprofile**.

When adding or replacing, the **trcupdate** command compares the version numbers of each input template with the version number of the existing template of the same hook IDs. If the version number of the input template is later, it replaces the old template with the input template. If the template does not already exist, it is added to the file. The input file **must** contain the identifier *** /etc/trcfmt** on the first line.

The **file.evt** file contains a table of trace system event types and hook IDs that fall under these types. **trcupdate** reads in the file **/usr/adm/ras/.trcevents** and adds in any hook IDs from **file.evt** that are not already accounted for or reassigns/deletes hook IDs to the event type given in the update file. The first line of the event/hook update file **must** be: *** /ras/.trcevents** or **trcupdate** rejects the input file.

The **file.pro** contains the lines that are to be added to or deleted from **/etc/trcprofile**. **trcupdate** reads **/etc/trcprofile** and adds or deletes the specified event type line from **file.pro**. The first line of the event type file **must** be: *** /etc/trcprofile** or **trcupdate** rejects the input file.

Flags

-o Does no version number checking.

Examples

1. The following is a sample **trc** file:

```
* /etc/trcfmt
+ 355 1.0 new_fmt
- 351
- 352
```

2. The following is a sample **evt** file:

```
* ras/.trcevents
350 355 356 357
```

Files

/etc/trcfmt
/usr/adm/ras/.trcevents
file.evt
file.undo.evt
file.trc
file.undo.trc
file.pro
file.undo.pro

Related Information

See the following command: "trcrpt" in topic 1.1.481.

1.1.484 true, false, i386, u370, i370, xa370

Purpose
Returns an exit value of zero (true) or non-zero (false).

Syntax

```
true ---¦         i370 ---¦

false ---¦        xa370 ---¦

i386 ---¦

u370 ---¦
```

Description
The **true** command returns a zero exit value. The **false** command returns a nonzero value. These commands are usually used in input to the **sh** command.

The following commands are links to either true or false depending on the machine type on which they are executed.

i386	The **i386** command returns true if executed on an AIX PS/2 machine and false if executed on an AIX/370 machine.
u370	The **u370** command returns true if executed on an AIX/370 machine and false if executed on an AIX PS/2 machine.
i370	True, if executed on a non-XA AIX/370 machine, false otherwise.
xa370	True, if executed on a XA AIX/370 machine, false otherwise.

Example

To construct an infinite loop in a shell procedure:

```
while true
do
   date
   sleep 60
done
```

This shell procedure displays the date and time once a minute. To stop it, press INTERRUPT (**Ctrl-C**).

Related Information

See the following command: "sh, Rsh" in topic 1.1.420.

1.1.485 tset, reset

Purpose
Initializes terminal for new user.

Syntax

```
         +-------------+   +--------+   +------------+
tset ----¦ +---------+ +---¦ one of +---¦            +---
reset    +-¦ -ec  -I +-+   ¦ +----+ ¦   +- -m ident -+
           ¦ -Rc  -Q ¦¦    +-¦ -b ¦-+
          ¦¦ -ic  -S ¦¦      ¦ -s ¦
          ¦¦ -       ¦¦      +----+
          ¦¦ r       ¦¦
          ¦¦ N       ¦¦
          ¦+---------+¦
          +-----------+

    +-------------------------------+
 ---¦                  +---------+ +---¦
    +- tset baudrate --¦         +-+
                       +- :type -+¦
                      +-----------+
```

Description
The **tset** command sets up your terminal when you first log in to an AIX system. It does terminal dependent processing such as setting erase and kill characters, setting or resetting delays, sending any sequences needed to properly initialized the terminal, and the like. It first determines the **type** of terminal involved, and then does necessary initializations and mode settings. The type of terminal attached to each AIX port is specified in the **/etc/ports** data base. Type names for terminals may be found in the **terminfo** data base. If a port is not wired permanently to a specific terminal (not hardwired), it is given an appropriate generic identifier such as **dialup**.

In the case where no arguments are specified, the **tset** command simply reads the terminal type out of the environment variable **TERM** and re-initializes the terminal. The rest of this manual concerns itself with mode and environment initialization, typically done once at login, and options used at initialization time to determine the terminal type and set up terminal modes.

When used in a startup script (**.profile** for **sh** users or **.login** for **csh** users) it is desirable to give information about the type of terminal you usually use on ports which are not hardwired. These ports are identified in **/etc/ports** as **dialup** or **plugboard** or **arpanet**, etc. To specify what terminal type you usually use on these ports, the **-m** (map) option flag is followed by the appropriate port type identifier, an optional baud rate specification, and the terminal type. (The effect is to "map" from some conditions to a terminal type, that is, to tell **tset** "If I'm on this kind of port, guess that I'm on that kind of terminal".) If more than one mapping is specified, the first applicable mapping prevails. A missing port type identifier matches all identifiers. Any of the alternate generic names given in **terminfo** may be used for the identifier.

A **baudrate** is specified as with **stty**, and is compared with the speed of the diagnostic output (which should be the control terminal). The baud rate **test** may be any combination of: **>**, @, **<**, and **!**; @ means "at" and **!**

inverts the sense of the test. To avoid problems with metacharacters, it is best to place the entire argument to **-m** within " ' " characters; users of **csh** must also put a "\" before any "!" used here.

Thus:

```
  tset -m 'dialup>300:adm3a' -m dialup:dw2 -m 'plugboard:?adm3a'
```

causes the terminal type to be set to an **adm3a** if the port in use is a dialup at a speed greater than 300 baud; to a **dw2** if the port is (otherwise) a dialup (for example at 300 baud or less).

If the **type** finally determined by the **tset** command begins with a question mark, the user is asked if s/he really wants that type. A null response means to use that type; otherwise, another type can be entered which is used instead. Thus, in the above case, the user is queried on a plugboard port as to whether they are actually using an **adm3a**.

If no mapping applies and a final **type** option, not preceded by a **-m**, is given on the command line then that type is used; otherwise, the type found in the **/etc/ports** data base is taken to be the terminal type. This should always be the case for hardwired ports.

It is usually desirable to return the terminal type, as finally determined by **tset**, and information about the terminal's capabilities to a shell's environment. This can be done using the **-** option; using the Bourne shell, **sh**:

```
  TERM= `tset - options...`; export TERM
```

or using the C shell, **csh**:

```
  setenv TERM `tset - options...`
```

With **csh** it is preferable to use the following command in your .login file to initialize the **TERM** environment variable.

```
  eval `tset -s options... `
```

It is also convenient to make an alias in your .cshrc:

```
  alias tset 'eval `/usr/ucb/tset -s\!*`'
```

This allows the command **tset 2621** to be invoked at any time to set the terminal and environment. **Note to C Shell users:** It is **not** possible to get this aliasing effect with a shell script, because shell scripts cannot set the environment of their parent.

These commands cause **tset** to place the name of your terminal in the variable TERM in the environment; see **environ**.

Once the terminal type is known, **tset** engages in terminal driver mode setting. This normally involves sending an initialization sequence to the terminal, setting the single character erase (and optionally the line-kill (full line erase)) characters, and setting special character delays. Tab and newline expansion are turned off during transmission of the terminal initialization sequence.

On terminals that can backspace but not overstrike (such as a CRT), and when the erase character is the default erase character ('#' on standard systems), the erase character is changed to BACKSPACE (Ctrl-H).

The options are:

-b Prints the sequence of **sh** commands which initialize the environment variable **TERM** based on the type of terminal decided upon.

-ec Set the erase character to be the named character **c** on all terminals, the default being the backspace character on the terminal, usually Ctrl-H. The character **c** can either be typed directly, or entered using the caret notation used here.

-kc Similar to **-e** but sets the line kill character rather than the erase character; **c** defaults to Ctrl-X. The caret notation can also be used for this option.

-ic Similar to **-e** but sets the interrupt character rather than the erase character; **c** defaults to Ctrl-C. The caret notation can also be used for this option.

- Outputs name of the terminal type to the standard output. This is intended to be captured by the shell and placed in the environment variable **TERM**.

-r Reports to standard error the type of the terminal decided upon.

-s Prints the sequence of **csh** commands which initialize the environment variable **TERM** based on the type of the terminal decided upon. For example, if **tset -s** is entered on an hft, the output is:

```
set noglob;
setenv TERM hft;
unset noglob;
```

-m ident Maps the specified terminal type to the port type given. The map option must be followed by a port type identifier, optional baud rate and terminal type.

-n On systems with the Berkeley 4.3 BSD tty driver, specifies that the new tty driver modes should be initialized for this terminal. For a CRT, the CRTERASE and CRTKILL modes are set only if the baud rate is 1200 or greater. See "tty" in topic 1.1.488 for more detail.

-I Suppresses transmitting terminal initialization strings (for example, strings used to set special character delays), which configure the hardware terminal.

-Q Suppresses printing the "Erase set to" "Kill set to" and "Interrupt set to" messages. These messages are usually written to standard error when **-s**, **-S** or **-r** is used.

-S Writes to standard output the type of the terminal decided upon.

If **tset** is invoked as **reset**, it sets cooked and echo modes, turns off cbreak and raw modes, turns on newline translation, and restores special characters to a sensible state before any terminal dependent processing is done. Any special character that is found to be NULL or "-1" is reset to its default value. All arguments to **tset** may be used with reset.

Reset is most useful after a program dies leaving a terminal in a funny state. You may have to type **<LF>reset<LF>** to get it to work since **<CR>** may not work in this state. Often none of this echoes.

Examples

These examples all assume the Bourne shell and use the - option. If you use **csh**, use one of the variations described above. A typical use of **tset** in a .profile or .login also uses the **-e** and **-k** options, and often the **-n** or **-Q** options as well. These options have not been included here to keep the examples small.

1. At the moment, you are on a 2621. This is suitable for typing by hand but not for a .profile, unless you are **always** on a 2621.

   ```
   TERM=`tset - 2621`; export TERM
   ```

2. You have an h19 at home which you dial up on, but your office terminal is hardwired and known in /etc/ports.

   ```
   TERM=`tset - -m dialup:h19`; export TERM
   ```

3. You have a switch which connects everything to everything, making it nearly impossible to key on what port you are coming in on. You use a vt100 in your office at 9600 baud, and dial up to switch ports at 1200 baud from home on a 2621. Sometimes you use someone else's terminal at work, so you want it to ask you to make sure what terminal type you have at high speeds, but at 1200 baud you are always on a 2621. Note the placement of the question mark, and the quotes to protect the greater than and question mark from interpretation by the shell.

   ```
   TERM=`tset - -m 'switch>1200:?vt100' -m 'switch<=1200:2621'; export TER
   ```

4. All of the above entries fall back on the terminal type specified in **/etc/ports** if none of the conditions hold. The following entry is appropriate if you always dial up, always at the same baud rate, on many different kinds of terminals. Your most common terminal is an adm3a. It always asks you what kind of terminal you are on, defaulting to adm3a.

   ```
   TERM=`tset - ?adm3a`; export TERM
   ```

5. If the file **/etc/ports** is not properly installed and you want to key entirely on the baud rate, the following can be used:

   ```
   TERM=`tset - -m '>1200:vt100' 2621`; export TERM
   ```

Files

/etc/ports	Port name to terminal type mapping data base.
/usr/lib/terminfo/?/*	Terminal capability data base.

Related Information

See the following commands: "csh" in topic 1.1.100, "sh, Rsh" in topic 1.1.420, and "stty, STTY" in topic 1.1.447.

See **terminfo** and **environ** in the *AIX Operating System Technical Reference.*

Compatibility Note
4.3 ports and terminal information are now in **/etc/ports** and terminfo data base (instead of **/etc/ttys** and **termcap**).

1.1.486 tsort

Purpose
Sorts an unordered list of ordered pairs (a topological sort).

Syntax

```
        +--------+
tsort ---¦        +---¦
        +- file -+
```

Description
The **tsort** command reads from **file** or standard input an unordered list of ordered pairs, it builds a completely ordered list, and writes it to standard output.

The input **file** should contain pairs of non-empty strings separated by blanks. Pairs of different items indicate a relative order. Pairs of identical items indicate presence, but no relative order. You can use **tsort** to sort the output of the **lorder** command.

If **file** contains an odd number of fields, **tsort** writes the error message **Odd data**.

Example

To create a subroutine library:

```
  lorder  charin.o  scanfld.o  scan.o  scanln.o
  |  tsort  |  xargs  ar  qv  libsubs.a
```

This creates a subroutine library named **libsubs.a** that contains **charin.o**, **scanfld.o**, **scan.o**, and **scanln.o**. The ordering of the object modules in the library is important. The **ld** command requires each module to precede all the other modules that it calls or references. The **lorder** and **tsort** commands together add the subroutines to the library in the proper order.

Suppose that **scan.o** calls **scanfld.o** and **scanln.o**. **scanfld.o** also calls **charin.o**. First, the **lorder** command creates a list of pairs that shows these dependencies:

```
  charin.o charin.o
  scanfld.o scanfld.o
  scan.o scan.o
  scanln.o scanln.o
  scanfld.o charin.o
  scan.o scanfld.o
  scan.o scanln.o
```

Next, the **|** (vertical bar) sends this list to the **tsort** command, which converts it into the ordering we need:

```
  scan.o
  scanfld.o
  scanln.o
  charin.o
```

Each module precedes the module it calls. **charin.o**, which does not call another module, is last.

The second **|** then sends this list to **xargs**, which constructs and runs the following **ar** command:

```
  ar qv libsubs.a scan.o scanfld.o scanln.o charin.o
```

This **ar** command creates the properly ordered library.

Related Information

See the following commands: "ar" in topic 1.1.23, "lorder" in topic 1.1.245, and "xargs" in topic 1.1.544.

1.1.487 ttt

Purpose
Plays tic-tac-toe.

Syntax

```
                  +--------+
/usr/games/ttt ---¦ +----+ +---¦
                  +-¦ -e +-+
                    ¦ -   ¦¦
                   ¦+----+¦
                   +-------+
```

Description
The **ttt** game plays the popular X and O game. This is a learning version, but it learns slowly. It loses nearly 80 games before completely mastering the game.

Flags

-e Increases the speed of the learning.

-i Displays the instructions prior to the start of the game.

To quit the game, press INTERRUPT (**refer to keyboard definition**) or END OF FILE (**Ctrl-D**).

Files

/usr/games/ttt.a Learning file.

1.1.488 tty

Purpose
Writes to standard output the full path name of your work station.

Syntax

```
       +------+
tty ---¦      +---¦
       +- -s -+
```

Description
The **tty** command writes the name of your work station to standard output.

Flag

-s Suppresses reporting the path name. The exit value has the following possible meanings:

0	Standard input is a work station.
1	Standard input is not a work station.
2	Invalid flags specified.

If your standard input is not a work station and you do not specify the **-s** flag, you get the message **not a tty**.

Examples

1. To display full path name of your work station:

```
tty
```

2. To test whether the standard input is a work station:

```
if tty -s
then
   echo 'Enter the text to print:' >/dev/tty
fi
print
```

If the standard input is a work station, this displays the message **Enter the text to print:** as a prompt and prints the text that the user types. If the standard input is not a work station, this displays nothing. It merely prints the text read from the standard input.

The **echo...>/dev/tty** displays the prompt on the screen even if you redirect the standard output of the shell procedure. This way the prompt is never written into an output file. The special file **/dev/tty** always refers to your work station, although it also has another name like **/dev/console** or **/dev/tty2**.

1.1.489 ul

Purpose
Does underlining.

Syntax

```
      +-----------------+   +--------+
ul ---¦ +-------------+ +---¦        +---¦
      +-¦ -i          +-+   +- file -+ ¦
        ¦ -t terminal ¦¦  +------------+
       ¦+-------------+¦
       +---------------+
```

Note: This command does not have MBCS support.

Description
The **ul** command reads the named files (or standard input if none are given) and translates occurrences of underscores to the sequence which indicates underlining for the terminal in use, as specified by the environment variable **TERM**. The **-t** option overrides the terminal kind specified in the environment. The terminal description file in the directory **/usr/lib/terminfo** is read to determine the appropriate sequences for underlining. If the terminal is incapable of underlining, but is capable of a standout mode then that is used instead. If the terminal can overstrike, or handles underlining automatically, **ul** behaves like **cat**. If the terminal cannot underline, underlining is ignored.

Flags

-tterminal Overrides the terminal type specified in the environment.

Note: When using **ul -t** with a non-existent terminal type, the default terminal specified in the environment will be used.

-i Causes the **ul** command to indicate underlining by a separate line containing appropriate dashes '**-**'; this is useful when you want to look at the underlining which is present in an **nroll** output stream on a crt-terminal.

Related Information

See the following commands: "man" in topic 1.1.258, "nroff, troff" in topic 1.1.301, and "colcrt" in topic 1.1.78.

1.1.490 umask

Purpose
Displays and sets file-creation permission code mask.

Syntax

```
        +-------+
umask ---¦       +---¦
        +- nnn -+
```

Description
The **umask** command sets your file-creation mask to **nnn**, three octal digits that represent the read/write/execute permissions for owner, group, and others, respectively. When you create a file the system ANDs the complement of **nnn** to 777 for directories and 666 for files, in effect removing the corresponding permissions. (See "chmod" in topic 1.1.67 for more information on file and directory permission codes.)

If you do not specify **nnn**, **umask** displays the current value of your file-creation permission code mask. The initial system mask (set in **/etc/profile**) is **022**.

The **umask** command is actually a built-in subcommand of the **sh** and **csh** commands.

Examples

1. To display the current file creation mask:

    ```
    umask
    ```

2. To prevent other people from writing to the directories or files you create in the future:

    ```
    umask 022
    ```

    This sets the file creation mask to **022**, which takes away write permission for group members and others. Directories are created with the permission code **755**. Files are created with **644**.

3. To prevent other people from using your files:

    ```
    umask 077
    ```

    This sets the file creation mask to **077**, which removes read, write, and execute permission for group members and others. Now files are created with permission code **600**.

Related Information

See the following commands: "chmod" in topic 1.1.67, "csh" in topic 1.1.100, and "sh, Rsh" in topic 1.1.420.

See the **creat**, **chmod**, **mknod**, **open**, and **umask** calls in *AIX Operating System Technical Reference*.

See the discussion of file permissions in *Using the AIX Operating System*.

See the discussion of tailoring the user environment in *Managing the AIX Operating System*.

1.1.491 umount, unmount

Purpose
Unmounts a previously mounted file system.

Syntax

```
                                 one of
                                 +-----+
                                 ¦ all ¦
                           +-----¦ -a  +-----+
                           ¦     +-----+     ¦
  one of                   ¦     one of      ¦
+---------+   +---------+  ¦  +-----------+  ¦
¦ umount  +---¦ +----+ +---+--¦ device    ¦  +---¦
¦ unmount ¦   +-¦ -f +-+   ¦  ¦ directory ¦  ¦
+---------+     ¦ -  ¦¦    ¦  +-----------+  ¦
               ¦+----+¦    ¦     one of      ¦
               +------+    ¦ +-------------+ ¦
                           +-¦ -s device   +-+
                             ¦ -t type     ¦
                             ¦ -h hostname¦¦
                             +-------------+
```

```
-----------------
¦ -h is for NFS only
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **umount** command unmounts a previously mounted file system. Processing on the file system completes and it is unmounted. Only members of the system group and users operating with superuser authority can issue the **umount** command. You can specify the file system as either the **directory** or **device** on which it is mounted.

If TCF is installed, file systems must be unmounted by running the **umount** command on the same site within the cluster when the corresponding **mount** command was run. If a replicated file system is being unmounted, only the local copy of the file system is unmounted. If other file system copies remain mounted on other cluster sites, the file system remains available to processes on all sites throughout the cluster.

Note: You cannot use the **umount** command on a device that is in use unless the **-f** flag is specified. A device is in use if any file is open for any reason or if a user's current directory is on that device.

You can also specify the following parameter:

all	Unmounts all mounted file systems.

Flags

-a	Unmounts all mounted file systems (same as **all**).
-f	The file system is unmounted even if it is in use.

-h hostname	Unmounts all file systems which have been mounted from the specified host using NFS.
-s device	Prohibits the use of the **/etc/mtab** file if it is damaged or not writable. If you use this flag, you must specify the name of the **device** to be unmounted.
-t type	Unmounts all stanzas in **/etc/filesystems** that contain **type = type** and are mounted. (**type** is a string value, such as **remote**.)
-v	Displays a message indicating the file system being mounted.

Examples

(Note: The first two examples do not apply to AIX/370.)

1. To unmount a diskette drive:

   ```
   umount /dev/fd0
   ```

2. To unmount the device mounted on **/diskette0**:

   ```
   umount /diskette0
   ```

3. To unmount files and directories of a specific type:

   ```
   umount -t prod
   ```

   This unmounts all files or directories that have a stanza in the **/etc/filesystems** file that contains the attribute **type = prod**.

Files

/etc/filesystems	Descriptions of mountable file systems.
/etc/mtab	Table of currently mounted file systems.

Related Information

See the following command: "mount" in topic 1.1.278.

See the **mount** and **umount** system calls and the **/etc/mtab** file in *AIX Operating System Technical Reference*.

1.1.492 uname

Purpose
Displays the name of the current operating system.

Syntax

```
         +------- -s -------+
uname ---¦      one of      +---¦
         ¦      +----+      ¦
         ¦ +----¦ -a +----+ ¦
         ¦ ¦    ¦ -x ¦    ¦ ¦
         +-¦    +----+    +-+
           ¦ +----------+ ¦
           +-¦ -m -n -l +-+
             ¦ -r -s -  ¦¦
            ¦+----------+¦
            +------------+
```

Description
The **uname** command writes to the standard output the name of the operating system that you are using.

Note: In AIX, changing a machine name is done with the **chparm** command.

Flags

-a Displays all information specified with the **-m**, **-n**, **-r**, **-s**, and **-v** flags.

-l Displays the TCF cluster site number.

-m Displays the type of hardware running the system.

-n Displays the name of the node (this may be a name that the system is known by to a communications network).

-r Displays the release number of the operating system.

-s Displays the operating system name. (This flag is on by default.)

-v Displays the operating system version.

-x Displays the information specified with the **-a** flag and the LAN network number.

If you enter a flag that is not valid, **uname** exits with an error message, an error return status, and no output.

Example

To display the complete system name and version banner:

```
uname -a
```

Related Information

See the **uname** system call in *AIX Operating System Technical Reference*.

1.1.493 unget

Purpose
Cancels a previous **get** command.

Syntax

```
         +-----------+
unget ---¦ +-------+ +-- file --¦
         +-¦ -n    +-+        ¦
           ¦ -rSI  ¦¦  +------+
          ¦¦ -s    ¦¦
          ¦+-------+¦
          +---------+
```

Description
The **unget** command allows you to restore a g-file created with a **get -e** before the new delta is created, and therefore discarding the changes (see "get" in topic 1.1.186 and "delta" in topic 1.1.117). If you specify a **-** (hyphen) in place of **file**, standard input is read, and each line of standard input is interpreted as the name of a Source Code Control System (SCCS) file. The **unget** command continues to take input until it reaches an end of file character, which is a **Ctrl-D** if input is from the keyboard.

If you specify a directory in place of **file**, the **unget** command performs the requested actions on all SCCS files (those files with the **s.** prefix).

Flags

Each flag or group of flags applies independently to each named file.

-n	Prevents the automatic deletion of the g-file. This flag allows you to retain the edited version of the file without making a delta.
-rSID	Specifies the new delta that would have been created by the next use of the **delta** command. You must use this flag if you have two or more pending deltas to the file under the same login name. You can look at the p-file to see if you have more than one delta pending to a particular SID under the same login name. The **SID** specification must unambiguously specify only one SID to discard, or **unget** displays an error message and stops running.
-s	Suppresses writing the deleted SID to standard output.

Example

To discard the changes you have made to an SCCS file after doing a **get -e**:

```
  unget  s.prog.c
```

Related Information

See the following commands: "delta" in topic 1.1.117, "get" in topic 1.1.186, and "sact" in topic 1.1.404.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and*

Interfaces.

1.1.494 unifdef

Purpose
Removes "ifdef'ed" lines.

Syntax

```
            +--------------+   +--------+
unifdef ---¦ +----------+ +---¦        +---¦
            +-¦ -c idsym +-+   +- file -+
              ¦ -l iusym ¦
             ¦¦ -t       ¦¦
             ¦¦ Dsym     ¦¦
             ¦¦ Usym     ¦¦
             ¦+----------+¦
             +------------+
```

Description

The **unifdef** command is useful for removing ifdef'ed lines from a file while otherwise leaving the file alone. The command **unifdef** is like a stripped-down C preprocessor: it is smart enough to deal with the nested ifdefs, comments, single and double quotes of C syntax so that it can do its job, but it doesn't do any including or interpretation of macros. Neither does it strip out comments, though it recognizes and ignores them. You specify which symbols you want defined **-Dsym** or undefined **-Usym** and the lines inside those ifdefs are copied to the output or removed as appropriate. The ifdef, ifndef, else, and endif lines associated with **sym** are also be removed. Ifdefs involving symbols you don't specify are untouched and copied out along with their associated ifdef, else, and endif lines. If an ifdef X occurs nested inside another ifdef X, then the inside ifdef is treated as if it were an unrecognized symbol. If the same symbol appears in more than one argument, only the first occurrence is significant.

The **-l** option causes **unifdef** to replace removed lines with blank lines instead of deleting them.

If you use ifdefs to delimit non-C lines, such as comments or code which is under construction, then you must tell **unifdef** which symbols are used for that purpose so that it won't try to parse for quotes and comments in those ifdef'ed lines. You specify that you want the lines inside certain ifdefs to be ignored but copied out with **-idsym** and **-iusym** similar to **-Dsym** and **-Usym** above.

If you want to use the **unifdef** command for plain text (not C code), use the **-t** option. This makes **unifdef** refrain from attempting to recognize comments and single and double quotes.

The **unifdef** command copies its output to **stdout** and takes its input from **stdin** if no **file** argument is given. If the **-c** argument is specified, then the operation of **unifdef** is complemented, for example the lines that would have been removed or blanked are retained and vice versa.

Related Information

See the following command: "diff" in topic 1.1.124.

1.1.495 uniq

Purpose
Deletes repeated lines in a file.

Syntax

```
       +--------+   +----------+   +-------------------------+
uniq ---¦ one of +---¦ +------+ +---¦           +-----------+ +---¦
       ¦ +----+ ¦   +-¦ +num +-+   +- infile --¦           +-+
       +-¦ -c +-+     ¦ -nu  ¦¦              +- outfile -+
         ¦ -d ¦      ¦+------+¦
         ¦ -u ¦      +--------+
         +----+
```

Description

The **uniq** command reads standard input or **infile**, compares adjacent lines, removes the second and succeeding occurrences of a line, and writes to standard output or the specified file **outfile**. **infile** and **outfile** should always be different files. Repeated lines must be on consecutive lines in order to be found. You can arrange them with the **sort** command (see page 1.1.432) before processing.

Flags

-c Precedes each output line with a count of the number of times each line appears in the file. This flag supersedes **-d** and **-u**.

-d Displays only the repeated lines.

-u Displays only the non-repeated lines.

-num Skips over the first **num** fields. A field is a string of nonspace, nontab characters separated by tabs and or spaces from adjacent data on the same line.

+num Skips over the first **num** characters. Fields specified by **num** are skipped before characters.

Related Information

See the following commands: "comm" in topic 1.1.83 and "sort" in topic 1.1.432.

1.1.496 units

Purpose
Converts units in one measure to equivalent units in another.

Syntax

units ---¦

Description
The **units** command converts quantities expressed in one measurement to their equivalents in another. The **units** command is an interactive command. It prompts you for the unit you want to convert from and the unit you want to convert to (see "Examples"). This command only does multiplicative scale changes. That is, it can convert from one value to another only when the conversion is done with a multiplication factor. For example, it can not convert between degrees Fahrenheit and degrees Celsius, because 32 must be added or subtracted in the conversion.

You can specify a quantity as a multiplicative combination of units, optionally preceded by a numeric multiplier.

Indicate powers by suffixed positive integers and division by / (slash).

The **units** command recognizes **lb** as a unit of mass, but considers **pound** to be the British pound sterling. Compound names are run together (such as **lightyear**). Prefix British units differing from their American counterparts with **br** (**brgallon** for instance). The file **/usr/lib/unittab** contains a complete list of the units that the **units** command uses.

Most familiar units, abbreviations, and metric prefixes are recognized, together with the following:

pi	Ratio of circumference to diameter
c	Speed of light
e	Charge on an electron
g	Acceleration due to gravity
force	Same as **g**
mole	Avogadro's number
water	Pressure head per unit height of water
au	Astronomical unit.

Examples

To start the **units** command, enter:

```
  units
```

Now you can try the following examples. In these examples, the text that you enter is shown in **bold type** and the output from **units** is shown in **non-bold type**.

1. To display conversion factors:

```
      you have: in
      you want: cm
              * 2.540000e+00
              / 3.937008e-01
```

The output from **units** tells you to multiply the number of inches by **2.540000e+00** to get centimeters, and to divide the number of inches by **3.937008e-01** to get centimeters.

These numbers are in standard exponential notation, so **3.937008e-01** means **3.937008 ¦ 10(-1)**, which is the same as **0.3937008**. The second number is always the reciprocal of the first. That is, **2.54 = 1** ÷ **0.3937008**.

2. To convert a measurement to different units:

```
you have: 5 years
you want: microsec
        * 1.577846e+14
        / 6.337753e-15
```

   The output shows that **5 years** equals **1.577846¦10(14)** microseconds, and that one microsecond equals **6.337753¦10(-15)** 5-year periods.

3. To give fractions in measurements:

```
you have: 1|3 mi
you want: km
        * 5.364480e-01
        / 1.864114e+00
```

   The | (vertical bar) indicates division, so **1|3** means one-third. This shows that one-third mile is the same as **0.536448** kilometers.

4. To include exponents in measurements:

```
you have: 1.2-5 gal
you want: floz
          * 1.536000e-03
          / 6.510417e+02
```

   The expression **1.2-5 gal** stands for **1.2¦10(-5)**. Do **not** type an **e** before the exponent. This example shows that **1.2¦10(-5)** (**0.000012**) gallons equal **1.536¦10(-3)** (**0.001536**) fluid ounces.

5. To specify complex units:

```
you have: gram centimeter/second2
you want: kg-m/sec2
          * 1.000000e-05
          / 1.000000e+05
```

   The units **gram centimeter/second2** mean "grams ¦ centimeters ÷ second(2)." Similarly, **kg-m/sec2** means "kilograms ¦ meters ÷ sec(2)," which is often read as "kilogram-meters per seconds squared." You can show multiplication of units with a **-** (hyphen) or with a blank.

6. If the units you specify after "**you have**" and "**you want**" are incompatible:

```
you have: ft
you want: lb
conformability
          3.048000e-01 m
          4.535924e-01 kg
```

The message **conformability** means that the units you specified cannot be converted. Feet measure length, and pounds measure mass, so converting from one to the other doesn't make sense. Therefore, the **units** command displays the equivalent of each value in standard units.

In other words, this example shows that one foot equals **0.3048** meters and that one pound equals **0.4535924** kilograms. **units** shows the equivalents in meters and kilograms because the command considers these units to be "standard" measures of length and mass.

Files

/usr/lib/unittab	List of units used by the **units** command.

1.1.497 update

Purpose
Periodically updates the super block.

Syntax

/usr/ucb/update

Description

This command is superseded by the **cron** command but provided for compatibility.

The **update** command is a program that runs the **sync** command every 30 seconds.

Related Information

See the following commands: "sync" in topic 1.1.453, "init, telinit" in topic 1.1.208, and "cron" in topic 1.1.97.

See in the *AIX Technical Reference Vol I*, the **sync** system call.

1.1.498 updatedb

Purpose
Update the file list used by the **find** command.

Syntax

/usr/lib/find/updatedb ---¦

Description
The **updatedb** command is a script used to create or recreate the file **/usr/lib/find/find.codes**. This file, when created, contains a list of every file in the system which is located in a publicly readable directory.

The file **/usr/lib/find/find.codes** provides users with a quick way to locate files using **find**. See the **find** command on page 1.1.165 for more information on using this file.

The **updatedb** command is usually run once each week by the root user by the following entry in the file **/usr/adm/weekly**:

```
  if [ -L /usr/lib/find/find.codes ]; then
               echo ""
               echo "Rebuilding find database:"
               su -f restrict -c "/usr/lib/find/updatedb"
  fi
```

This runs the **updatedb** command as the user **restrict** so that only publicly-readable files and directories can be searched.

If the Transparent Computing Facility is installed, this script is only run on one site in the cluster, the site which stores the writable copy of **/usr/lib/find/find.codes**.

Related Information
In this book: "find" in topic 1.1.165, "cron" in topic 1.1.97.

1.1.499 updatep

Purpose
Updates one or more Licensed Program Products (LPPs).

Syntax

```
                  one of
                 +------+
               +-¦ -a   +-+
               ¦ ¦ -ac  ¦ ¦
               ¦ ¦ -ai  ¦ ¦   +---------------+   +-----------------+   +----
           +---¦ ¦ -aci ¦ +---¦ +-----------+ +---¦ +-------------+ +---¦
           ¦   ¦ +------+ ¦   +-¦           +-+   +-¦             +-+   +- -q
           ¦   ¦  +----+  ¦     +- -n user -+       +- -d device -+
           ¦   +--¦ -s +--+
           ¦      +----+
           ¦
           ¦   one of
updatep ---¦   +-----+
           ¦   ¦ -r  ¦
           +---¦ -rx +---¦
           ¦   ¦ -c  ¦
           ¦   +-----+
           ¦
           ¦   one of
           ¦   +-----+                                     +---------------+
           +---¦ -u  +--- lpp.name --- vv.rr.llll.ffff ---¦ +-----------+ +---
               ¦ -ur ¦                +---------------+   +-¦           +-+
               +-----+                      vv = version    +- -n user -+
                                            rr = release
                                          llll = level
                                          ffff = fix
```

Description

Warning: Under no circumstances attempt to run more than one installation or update process at the same time. The primary site must be up when executing **updatep**.

Failure to perform these steps may result in a failure in the installation or update of the active files being serviced.

The **updatep** command controls the update process for one or more **LPP**s. It also lets you determine the status of pending LPP updates and provides documentation about the updates. You must be a operating with superuser authority to run this command.

The **updatep** command supports an apply/commit/reject philosophy. To apply one or more **LPP**s, you use the **-a** or the **-ai** flags. You must then use either the **-c** flag to commit the LPP or the **-r** flag to reject the LPP. Normally you should not use **-r** until you have tested the LPP on your system. If you specify **-ac** or **-aci**, you can apply and commit in one operation. The **-r** flag must be used separately. During an apply, **updatep** normally saves the current versions of files that are being updated. If needed, these files can be used to do a recovery or reject.

You are responsible for reserving update save space in the **/usr** file system. **updatep** checks to insure that there is adequate save space in

/usr before it applies an update. If there is insufficient free space, **updatep** gives you the option of either ending the command or of allowing it to continue. If you end the command, you can take action to increase the free space in your **/usr** file system. If you continue, no current versions of files are saved and **updatep** automatically commits the update, even though you may not have requested a commit originally. Normally, you should reserve 4000 blocks (2 megabytes) of free space in the **/usr** file system for updates.

You cannot use INTERRUPT to stop the **updatep** command. To stop **updatep**, press QUIT WITH DUMP. This should be used only in extreme circumstances since the state of the system cannot be predicted. For example, some update control files may need to be deleted. If the program is interrupted, you must run **/etc/lpp/inuconfig** to reset the status of the system to what it was before **updatep** was issued.

Flags

-a[i] Applies the updates for one or more **LPP**s. If there is a pending update for any LPP on the system, **updatep** does not permit an apply. You must either commit or reject all pending updates before it accepts another update apply.

The **updatep** command asks you to select the LPP you wish to update. After you select a **LPP**, **updatep** runs the **inudocm** command for any specific update instructions. If it finds any, it copies them into the file **/usr/lpp/***lpp_idd***/ui.vv.rr.llll**., where **vv** is the version, **rr** the release, and **llll** the level of the **LPP**. For fixes, this is **/usr/lpp/***lpp_idd***/ui.vv.rr.llll.ffff**, where **ffff** is the fix number. Normally you should review instructions before continuing. To restart the update procedure and ignore the check for existing update instructions, enter **updatep -ai** or **updatep -aci**.

The **updatep** command applies the update for each LPP by running **inuupdt** for each name. After each update, it deletes the **/usr/lpp/***lpp_idd***/inst_updt** directory. It then runs **inudocm** to check for any update documentation. If there are changes to any documents shipped with this update, **updatep** copies it into the file **/usr/lpp/***lpp_idd***/me.vv.rr.llll** and writes a message.

-c Commits a previous update apply. **updatep** presents selection information for **LPP**s that have pending updates. You select the **LPP**s that you want to commit.

Any **LPP**s that you apply as a group must be committed as a group. Management control information about the update changes to indicate that the LPP is committed. **updatep** deletes the directory that contains the update recovery information, **/usr/lpp/***lpp_idd***/inst_updt.save**.

-d device
Specifies the input device name. The default input device is **/dev/rfd0**. If this is a tape device, it must be the rewind device, **/dev/rmt0rh**.

-n user Lets you specify a name in the LPP history file that is responsible for the **LPP**. The default is the value of the system variable **$LOGNAME**. If you specify **user**, the first 8 nonblank characters are stored in the LPP history file.

-q Runs in quiet mode suppressing most of the interactive queries.

-r Rejects a previous update apply for one or more **LPP**s. **updatep** presents selection information for the **LPP**s that have pending updates. You select the **LPP**s to reject.

Any **LPP**s that are grouped together by the system must be rejected or applied as a group. Specify **-r** without **-x** if you want automatic recovery of saved files. If you do specify the **-x** flag, the management control information about the update reflects that the update is rejected, but **updatep** does not recover saved files. To recover the necessary files, look at the information in **/usr/lpp/***lpp_id***/inst_updt.save**. This flag should be used only by someone very knowledgeable about the system.

-s Writes status information about all pending LPP commits.

-u lpp_idd vv.rr.llll.ffff
Uncommit takes a committed update to the **lpp_idd** with version, release, level and fix numbers equal to **vv.rr.llll.ffff** and puts it back in the applied state where it can be either committed or rejected. Uncommit is only permitted if the requested uncommit is for the last committed update on the system. Otherwise, if there have been any updates or **installp(s)** of any LPP since the requested uncommit, a list of **lpps** installed/updated after the requested one is displayed and the user is asked to first **uncommit** them.

-x Cancels the automatic recovery of saved files (use with **-r**).

When the update completes, it is sometimes necessary for the service tools to take special actions, such as build the kernel or reboot. In these cases, a special code must be returned to the installation tools via the installation scripts **update** or **inst_updt.loc.** described in the *AIX Operating System Programming Tools and Interfaces*. Some codes with local implications are returned by **inst_updt.loc** (**for example 2, 3, 4, and 5**). **Codes 5 and 7 are specific to update**. Code 0 and error codes may be returned by either script. This return code signals the end of the update LPP, and determines what actions the installation tools perform next. Use one of the following values for a return code.

Code	**Description**
0	Successful completion. No additional action is needed.
2	Successful completion. The service tools update superblocks and the inode list and flush the buffers (**sync**).
3	Successful completion. The service tools build a new kernel. Then they update superblocks and the inode list and flush the buffers. Then they instruct the user to reboot the system.
4	Successful completion. The service tools build a new kernel. Then they update superblocks and the inode list and flush the buffers before they perform an IPL on the operating system.
5	Update cancelled by the **update** procedure without errors.
6	Successful completion. The service tools update superblocks

and the inode list and flush the buffers (**sync**). Then they instruct the user to reboot the system.

7	The update was cancelled by the **update** procedure with errors. The service tools recover the previous state of the system.

Subtopics

1.1.499.1 Internal Commands

The **updatep** command uses the **inudocm** command for update documentation control. It uses the **inuupdt** command to apply an update to a single program. **inuupdt** runs a program-provided update procedure, **update**. **updatep** passes the following parameters to **update** procedure:

The full path name of the apply-list

The full path name of the device (file) where the update informatio is stored in **backup** format.

In addition to the commands discussed here, program-provided update procedures can use all of the internal commands discussed under "installp" in topic 1.1.212 Since they are internal commands, they do a minimum validation of input parameters. Their purpose is to provide common code for functions frequently needed by most program-provided procedures. Since these internal commands function as subcommands, they return exit values rather than issue error messages. However, messages may come from other system commands that they run. C Language programmers of update procedures that call these commands can use the **/usr/include/inu21.h** file to define the return codes for them.

1.1.499.2 inudocm

The **inudocm** command is normally used as an internal command to get copies of specific update instructions or manual errata information that you can print out. There may be cases, however, when you would enter this command from the command line (for example, if you have misplaced the manual errata information that came with a previous update). You must be a member of the system group to run this command.

The **inudocm** command has the following syntax:

inudocm -cu {**-d device**} {**lpp_idd**} {**level**} {**-f file**}

where **lpp_idd** specifies the name of the program being checked. It must be specified unless you use the **-f** flag. It can be a maximum of 8 characters. **level** specifies the current level of **lpp_idd**. This value must be identical to the level value for the last committed entry in **/usr/lpp/lpp_idd/lpp.hist**. It must be specified unless you use the **-f** flag.

Flags

-d device Restores **file** from this **device**. **device** must be the full path name of a device special file. The default **device** is **/dev/rfd0**. You must not specify this flag if you use the **-f** flag.

-e Requests the existing manual errata information for **lpp_idd** from **level**. If you select this flag, **inudocm** uses the **ar x** command to extract the archive file **/usr/sys/inst_updt/lpp_idd_erata**. (If this file is not present, no information is available.) **inudocm** extracts any level-dependent manual errata information files if there are any more recent than the current level. Selected files are moved to **/usr/lpp/lpp_idd/me.vv.rr.llll**.

-f file Identifies a file that already contains the **lpp_idd** and the **level**. Only **updatep** itself should use this flag.

-u Requests the existing specific update instructions for **lpp_idd** from **level**. If you select this flag, **inudocm** uses the **ar x** command to extract the archive file **/usr/sys/inst_updt/lpp_idd_instr**. (If this file is not present, no information is available.) **inudocm** extracts any level-dependent specific update instruction files if there are any more recent than the current level. Selected files are moved to **/usr/lpp/lpp_idd/ui.vv.rr.llll.**

The **inudocm** command returns the following exit values:

0 Normal return, no error occurred.

1 The system cannot run **inudocm**.

2 Specific update instruction files were requested but not found.

4 Manual errata information was requested, but none were found.

6 Specific update instructions and manual errata were both requested but not found.

201 An invalid flag was specified, or the first argument was not **e, -eu, or -u**.

202 One or more parameters were missing.

204 Too many parameters were entered.

250 The **level** parameter did not contain exactly 4 characters, or they were not numeric.

251 An error occurred while attempting to restore **/usr/sys/inst_updt/control**.

253 The directory **/usr/lpp/lpp_idd** does not exist.

1.1.499.3 inuupdt

The **inuupdt** command provides a common interface for applying an update to a single program. Normally, **updatep** runs **inuupdt**.

The **inuupdt** command has the following format:

inuupdt -d device current-level new-level lpp_idd

where **lpp_idd** is the name of a program and **current-level** specifies the current maintenance level. **new-level** is the level of the update to be applied. **lpp_idd** can be a maximum of 8 characters and **current-level** must be identical to the level value in the **/usr/lpp/lpp_idd/lpp.hist file**.

/etc/lpp/qproc onsite sitenumber /etc/lpp/qproc
Program to trigger the processing of all pending queue entries on an individual site. **qproc** runs on each individual site and processes all pending **updatep** or **installp** actions for that site. **qproc** first compares the local queue pointer to the global queue pointer. If local queue pointer file is not found, it is assumed that the local queue pointer value is null meaning the site is new and all queue entries are to be processed. If local queue pointer is less than global queue pointer, a "local" action is pending. **qproc** processes the entries one by one from the entry beginning with entry after the one pointed to by the local pointer and ending by processing the last queue entry.

/etc/lpp/qinvoke
Program to coordinate when **qproc** is to run on each site of the multisite cluster. On the multisite system, this program uses the command **at** to set up jobs for each site's **qproc** to run.

On the single site systems, **qproc** is run as a part of **installp/updatep** because to time synchronization is required for a one site system. In such cases, **qinvoke** simply calls **qproc**.

Files

See **installp** command for internal command files.

/usr/include/inu21.h	Error code definitions for internal routines.
/usr/lpp/*lpp_idd***/inst_updt**	Temporary directory.
/usr/lpp/*lpp_idd***/inst_updt.save**	Directory for saved files.
/usr/lpp/*lpp_idd***/inst_updt/arp**	Program specific control library.
/usr/lpp/*lpp_idd***/me.vv.rr.llll**	Document change file.
/usr/lpp/*lpp_idd***/ui.vv.rr.llll**	Update instruction file.
/usr/lpp/*lpp_idd***/me.vv.rr.llll.ffff**	Document change file for fixes.
/usr/lpp/*lpp_idd***/ui.vv.rr.llll.ffff**	Update instruction file for fixes.
/usr/sys/inst_updt	Temporary directory.
/usr/sys/inst_updt/control	Update control library.
/usr/sys/inst_updt/inutemp.xx...x	Temporary files.
/usr/sys/inst_updt/*lpp_idd***_erata**	Document change library.
/usr/sys/inst_updt/*lpp_idd***_instr**	Update instruction library.
/usr/sys/inst_updt/updt_cntrl	Temporary file.

Related Information

See the following command: "installp" in topic 1.1.212.

See the **lpp.hist** file in *AIX Operating System Technical Reference*.

See the discussion of updating programs in *AIX Operating System Programming Tools and Interfaces*.

1.1.500 uptime

Purpose

Show how long system has been up.

Syntax

uptime ---¦

Description

The **uptime** command prints the current time, the length of time the system has been up, and the average number of jobs in the run queue over the last 1, 5 and 15 minutes. It is essentially, the first line of a **w** command.

Files

/unix	System name list

Related Information

See the following command: "w" in topic 1.1.528.

1.1.501 users

Purpose
Makes a compact list of **users** who are on the system.

Syntax

```
         +------------+
users ---+---- -L ----+---¦
         +- filename -+
```

Description

The **users** command lists the login names of the **users** on the system in a compact, one-line format. The default (no options) lists the users on each site with the site name prepended. The '-L' option lists the **users** who are on the local system only. An optional utmp-like file can be specified and that file is used instead of the default **utmp** file.

Examples

```
$ users
fafnir:      fred rolf root rorex
viper:
atlas:       joe joe joel rabii root tracy
swords:      diahanne hyland jackv jefff
snap:        root
coins:       marie miriam shane
spica:       bill ckm matt tussingt
polaris:
carmen:      root
merak:       krissie sandy steves sulentic
hermit:      mohammad
electra:     gth oleg
wands:       bonnie dixon eastin ennis lang pvrc rich swift
wheels:
procyon:     steves steves
sabik:       cooper cooper dinhl franks noh
mimosa:
cups:        glen steve suri timabel
izar:

$ users -L
bonnie dixon eastin ennis lang pvrc rich swift

$ users /fafnir/utmp
fred rolf root rorex
```

Files
/sitename/utmp
/etc/utmp

Related Information

See the following command: "who" in topic 1.1.537.

1.1.502 uucheck

Purpose
Checks for files and directories required by BNU.

Syntax

```
           +------+   +-----------------+
uucheck ---¦      +---¦                 +---¦
           +- -v -+   +- -x debug-level -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **uucheck** command checks for the presence of the files and directories required by the Basic Networking Utilities (BNU) facility. The command also checks for errors in the permissions file, **/usr/adm/uucp/Permissions**.

Note: The **uucheck** command does not check file/directory modes or some errors in the permissions file, such as duplicate login or machine names.

When BNU is installed, **uucheck** verifies that the directories, programs, and support files required to operate the networking facility are present. The command is executed automatically, as one of the first steps in the installation process, before the required BNU directories, programs, and files are actually installed.

Note: The **uucheck** command can be issued from the command line. For example, it would be useful to issue **uucheck** after making changes in part of the BNU facility such as the **Permissions** file.

Flags

-v	Gives a detailed explanation of how the BNU programs interpret the permissions file.
-x debug_level	Displays debugging information on the screen of the local terminal. The valid range for **debug_level** is 0 to 9. The higher the number, the more detailed the final report.

Files

/etc/locks/LCK*	Prevent multiple use of device.
/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Maxuuscheds	Limits scheduled jobs.
/usr/adm/uucp/Maxuuxqts	Limits remote command executions.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.
/usr/spool/uucp/*	Spooling directory.
/usr/spool/uucppublic/*	Public directory.

Related Information

See the following commands: "uucico" in topic 1.1.503, "uusched" in topic 1.1.511, "uucp" in topic 1.1.506, "uustat" in topic 1.1.512, and "uux" in topic 1.1.515.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.503 uucico

Purpose
File transport program for the BNU facility.

Syntax

```
          +------------------+   +------------------+   +------------------+
uucico ---¦                  +---¦                  +---¦                  +-
          +- -r role-number -+   +- -x debug-level -+   +- -s system-name -+
```

Note: This command does not have MBCS support.

Description

The **uucico** program transports Basic Networking Utilities (BNU) requests. The BNU commands **uucp** and **uux** both queue jobs that are transferred to the specified computer by **uucico** after the required data, work, or execute files have been created.

The **uucico** program is normally started by the scheduler, **uusched**, but it can be started manually for debugging. The BNU commands **uutry**, **Uutry**, **Nutry**, and **uukick** also start **uucico** with debugging turned on.

The **uucico** program is a BNU daemon (a program executed internally to handle file transfers and command executions). It selects the device used for the communications link, establishes the connection to the remote computer, and performs the required login sequence. The program also performs permission checks, transfers data (**D.***) and command (**C.***) files, logs results, and notifies specified users of transfer requests. The **uucio** program is also the shell for **uucp** call in.

Flags

-r role_number	The role numbers are the number 1 for the server mode and the number 0 for client mode. The default is 0. If **uucico** is started manually, this flag should be set to 1.
-x debug_level	Displays debugging information on the screen of the local terminal. The valid range for **debug_level** is 0 to 9. The higher the number, the more detailed the final report. This flag is useful in correcting problems with the **expect-send** sequence in the **Systems** file.
-s system_name	The name of the remote system. Use only when starting **uucico** manually. The **system_name** is supplied internally when **uucico** is started automatically.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.
/usr/adm/uucp/Maxuuscheds	Limits scheduled jobs.
/usr/adm/uucp/Maxuuxqts	Limits remote command executions.
/usr/adm/uucp/Myname	Contains the cluster's uucp node name.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.

/usr/spool/uucp/*	Spooling directory.
/usr/spool/uucppublic/*	Public directory.

Related Information

See the following commands: "cron" in topic 1.1.97, "uucp" in topic 1.1.506, "uusched" in topic 1.1.511, "uustat" in topic 1.1.512, "uutry, Uutry, uukick, Nutry" in topic 1.1.514, and "uux" in topic 1.1.515.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.504 uucleanup

Purpose
Deletes selected files older than a specified number of days from the BNU spool directory or a named directory.

Syntax

```
                           +-----------+
             +-------------¦           +---------+
             ¦             +- -C time -+         ¦   +-----------+   +---------
uucleanup ---¦                                  +---¦           +---¦
             ¦   +-----------------------------+ ¦   +- -D time -+   +- -X tim
             +---¦            +--------------+ +-+
                 +- -W time --¦              +-+
                              +- -m string -+

    +-------------+   +-------------+
 ---¦             +---¦             +---¦
    +- -s system -+   +-- -T time --+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description
The Basic Networking Utilities (BNU) program **uucleanup** scans the spool directory (**/usr/spool/uucp**) for old files and takes appropriate action to remove them in a useful way. Used primarily by the BNU program administrator, **uucleanup** performs the following tasks:

Informs the requester of send/receive requests for systems that canno be reached
Warns users about requests that have been waiting for a given numbe of days; the default is 1 day
Returns mail that cannot be delivered to the sende
Removes all other files older than a specified number of days from th spool directory.

The **uucleanup** program is started by the shell **uudemon.cleanup**, located in **/usr/adm/uucp**, which in turn is started by the **cron** script, located in **/usr/spool/cron/crontabs/uucp**. In general, **uucleanup** is executed automatically. You can also start the **uucleanup** program manually if you have superuser privileges.

Note: When BNU is installed, automatic cleanup is not enabled. Edit the file **/usr/spool/cron/crontabs/uucp** and remove the comment character **.#** from the beginning of the **uudemon.cleanup** line.

Flags

-Ctime	Removes any **C.*** (command) files as old as, or older than, the number of days specified in **time**, and sends appropriate information to the requester. Unless specified otherwise, the default **time** is 7 days.
-Dtime	Removes any **D.*** (data) files as old as, or older than, the number of days specified in **time**. Also attempts to deliver any remaining mail messages.

	The default **time** is 7 days.
-Ttime	Removes any **T.*** (temporary) files as old as, or older than, the number of days specified in **time**. Also attempts to deliver any remaining mail messages. The default **time** is 7 days.
-Wtime	Sends a mail message to the requester warning that **C.** files as old as, or older than, the number of days specified in **time** are still in the spool directory. The message includes the job ID and, in the case of mail, the mail message. The administrator may use the **-m** option to include a message line telling whom to call to check the problem. The default **time** is 1 day.
-Xtime	Removes any **X.*** (execute) files as old as, or older than, the number of days specified in **time**. The default **time** is 2 days. **Note:** There are probably no related data files. If any related data files remain, however, they are handled by **D.*** processing, as described above.
-mstring	Includes a specified line of text in the warning message generated by the **-Wtime** option. The default line is: **See your local administrator to locate the problem**.
-otime	Removes other files as old as, or older than, the number of days specified in **time**. The default **time** is 2 days.
-ssystem	Executes **uucleanup** only on the spool directory specified by **system** The default is to clean up all BNU spool directories. **Note:** Unless one of the **time** flags is set to a specific number of days, **uucleanup** uses the default **times** values.

Files

/etc/cron	Command that starts **uudemon.cleanup**.
/usr/adm/uucp	Directory with commands used internally by **uucleanup**.
/usr/spool/cron/crontabs/uucp	File that tells **cron** to start uucp-related daemons, including **uudemon.cleanup**.
/usr/spool/uucp	Spooling directory.

Related Information

See the following commands: "cron" in topic 1.1.97, "uucp" in topic 1.1.506, and "uux" in topic 1.1.515.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.505 uucollect

Purpose
Collects files from local spool directories.

Syntax

```
uucollect ---¦
```

Description
The **uucollect** command is used in a TCF cluster to collect files spooled locally on individual cluster sites and put them into the master spool directory for transmission. Usually, **uucollect** is run from the **uucp cron** job just before polling remote sites.

Note: The **uucollect** command collects files from local spool directories only if that local site is included in the **/usr/adm/uucp/Spools**.

Files

/usr/spool/uucp/*	Spooling directory.
/usr/adm/uucp/Spools	Spooling file.

Related Information

See the following commands: "uusched" in topic 1.1.511 and "uustat" in topic 1.1.512.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.506 uucp

Purpose
Copies files from one AIX system to another AIX or Unix system.

Syntax

```
       +-- -c --+   +-- -d --+   +------------------+
uucp ---¦ one of +---¦ one of +---¦ +----------------+ +-- source-files -- de
       ¦ +----+ ¦   ¦ +----+ ¦   +-¦ -g grade       +-+
       +-¦ -c +-+   +-¦ -d +-+     ¦ -              ¦¦
         ¦ -C ¦       ¦ -f ¦      ¦¦ -m             ¦¦
         +----+       +----+      ¦¦ -n user        ¦¦
                                  ¦¦ -r             ¦¦
                                  ¦¦ -s file        ¦¦
                                  ¦¦ -x debug-level ¦¦
                                  ¦+----------------+¦
                                  +------------------+
```

Note: This command does not have MBCS support.

Description

The Basic Networking Utilities (BNU) command **uucp** copies one or more **source** files from one AIX system to one or more **destination** files on another AIX or Unix system.

The **uucp** command accomplishes the file transfer in two steps: first, by creating a command (**C.***) file in the spooling directory on the local computer, and then by sending the request to the specified computer via the **uucico** command.

Command files include information such as the full path name of the source and destination files, the sender's login name, and so on. The full path name of a command file is a form of the following:

/usr/spool/uucp/system_name/C.system_nameNxxx

where **N** is the grade of the request and **xxxx** is the hexadecimal sequence number used by BNU.

Note: If the **uucp** command is used with the **-C** flag to copy the files to the spool directory for transfer, **uucp** creates not only a command file, but also a data (**D.***) file that contains the actual source file. The full path name of a data file is in the following form:

/usr/spool/uucp/system_name/D.system_namexxxx###

Once the command files (and data files, if necessary) are created, **uucp** then calls the **uucico** daemon, which in turn attempts to contact the remote computer to deliver the files.

Note: It is useful to issue the **uuname** command to determine the exact name of the remote system before issuing **uucp**. The **uulog** command provides information about **uucp** activities on a system.

Subtopics

1.1.506.3 Permissions

1.1.506.1 Path Names Used with uucp

Path names for the source and destination of the **uucp** transfer may be one of the following:

a full path nam

a relative path nam

a path name preceded by **~user**, where **user** is a login name on the specified system. The specified user's login directory is then considered the destination of the transfer.

If the user specifies an invalid login name, the files are transferred to the public directory, **/usr/spool/uucppublic**, which is the default.

A path name preceded by **~/destination**, where **destination** is appended to **/usr/spool/uucppublic**.

This destination is treated as a file name unless more than one file is being transferred by this request, or the destination is a directory. To ensure that it is a directory, follow the destination name with a **/** (slash). For example, **~/amy/** as the destination creates the directory **/usr/spool/uucppublic/amy**, if it does not already exist, and puts the requested files in that directory.

1.1.506.2 Source and Destination File Names

A file name may be a path name on the local system, or may have th
following form:

system_name!path_name

where **system_name** is taken from a list of system names that BNU knows about.

The destination **system_name** may also be a list of names, such as the following:

system_name!system_name!...!
system_name!path_name

In this case, an attempt is made to send the file via the specified route to the destination. Make sure that intermediate nodes in this route are willing to forward information (see *Managing the AIX Operating System*).

The shell pattern-matching characters **?**, *****, and **[...]** may be used in the path names; the appropriate system expands them.

Note: The shell pattern-matching characters should not be used in the path name of the destination file.

If the **destination** is a directory rather than a file, **uucp** uses the last part of the **source** name.

1.1.506.3 Permissions

The system administrator should restrict the access to local files b users on other systems.

When transmitting files, **uucp** preserves execute permissions and grants read and write permissions to the owner, the group, and all others. (The **uucp** command owns the file.)

Sending files to arbitrary **destination** path names on other systems, or getting files from arbitrary **source** path names on other systems, often fails because of security restrictions. The files specified in the path name must give read or write permission not only for the same group of users, but also for any group.

Protected files and files in protected directories owned by th requestor can be sent by **uucp**.

Flags

-c Transfers the source files to the destination on the specified computer. The source files are not transferred via the spool directory. This saves the system from copying possibly large files to the spooling directory for transfer. (See the discussion of the **-C** flag.) This flag is on by default.

-C Copies local files to the spool directory for transfer. Depending on the configuration of the **Poll** and **Systems** files, and on how often the **uusched** command is run, the files could be transferred immediately (on demand polling), or in the future.

Note: Occasionally, there are problems in transferring a source file. For example, the remote computer may not be working or the login attempt may fail. In such a case, the file remains in the spool directory until it is either transferred successfully or removed by the **uucleanup** command.

-d Creates any intermediate directories needed to copy the source files to the destination. This flag is on by default.

-f Does not create intermediate directories during the file transfer.

-ggrade Specifies when the files are to be transmitted during a particular connection. **Grade** is a single number (0-9) or letter (A-Z, a-z); lower ASCII-sequence characters cause the files to be transmitted earlier than do higher sequence characters. The number 0 is the highest (earliest) grade; z is the lowest (latest) grade. The default is **N**.

-j Displays the job identification number of the transfer operation on standard output. This job ID can be used by the BNU command **uustat** to obtain the status of a information about the status of a particular job or with **uustat -k** to terminate the transfer before it is completed.

-m	Sends mail to the requester when the transfer to the remote system is completed. The message is sent to the requester's mailbox, **$HOME/.newmail**. The **mail** command does not send a message for a local transfer. **Note:** The **-m** flag works only when sending files or receiving a single file. It does not work when forwarding files. Receiving multiple files specified by the shell pattern-matching characters **?**, *****, and **[...]** does not activate the **-m** option.
-nuser_name	Notifies the user specified by **user_name** on the designated system that files have been sent. The mail system does not send a message for a local transfer.
-r	Prevents the starting of the file transfer program, **uucico**, even if the command was issued at a time when calls to the remote system are permitted. By default, a call to the remote system is attempted if the command is issued during a time period specified in the **Poll** and **Systems** files.
-sfile	Reports the status of the transfer to the specified file. In this case, the **file** designation must be a full path name.
-xdebug_level	Displays debugging information on the screen of the local system. The **debug_level** is a number between 0 and 9. The higher number gives a more detailed report.

Examples

1. To copy one or more files locally, within the same directory:

   ```
   uucp file1 file2
   ```

2. To copy file **f1** from the local system to a remote system named **hera**:

   ```
   uucp /u/geo/f1 hera!/u/geo/f1
   ```

3. To copy file **f2** from the remote system **hera** and place it in the public directory:

   ```
   uucp hera!/u/geo/f2 /usr/spool/uucppublic/f2
   ```

4. To place the **f2** file in a directory other than the public directory:

   ```
   uucp hera!/u/geo/f2 /u/geo/f2
   ```

   In this case, make sure that the **geo** login directory allows write permission to both "other" user and "other" group (for example, with mode 777).

Files

/usr/spool/uucp	Spooling directory.
/usr/spool/uucppublic	Public directory.
/usr/lib/uucp	Contains **uucico** daemon.

Related Information

See the following commands: "mail, Mail" in topic 1.1.253, "uucleanup" in topic 1.1.504, "uulog" in topic 1.1.508, "uuname" in topic 1.1.509, "uusched" in topic 1.1.511, "uustat" in topic 1.1.512, "uux" in topic 1.1.515, and "uuxqt" in topic 1.1.516.

See the information about international character support in *Managing the AIX Operating System*. Also, see the chapter on basic networking utilities.

1.1.507 uuencode, uudecode

Purpose

These commands encode/decode a binary file for transmission via mail.

Syntax

```
            +--------------+
uuencode ---¦              +--- remotedest ---¦
            +- sourcefile -+

            +--------+
uudecode ---¦        +---¦
            +- file -+
```

Description

The **uuencode** command and the **uudecode** command send a binary file via uucp (or other) mail. This combination can be used over indirect mail links even when **uusend** is not available.

The **uuencode** command takes the named source file (default standard input) and produces and encoded version on the standard output. The encoding uses only printing ASCII characters, and includes the mode of the file and the **remotedest** for recreation on the remote system.

The **uudecode** command reads an encoded file, strips off any leading and trailing lines added by mailers, and recreates the original file with the specified mode and name.

The intent is that all mail to the user "decode" should be filtered through the **uudecode** program. This way the file is created automatically without human intervention. This is possible on the uucp network by either using **sendmail** or by making **rmail** be a link to **Mail** instead of **mail**. In each case, an alias must be created in a master file to get the automatic invocation of **uudecode**.

If these facilities are not available, the file can be sent to a user on the remote machine who can uudecode it manually.

The encode file has an ordinary text form and can be edited by any text editor to change the mode or remote name.

The file is expanded by 35% (3 bytes become 4 plus control information) causing it to take longer to transmit.

The user on the remote system who is invoking the **uuencode** command (often **uucp**) must have write permission on the specified files.

Related Information

See the following commands: "uucp" in topic 1.1.506, "uux" in topic 1.1.515, "mail, Mail" in topic 1.1.253.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.508 uulog

Purpose

Provides information about UUCP and **uux** activities on a system.

Syntax

```
        +------+   +-----------+   +----------------+
uulog ---¦      +---¦           +---¦ +- -f system -+ +---¦
        +- -x -+   +- -number -+   +-¦             +-+
                                     +- -s system -+
```

Note: This command does not have MBCS support.

Description

The Basic Networking Utilities (BNU) command **uulog** displays the contents of a log file of **uucico** or **uuxqt** activities. Individual log files are created for each remote system with which the local system communicates using the **uucp**, **uuto**, or **uux** commands.

The log file of **uucico** activities is named **/usr/spool/uucp/.Log/uucico/system**. The log file of **uuxqt** activities is named **/usr/spool/uucp/.Log/uuxqt/system**.

Flags

-fsystem	Performs a "tail **-f**" on the file transfer log for the specified **system**, in this case displaying the end of the log file. Use **INTERRUPT** (**Ctrl-C**) to leave the file and return to the prompt.
-ssystem	Prints information about copy requests involving the specified **system**.
-x	Looks in the **uuxqt** log file for the given system.
-number	Indicates that a **tail** command should be executed for the specified **number** of lines.

Files

/usr/bin	Contains **uulog** command.
/usr/spool/uucp	Spooling directory.
/usr/spool/uucppublic	Public directory.

Related Information

See the following commands: "tail" in topic 1.1.460, "uucp" in topic 1.1.506, "uuname" in topic 1.1.509, and "uux" in topic 1.1.515.

See the information about international character support in *Managing the AIX Operating System*. Also, see the chapter about basic networking utilities.

1.1.509 uuname

Purpose
Provides information about other systems accessible to the local system.

Syntax

```
          +------+
uuname ---¦      +---¦
          +- -l -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The Basic Networking Utilities (BNU) command **uuname** displays a list of all the computers networked to the local system; the list of accessible systems is displayed on the screen of the local terminal.

In order for a local system to communicate with a remote system via BNU, the remote system must:

have a UNIX-based operating syste

be connected to the local system

Note: BNU can be used to communicate between a PS/2 and a non-UNIX based operating system, but such communications may require special hardware or software. The remote systems accessible with BNU commands are identified when the BNU programs are installed, and are listed in **/usr/adm/uucp/Systems**.

Before copying a file to another system with the **uucp** command, issue **uuname** to determine the exact name of the remote system.

Flags

-l Displays the name of the local system. The **uuname -l** command reads the name of the local system from the **/usr/adm/uucp/Myname** file.

Examples

1. To identify the remote systems connected to the local systems:

```
uuname
```

The system responds with a list like the following:

```
hera
zeus
merlin
arthur
```

2. To identify the local system:

```
uuname -l
```

The system responds:

```
venus
```

Files

/usr/spool/uucp	Spooling directory.
/usr/spool/uucppublic	Public directory.
/usr/adm/uucp	Directory containing **Systems** file.

Related Information

See the following commands: "uucp" in topic 1.1.506, "uulog" in topic 1.1.508, and "uux" in topic 1.1.515.

See the chapter on Basic Networking Utilities in *Managing the AIX Operating System*.

1.1.510 uupick

Purpose
Accepts or rejects files transmitted to a user.

Syntax

```
          +-------------+
uupick ---¦             +---¦
          +- -s system -+
```

Note: This command does not have MBCS support.

Description

The Basic Networking Utilities (BNU) command **uupick** accepts or rejects files that the BNU command **uuto** has transmitted to a designated user ID.

After the files have arrived, the **rmail** command notifies the specified user. At that point, the user issues **uupick** to receive and handle the files.

Specifically, the **uupick** command searches the public directory on the local system for files with some form of the following name:

/usr/spool/uucppublic/receive/user_ID/system/file

For each entry (file or directory) found, **uupick** displays the following message on the screen of the local system:

```
from system: [file file-name] [dir dirname]
?
```

It then waits for a response from standard input to determine the disposition of the file. Issuing the **uupick** command with the appropriate file-handling option completes the transfer.

Subtopics

1.1.510.1 File-Handling Options

After notifying the specified user that a file has been sent from **system**, **uupick** displays a question mark (?), prompting for one of the following file-handling options:

*****	Displays all the file-handling options.
Enter	Moves on to the next entry in the **receive** directory.
a [dir]	Moves all **uuto** files currently in the **receive** directory into a specified directory on the local system. The default is the current working directory. Use a full or relative path name to specify **dir**.
d	Deletes the specified file.
m [dir]	Moves the specified file to a specified directory. If **dir** is not specified as a complete path name, a destination relative to the current directory is assumed. If no destination is given, the default is the current working directory on the local system.
p	Displays the contents of the file on the work station screen.
q	Stops processing and exits from the **uupick** command.
Ctrl-D	Same as **q**.
!cmd	Escapes to a shell to run the specified AIX command. After the command executes, returns automatically to **uupick** so the user can continue to handle the **uuto** files in the **receive** directory.

Flags

-ssystem	Searches **/usr/spool/uucppublic/receive/user_ID/system/file** only for files sent from the specified system.

Examples

1. To receive **file1** sent with the **uuto** command from user **msg** on system **apollo**:

   ```
   uupick
   ```

   The system responds:

   ```
   from system apollo: file file1
   ?
   ```

2. Enter an asterisk (*) to display the **uupick** file-handling options:

   ```
   ?
   *
   ```

   The system responds:

   ```
   usage [d] [m dir] [a dir] [p] [q] [cntl-d] [!cmd] [*] [new-line]
   ```

Enter the appropriate option, or use the **q** option or the **Ctrl-D** sequence to exit from the **uupick** command.

Files

/usr/spool/uucppublic Public directory.

Related Information

See the following commands: "bellmail" in topic 1.1.38, "uuto" in topic 1.1.513, "uucp" in topic 1.1.506, and "uux" in topic 1.1.515.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.511 uusched

Purpose
Schedules work for the BNU file transport program.

Syntax

```
          +-----------------+   +-----------------+
uusched ---¦                +---¦                 +---¦
          +- -u debug-level -+   +- -x debug-level -+
```

Warning: See restrictions, Chapter 18, *AIX Programming Tools and Interfaces*.

Description

The **uusched** program is the Basic Networking Utilities (BNU) file-transport scheduler. It is one of the BNU daemons (a program executed internally to handle file transfers and command executions).

The **uusched** daemon schedules the transfer of files that are queued in the **/usr/spool/uucp** directory. The scheduling program first randomizes the work and then starts the **uucico** daemon with the **-s** option. This option specifies the computer for which the particular job is scheduled.

The **uusched** program itself is usually started by the shell **uudemon.hour**, which is started from **/usr/spool/cron/crontabs/uucp**, which is, in turn, started by **cron**.

Flags

-udebug_level	Passes as **-xdebug_level** to **uucico**. The **debug_level** is a number from 0 to 9. Higher numbers give more detailed debugging information, which is displayed on the screen of the local system.
-xdebug_level	Outputs debugging messages from **uusched**. The **debug_level** is a number from 0 to 9. Higher numbers give more detailed debugging information.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.
/usr/spool/cron/crontabs/uucp	Contains **uudemon.cleanup**.
/usr/spool/uucp/*	Spooling directory.
/usr/spool/uucppublic/*	Public directory.

Related Information

See the following commands: "cron" in topic 1.1.97, "uucico" in topic 1.1.503, "uucp" in topic 1.1.506, "uustat" in topic 1.1.512, and "uux" in topic 1.1.515.

1.1.512 uustat

Purpose

Reports the status of and provides rudimentary job control for BNU commands.

Syntax

```
                   one of
             +-------------------+
             ¦ -a       -p       ¦
          +--¦ -k jobid -q       +--+
          ¦  ¦ -m       -r jobid ¦  ¦
uustat ---¦  +-------------------+  +---¦
          ¦   +- -s system -+       ¦
          +---¦             +-------+
              +-- -u user --+ ¦
            +-----------------+
```

Note: This command does not have MBCS support.

Description

The Basic Networking Utilities (BNU) command **uustat** displays status information about several types of BNU operations. It is particularly useful in monitoring transfer (copy) requests issued with the **uucp** and **uuto** commands, and requests to run an AIX command(s) on a remote system made with the **uux** command.

In addition, **uustat** also gives a user limited control over BNU jobs queued to run on remote systems. By issuing the command with the appropriate flag, a user can check the general status of BNU connections to other systems, and cancel copy requests made with **uucp** and **uuto**.

If the **uustat** command is issued without any flags, the command reports the status of all BNU requests issued by the current user since the last time the holding queue was cleaned up (see the description of the **-a** flag for an explanation of the BNU queues). Such status reports are displayed in the following format:

jobid date/time status system_name user_ID size file

See the Examples for an explanation of this format.

Note: When sending files to a system that has not been contacted recently, it is a good idea to use **uustat** to see when the last access occurred, as the remote system may be down or out of service.

Flags

The following flags are mutually exclusive; you can use only one at a time with the **uustat** command:

-a Displays information about all the jobs in the holding queue, regardless of the user who issued the original BNU command.

Note: There are two types of BNU queues.

The current queue lists the BNU jobs either queued to run on, or currently executing on, one or more specified computers. Use the **uustat -q** command to examine this queue.

The holding queue, accessed with the **-a** flag, lists all jobs that have not executed during a set period of time. After the set time period has elapsed, the entries in the holding queue are deleted either manually with the BNU command **uucleanup** or automatically with the file **/usr/spool/cron/crontabs/uucp** (which includes **uudemon.cleanup**), which is started by **cron**.

-k jobid Cancels (kills) the BNU process specified by the **jobid**. The person using this flag must either be the one who made the **uucp** request now being canceled, or must be operating with superuser authority.

Note: This flag cancels a process only when that job is still on the local computer. Once BNU has moved the job to a remote system for execution, **-k jobid** cannot be used to cancel the remote job.

-m Reports the status of the most recent attempt to contact the specified system with a BNU command. If the BNU request was completed, the status report is SUCCESSFUL. If the job was not completed, the status report is an error message such as LOGIN FAILED.

-p Runs a **ps -flp** (process status: full, long list of specified process IDs) for all PID numbers in the lock files.

-q Lists the jobs currently queued to run on each system; these jobs are either waiting to execute or in the process of executing. If a status file exists for the system, its date, time, and status information are reported. Once the job is finished, BNU removes that job listing from the current queue.

Note: In a status report, a number in parentheses next to the number of a **C.*** (command) file or an **X.*** (execute) file represents the age in days of the oldest **C.*/X.*** file for that system. The retry field represents the number of times BNU tried and failed to execute the command because of such factors as a failed login, locked files or an unavailable device.

-r jobid Marks the files in the holding queue specified by **jobid** with the current date and time. Use this flag to ensure that a cleanup operation does not delete files until the job's modification time reaches the end of the specified period.

You can use either one or both of the following flags with **uustat**:

-ssystem Reports the status of BNU requests for the work station specified by **system**.

-uuser_ID Reports the status of BNU requests by the specified user for any work station.

Examples

1. To display the status of all BNU jobs in the holding queue:

   ```
   uustat -a
   ```

   The system responds with a display like the following:

   ```
   heraC3113   11/06-17:47 S hera   amy 289   D.venus471afd8
   zeusN3130   11/06-09:14 R zeus   geo 338   D.venus471bc0a
   merlinC3120 11/05-16:02 S merlin amy 828   /u/amy/tt
   merlinC3119 11/05-12:32 S merlin msg rmail amy
   ```

   The first field is the job ID of the operation, which is followed by the date and time the BNU command was issued. The third field is either an S or an R, depending on whether the job is to send or request a file. The fourth field is the name of the system on which the command was entered, followed by the user ID of the person who issued the command. The sixth field is the size of the file, or, in the case of a remote execution like the last entry in the example, the name of the remote command. When the size is given, as in the first three lines of the example output, the file name is also displayed. The file name can be either the name given by the user, as in the **/u/amy/tt** entry, or a name that BNU assigns internally to data files associated with remote executions, such as **D.venus471afd8**.

2. To display the status of all jobs in the current queue:

   ```
   uustat -q
   ```

   The system responds:

   ```
   merlin  3C      07/15-11:02  NO DEVICES AVAILABLE
   hera    2C      07/15-10:55  SUCCESSFUL
   zeus    1C (2)  07/15-10:59  CAN'T ACCESS DEVICE
   ```

   The output tells how many **C.*** (command) files are waiting for each system. The date and time refer to the current interaction with the system, followed by a report of the status of the interaction. The number in parentheses (2) in the third line of the example indicates that the **C.*** file has been in the queue for two days.

3. To display all process IDs in the lock file:

   ```
   uustat -p
   LCK..tty0: 881
   LCK.S.0: 879
   LCK..hera: 881
   F   S UID  PID PPID C  STIME     PRI NI ADDR SZ  WCHAN     TTY
   TIME   CMD
   101 S uucp 881 879  26 09:57:03  39  39 370  296 3fffe800  -
   0:00   UUCICO -rl -shera
   101 S uuc  879 1    11 33  39 770  156 8d874     09:57:02  -
   0:00   /usr/lib/uucp/uusched
   ```

4. To cancel a job in the current queue, first determine the job ID and then execute the **uustat -k** command:

```
uustat -a
heraC3113   11/06-17:47 S hera   amy 289 D.venus471afd8
merlinC3119 11/06-17:49 S merlin geo 338 D.venus471bc0a
uustat -k heraC3113
```

5. To report the status of jobs requested by system **hera**:

```
uustat -s hera
heraNlbd7  07/15-12:09 S hera amy 522    /user/amy/A
heraClbd8  07/15-12:10 S hera amy  59    D.3b2a12ce4924
heraC3119  07/15-12:11 S hera amy rmail  msg
```

6. To report the status of jobs requested by user **amy**:

```
uustat -u amy
```

   This flag displays output similar to that produced by the **-s** flag.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/usr/spool/uucp	Spooling directory.

Related Information

See the following commands: "ps" in topic 1.1.337, "uucp" in topic 1.1.506, "uuto" in topic 1.1.513, and "uux" in topic 1.1.515.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.513 uuto

Purpose
Copies public files from one AIX system to another AIX or Unix system, with local system control of file access.

Syntax

```
        +--------+
uuto ---¦ +----+ +--- source --- destination --¦
        +-¦ -p +-+           ¦
          ¦ -  ¦¦  +----------+
         ¦+----+¦
         +------+
```

Note: This command does not have MBCS support.

Description

The Basic Networking Utilities (BNU) command **uuto** copies one or more **source** files from one AIX system to a specified user on another AIX or Unix system. The **uuto** command calls the BNU command **uucp** for the actual file transfer, but **uuto** enables the recipient to use the **uupick** options to handle the transferred files on the local system.

The **source** entry is the name of the files on the local system, or a path name to the files on the system that runs the command. The destination is a specific user ID. This entry has the following format:

system!user

where **system** is the name of a remote system connected to the local system, and **user** is the login name of the recipient of the transferred files on the specified system.

Note: When copying a file from one user to another user on the local system, omit the **system** entry; the destination is simply the ID of the user to whom the file is being sent.

The **uuto** command sends files to **/usr/spool/uucppublic** on the designated **system**; this is a public directory. The command also creates an additional directory called **receive** (if it does not already exist), plus the directory /**user/system**. The full path names to the copied files are therefore some form of the following:

/usr/spool/uucppublic/receive/**user/system/files**

Once the copied file is in the **receive** directory, **uuto** notifies the recipient by **rmail** that the file has arrived. The recipient then issues the **uupick** command, which searches the public directory for files sent to the specified user ID, displaying the message that file **name** has arrived from system **name** for each file it locates. The user then enters one of the **uupick** file-handling options to delete the file, move it to another directory, and so on.

Flags

-m Notifies the sender by **bellmail** when the copy is complete.

-p	Copies the source file to the spool directory on the local system. The source file resides in the spooling directory for a set period of time (defined in the **uusched** program) before the **uucp** command calls the **uucico** daemon, which actually transfers the copy to the public directory on the specified remote system. The default is to transfer a source file directly to the specified user.

Files

/usr/spool/uucppublic	Public directory.

Related Information

See the following commands: "bellmail" in topic 1.1.38, "uucleanup" in topic 1.1.504, "uucp" in topic 1.1.506, "uupick" in topic 1.1.510, "uusched" in topic 1.1.511, "uustat" in topic 1.1.512, and "uux" in topic 1.1.515.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.514 uutry, Uutry, uukick, Nutry

Purpose
Contacts a remote system with debugging turned on.

Syntax

```
+-------+
¦ Uutry ¦   +------------------+   +------+
¦ uutry +---¦                  +---¦      +-- system-name --¦
¦ Nutry ¦   +- -x debug-level -+   +- -r -+
+-------+


          +------------------+
uukick ---¦                  +-- system-name --¦
          +- -x debug-level -+
```

Note: This command does not have MBCS support.

Description

The **uutry** command contacts a specified system with debugging turned on, thus providing a means of checking call processing capabilities with debugging output. This command invokes the **uucico** program, which in turn establishes the actual connection to the remote system.

The debugging output (information about the progress of **uucico** in establishing the connection, performing the remote login, and so on) is scrolled on the screen of the local system. Once the system has finished displaying this information, use **INTERRUPT** (**Ctrl-D**) to return to the prompt. Because the debugging information scrolls rapidly, you may want to direct that output to a file by issuing **Uutry** rather than **uutry**. **nutry** works like **Uutry** except that it does not look for **uucico** in the current directory.

The **Uutry** command (note the uppercase "U") works almost exactly like **uutry**, with one exception. In addition to displaying the debugging output on the screen, **Uutry** also directs this information to a file named **/tmp/system_name**. Again, when the last of the output has been displayed, use **INTERRUPT** to return to the prompt.

Note: You can also press **INTERRUPT** while the system is scrolling the output generated by **Uutry**. This returns you to the prompt, while the **uucico** program continues to place the debugging information in **/tmp/system_name**. Use the **pg** command to examine this file.

The **uukick** command also works just like the **uutry** command. The only difference between the two commands is that **uukick** takes only the **-xdebug_level** flag. You cannot override the retry time with the **-rsystem_name** flag.

Flags

-xdebug_level	Used in debugging to override the default level of 5, and produce a detailed output of the program execution. The debugging level is a single digit between 0 and 9. Higher numbers produce more detailed debugging information, which is displayed on the screen of the local system.

-r	Overrides the retry time specified in **/usr/spool/uucp/.Status**.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/tmp/system_name	Temporary data file.
/usr/adm/uucp/Devices	Information about available devices.
/usr/adm/uucp/Dialcodes	Dialing code abbreviations.
/usr/adm/uucp/Dialers	Initial handshaking on a link.
/usr/adm/uucp/Maxuuscheds	Limits scheduled jobs.
/usr/adm/uucp/Maxuuxgts	Limits remote command executions.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/adm/uucp/Systems	Accessible remote systems.
/usr/spool/uucp/*	Spooling directory.
/usr/spool/uucppublic/*	Public directory.

Related Information

See the following commands: "uucico" in topic 1.1.503, "uucp" in topic 1.1.506, and "uux" in topic 1.1.515.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.515 uux

Purpose
Runs a command on another AIX or UNIX system.

Syntax

```
       +-- -c --+    +--------+    +----------------------------+
uux ---¦ one of +---¦ one of +---¦ +-------------------------+ +-- cmdstring ·
       ¦ +----+ ¦    ¦ +----+ ¦    +-¦ -         -p              +-+
       +-¦ -c +-+    +-¦ -n +-+      ¦ -a nam    -r              ¦¦
         ¦ -C ¦        ¦ -z ¦       ¦¦ -b        -s file         ¦¦
         +----+        +----+       ¦¦ -g grade -x debug-level   ¦¦
                                    ¦¦ -j                        ¦¦
                                    ¦+-------------------------+¦
                                     +---------------------------+
```

Note: This command does not have MBCS support.

Description

The Basic Networking Utilities (BNU) command **uux** runs a specified AIX command on a specified AIX or Unix system.

The command gathers various files from the designated systems, if necessary. It then runs a specified command on a designated system. The user can direct the output from the command to a specified file on a specified system.

Note: For security reasons, many installations permit **uux** to run only the **rmail** command.

The **uux** command creates execute (**X.***) files that run AIX commands on the local system. In addition, **uux** also creates both command (**C.***) files and data (**D.***) files. Execute files contain the command string to be executed on the designated system. Command files contain the same information as those created by the **uucp** command. Data files either contain the data for a remote command execution, or else become **X.*** files on remote systems for remote command executions.

Note: The full path name of an execute file is a form of the following:

/usr/spool/uucp/system_name/X.system_nameNxxxx

After creating the files in the spooling directory, **uux** calls the **uucico** daemon, which in turn attempts to contact the designated system to deliver the files. Once the files are transferred, the **uuxqt** daemon executes the **cmdstring** on the specified system.

The **cmdstring** is made up of one or more arguments that look like an AIX command line, except that **cmdstring** may be prefixed by **system_name!**. The default **system_name** is the local system.

Note: To run commands on more than one system, type the information on separate command lines:

```
uux merlin!print /reports/memos/charles
uux zeus!print /test/examples/examp1
```

Unless the **-n** flag is specified, **uux** notifies the user if the remote system does not run the command. The response comes by **mail** from the other system.

Subtopics

1.1.515.1 File Names, Path Names, and System Names

When specifying the destination of the output of a command, **uux** may be entered in either one of the following formats:

- **uux** [**options**] "**cmdstring** > **destination_name**"

- **uux** [**options**] **cmdstring** \{**destination_name**\}

Destination names may be either of the following

- a full path name

- a full path name preceded by **~user**, where **user** is a login name on the specified system. The **uux** command replaces this path name with the user's login directory.

The shell pattern-matching characters **?**, *****, and **[...]** may be used in the path name of a "source" file (such as files compared by the **diff** command); the appropriate system expands them. However, using the * character may occasionally produce unpredictable or unanticipated results.

Note: Shell pattern-matching characters should not be used in the destination path name.

Place either two backslashes (\...\) or a pair of quotation mark ("...") around pattern-matching characters in a path name so the local shell cannot interpret them before **uux** sends the command to a designated system.

If using the special shell characters > (greater than), < (less than) ; (semicolon), or ¦ (vertical bar) in a path name, place either \...\ or "..." around the individual character or around the entire command string.

Do not use the shell redirection characters << or >> in a path name

The **uux** command attempts to move all files specified on the command line to the designated system. Enclose the names of all output files in parentheses so that **uux** does not try to transfer them.

When specifying a **system_name**, always place it before the **cmdstring** in the entry.

The exclamation point preceding the name of the local system in command is optional. If you choose to include the ! to run a command on the local system using files from two different remote systems, use ! instead of **system**! to represent the local system, and add **system**! as the first entry in any path name on the remote systems.

The exclamation point representing a remote system in BNU syntax has different meaning in c shells (**csh**). When running **uux** in a c shell, place a backslash (\) before the exclamation point in a system name.

If the command being executed requests two files stored on the sam system, or two files with the same name that are stored on separate systems, the command executes, but does not produce the desired results.

The following two commands do execute:

```
uux "hera!/bin/diff /usr/amy/out1 hera!/u/amy/out > ~uucp/DF"
uux "hera!/bin/diff hera!/usr/amy/out1 venus!/u/amy/out > ~uucp/DF"
```

Note: The notation **~uucp** is the shorthand way of specifying the public spooling directory **/usr/spool/uucppublic**.

In the first command, **diff** is on system **hera**, the first source file is on the local system, the second source file (with a different name) is on system **hera**, and the output is directed to the file **DF** in the public directory on the local system. In the second command, **diff** is again on **hera**, the first file is also on **hera**, the second file (with a different name) is on **venus**, and the output is again directed to **DF** in the **~uucp** directory.

The following command does not execute properly:

```
uux "hera!/bin/diff venus!/u/amy/out merlin!/u/amy/out > ~uucp/DF"
```

This command does not execute because although the files are on two different systems, they still have the same file name.

Flags

-	Makes the standard input to **uux** the standard input to the **cmdstring**.
-aname	Replaces the user ID of the person issuing the command with user ID specified with **name**.
-b	Returns standard input to the command if the exit status is not zero.
-c	Transfers the source files to the destination on the specified system. The source files are not copied into the spool directory for transfer. (See the discussion of the **-C** flag.) This flag is on by default.
-C	Transfers the source files to the spool directory. After a set period of time (specified in the **uusched** program), the **uucico** daemon attempts to transfer the files to the destination on the specified computer. **Note:** Occasionally, there are problems in transferring a source file. For example, the remote computer may not be working or the login attempt may fail. In such cases, the file remains in the spool directory until it is either transferred successfully or removed by the **uucleanup** command.
-ggrade	Specifies when the files are to be transmitted during a particular connection. **Grade** is a single number (0-9) or letter (A-Z, a-z); lower ASCII-sequence characters cause the files to be transmitted earlier than do higher sequence characters. The number 0 is the highest (earliest) grade; z is the lowest (latest). The default is **N**.
-j	Displays the job identification number of the process

that is running the command on the specified system. Use this job ID with the BNU command **uustat** to check the status of the command, or with **uustat -k** to terminate the process.

-n	Prevents user notification by **mail** whether or not the command completes successfully. The default is to notify the user if the command fails.
-p	Uses the standard input to **uux** as the standard input to **cmdstring**. A **-** (minus) has the same effect.
-r	Prevents the starting of the spooling program that transfers files between systems. The default is to start the spooling program.
-sfile	Reports the status of the transfer in a file specified by **file** on the designated system.
-xdebug_level	Displays debugging information on the screen of the local system. The **debug_level** is a number between 0 and 9. The higher number gives a more detailed report.
-z	Notifies the user only if the command completes successfully.

Examples

1. To get the **jobid** of a job and then compare a file on the local system **zeus** with a file on a remote system when the **diff** command is stored on the local system, use either of the following formats:

   ```
   uux -j "/bin/diff /usr/amy/f1 hera!/u/amy/f2 > ~uucp/f1.diff"
   ```

   or

   ```
   uux -j /bin/diff /usr/amy/f1 hera!/u/amy/f2 \{~uucp/f1/diff\}
   ```

   This command gets the file **/u/amy/f2** from the remote system **hera**, compares it to the file **/usr/amy/f1** on the local system (**zeus**), and places the output of the command in the local public directory in a file named **f1.diff**. (The full path name of this file is **/usr/spool/uucppublic/f1.diff**.) Using the **-j** option produces the output **zeusN52d9**.

   Note: As shown in the example, the destination name must be entered either preceded by a > with the whole command string enclosed in "...", or entered enclosed in braces and backslashes, as \{...\}.

2. To compare files that are located on two different remote systems, **hera** and **venus**, using the **diff** command on the local system:

   ```
   uux "!/bin/diff hera!/usr/amy/f1 venus!/u/amy/f2 > !f1.diff"
   ```

   This command gets the **/usr/amy/f1** file from the system **hera** and the **/u/amy/f2** file from **venus**, runs a **diff** command on the two files, and places the results in the file **f1.diff**, located in the current working directory on the local system.

This output file must be write-enabled. If you are uncertain about the permission status of a specific target output file, direct the results to the public directory, as in the first example.

The exclamation points representing the local system are optional.

Both of the examples above use a > symbol preceding the name of the output file. When using the special shell characters >, <, ;, or ¦, either quote the entire **cmdstring**, or quote the special characters as individual arguments.

3. To specify an output file on a different remote system:

```
uux hera!uucp venus!/u/amy/f1 \{merlin!/u/geo/test\}
```

   This command runs **uucp** on system **hera**. The **uucp** command then sends the file **/u/amy/f1**, stored on system **venus**, to user **geo** on system **merlin** as **test**.

4. To get selected fields from a file on system **hera** and place them in a file on the local system:

```
uux "cut -f1 -d: hera\!/etc/passwd > ~uucp/passw.cut"
```

   This command runs **cut** on the local system, gets the first field from each line of the password file on system **hera**, and places the output in the file **passw.cut** in the public directory on the local system.

   Note: In this example, **uux** is running in a c shell, so a \ (backslash) must precede the exclamation point in the name of the remote system.

Files

/usr/spool/uucp	Spooling directory.
/usr/lib/uucp	Contains the **uucico** daemon.

Related Information

See the following commands: "mail, Mail" in topic 1.1.253, "uucico" in topic 1.1.503, "uucp" in topic 1.1.506, "uustat" in topic 1.1.512, and "uuxqt" in topic 1.1.516.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.516 uuxqt

Purpose
Executes remote command requests.

Syntax

```
        +------------+   +----------------+
uuxqt ---¦            +---¦                +---¦
        +- -s system -+   +- -x debug-level -+
```

Note: This command does not have MBCS support.

Description

The Basic Networking Utilities (BNU) program **uuxqt** executes specified commands on designated remote systems.

Once **uux** is entered by a user, the program creates the necessary command (**C.***), data (**D.**), and execute (**X.***) files and places them in the spooling directory on the designated system. The **uux** command then calls the **uucico** daemon, which in turn attempts to contact the designated system to deliver the files. When the files have been transferred, **uuxqt** executes the commands on the designated system.

The **uuxqt** program searches the spool directories on the designated system for **X.*** files whose names indicate that they have been sent from another system. The command checks each execute file for the following:

that all the required data **D.***) files are available and accessible

that file commands are permitted for the requesting system

Note: BNU uses the **Permissions** file to validate file accessibility and command execution permission.

The **uuxqt** program, one of the Basic Networking Utilities (BNU) daemons (a program executed to handle file transfers and command executions), can be executed manually by an individual with superuser privileges. This daemon is executed automatically by the **uudemon.hour** shell script, which is started periodically by **cron**.

Flags

-ssystem	The name of the remote system. Use only when starting **uuxqt** manually. The **system** name is supplied internally when **uuxqt** is started automatically.
-xdebug_level	Displays debugging information on the screen of the local system. The **debugg_level** is a single digit between 0 and 9. The higher the number, the more detailed the debugging information.

Files

/etc/locks/LCK*	Prevents multiple use of device.
/usr/adm/uucp/Maxuuxqts	Limits remote command executions.
/usr/adm/uucp/Permissions	Access permission codes.
/usr/spool/uucp/*	Spooling directory.

Related Information

The following commands: "uucico" in topic 1.1.503, "uucp" in topic 1.1.506, "uustat" in topic 1.1.512, and "uux" in topic 1.1.515.

See the chapter on basic networking utilities in *Managing the AIX Operating System*.

1.1.517 uvcp

Purpose
Sends (file transfer) files in NETDATA format.

Syntax

```
       +--------+   +--------+   +- -V 65533 -+
uvcp ---¦ +----+ +---¦ +----+ +---¦   one of   +---
       +-¦ -b +-+   +-¦ -a +-+   ¦ +--------+ ¦
         +----+       ¦ -n ¦     +-¦ -f num +-+
                      +----+       ¦ -v num ¦
                                   +--------+

  +------------+              +-----------+
--¦            +--- -d user ---¦           +---¦
  +--- file ---+              +- at node -+ ¦
             ¦   +--------------------------+
    +--------+
```

Note: This command is for the System/370 only.

Description
Using the **uvcp** command you can:

Specify that a binary file is being sent. In this case, all new-lin characters in the file are to be treated as data.

Request an acknowledgement when the target user has received the file

The **uvcp** command processor verifies the existence of the file you request to be sent and the translate table if specified. If either of these files does not exist, you are sent an error message and processing is terminated:

```
  UVC204 uvcp: CANNOT ACCESS FILE
```

If there is no problem, the file is processed and then transmitted by writing a VM spool file in the NETDATA format. Spool files are created by writing to **/dev/pun**.

If the target user is a CMS user on the local VM system, the spool file is punched directly to that user.

Otherwise, even if the target user is on another AIX/370 guest machine on your local VM system, the spool file is forwarded via the RSCS virtual machine. RSCS is responsible for transmitting the file across the network or to the target guest machine on the local VM system.

RSCS sends messages as the file is sent to the next node or as errors are detected. In this case, RSCS believes the sender to be your AIX/370 guest machine and therefore sends the messages to your AIX/370 system console and not to you. However, if you use the **-a** option, an acknowledgement is sent to you when the file has been received by the target user.

When you specify several options (for example, multiple **-t**) on the same **uvcp** command, the last one encountered upon scanning the command from left to right is used.

Defaults for the uvcp Command

The defaults for the **uvcp** command are:

Translate to IBM EBCDIC using the translation table specified via th environment variable **NLOUT**.

Note: If you have selected a language that supports multibyte characters, the environment variable **NLOUT** is not used. Instead, the **iconv** command is used to translate the file or network code set of the selected locale to EBCDIC. For information on supported code sets, see "iconv" in topic 1.1.204.

New-line character is considered as region/record delimiter

No acknowledgement is requested

The target CMS or MVS/TSO systems are to receive the file as a format file with a maximum record length of 65533. The real record length is calculated by the receiving system.

Standard input is used to read file names if no input file i specified.

Note: Use standard input for file names only. The **uvcp** command uses file names to create equivalent file names in VM.

Flags

You can modify the actions of the **uvcp** command with the following options:

-b Specifies that a binary file is being sent. This file is not translated. The new-line characters in the input file are treated as data and not as a region/record delimiter in the VM and MVS/TSO terminology.

-n Specifies that the file to be sent is treated as a **Note** by the receiving CMS or MVS/TSO user. The **-f** and **-v** options are ignored (in this case) by the receiving CMS or MVS/TSO system because a note file is always appended to a **notebook** file.

-a Specifies that an acknowledgement is requested. Unless you specify this option, acknowledgement is not requested from the target user.

-fnum Specifies that the CMS or the MVS/TSO system is to receive this file as a fixed format file; **num** specifies the fixed record length and must be a number between 1 and 65535. When receiving the file at the CMS or MVS/TSO systems, all records shorter than this length are padded with blanks and all records longer than this length are truncated.

-vnum Specifies that the CMS or MVS/TSO system is to receive this file as a variable format file; **num** specifies the maximum record length and must be a number between 1 and 65533. The **num** only has an effect when you use the **-b** option. However, if you do not specify a **num** within the required limits, you receive an error message.

Note: When a file is received by an AIX/370 user, the options **-f** and **-v** are ignored during **vucp** command processing. These options are ignored by the CMX or MVS/TSO systems if the **-n** option is used.

file Specifies the file or files to be sent. If **file** is not specified, standard input is used. Each file:

Must be a normal file; the sending of directories or special files is not supported. Object (binary) files should be sent by using the **-b** option.

Note: If you have selected a language that supports multibyte characters, the file to be transferred may be in the format of the selected file or network code set. For example, if you selected a Japanese locale, the file transferred is in Japanese. For information on supported code sets, see "iconv" in topic 1.1.204.

May be identified using either an absolute (full) file name or relative file name.

You must have read access to each file.

-d user [at node] Specifies the CMS user ID to whom and where the file is to be sent. The **at node** may be omitted if the target user is on the same physical 370 computer on which this AIX/370 guest is running. There is no restriction on the number of destinations that can be specified, except for the limit on the length of the command string in AIX/370. At least one destination must be specified.

Files

/usr/lib/nls/nlout Directory containing the ASCII/EBCDIC translation tables.

/etc/qconfig The pun and pundev stanzas delivered to the user commented out, should be uncommented. The administrator should then type **print -rr** on the 370 that is acting as the gateway.

/local/system.netid The VM userid can be found by typing **cpcmd q userid** on the gateway.

/usr/lpd/stat/rdrlog Holds the status/error log.

Related Information

See the following commands: "vucp" in topic 1.1.527, "iconv" in topic 1.1.204, and "axeb, ebxa" in topic 1.1.30.

1.1.518 vacation

Purpose
Returns "I am on vacation" indication.

Syntax

vacation --- -I ---¦

vacation --- **user** ---¦

Description

The **vacation** command returns a message to the sender of a message telling that you are on vacation. The intended use is in a **.forward** file. For example, your **.forward** file might have:

```
  \eric, "¦vacation eric"
```

which would send messages to you (assuming your login name was eric) and send a message back to the sender.

The **vacation** command expects a file **.vacation.msg** in your home directory containing a message to be sent back to each sender. It should be an entire message (including headers). For example, it might say:

```
  From: eric@ucbmonet.Berkeley.EDU (Eric Allman)
  Subject: I am on vacation
  Delivered-By-The-Graces-Of: the Vacation program

  I am on vacation until July 22.  If you have something urgent,
  please contact Joe Kalash <kalash@ucbingres.Berkeley.EDU>.
      --eric
```

This message is only sent once a week to each unique sender. The people who have sent you messages are kept in the files **.vacation.pag** and **.vacation.dir** in your home directory. The **-I** option initializes these files, and should be executed before you modify your **.forward** file.

If the **-I** flag is not specified, **vacation** reads the first line from the standard input for a UNIX-style "From" line to determine the sender. If this is not present, a nasty diagnostic is produced. **sendmail** includes the "From" line automatically.

No message is sent if the initial "From" line includes the string "-REQUEST@" or if a "Precedence: bulk" or "Precedence: junk" line is included in the header.

Related Information

See the following command: "sendmail, mailq, newaliases" in topic 1.1.417.

1.1.519 val

Purpose
Validates Source Code Control System (SCCS) files.

Syntax

```
       +------------+
val ---¦ +--------+ +-- file --¦
       +-¦ -mname +-+        ¦
         ¦ -rSID  ¦¦  +------+
        ¦¦ -s     ¦¦
        ¦¦ -ytype ¦¦
        ¦+--------+¦
        +----------+
```

Description
The **val** command reads **file**s and determines if the specified **file** is a Source Code Control System (SCCS) file meeting the characteristics specified by the flags. If you specify a **-** (minus) for **file**, **val** reads standard input and interprets each line of standard input as **val** flags and the name of an SCCS file. **val** continues to take input until it reaches an end-of-file character (**Ctrl-D**).

The **val** command displays error messages to standard output for each file processed. **val** also returns a single 8-bit code upon exit. The 8-bit code indicates possible mismatches or errors. It is interpreted as a bit string in which set bits (from left to right) are interpreted as follows:

bit 0 = missing file parameter
bit 1 = unknown or duplicate flag
bit 2 = damaged SCCS file
bit 3 = cannot open file or file not SCCS
bit 4 = SID is invalid
bit 5 = SID does not exist
bit 6 = %**Y**%, **-y** mismatch
bit 7 = %**M**%, **-m** mismatch

When **val** processes two or more files on a given command line or multiple command lines (when reading the standard input), a code is returned that is a logical **OR** of the codes generated for each command line and file processed. **val** can process up to 50 files on a single command line. Any number above 50 produces a dump.

Flags

Each flag or group of flags applies independently to each named file. The flags may appear in any order.

-mname	Compares the value **name** with the SCCS %**M**% identification keyword in **file**. See "Identification Keywords" in topic 1.1.186.2 for more information on the %**M**% keyword.
-rSID	Specifies the **SID** of the **file** to be validated. The **SID** must be valid and unambiguous.
-s	Suppresses the error message normally written to standard output.

-ytype Specifies a **type** to compare with the SCCS %**Y**% identification keyword in **file**. See "Identification Keywords" in topic 1.1.186.2 for more information on the %**Y**% keyword.

Related Information

See the following commands: "admin" in topic 1.1.16, "delta" in topic 1.1.117, "get" in topic 1.1.186, and "prs" in topic 1.1.336.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.520 vc

Purpose
Substitutes assigned values in place of keywords.

Syntax

```
      +-------------+   +-----------------+
vc ---¦ +---------+ +---¦                 +---¦
      +-¦ -a      +-+   +- keyword=value -+
        ¦ -t      ¦¦                     ¦
       ¦¦ -c char ¦¦     +---------------+
       ¦¦ -s      ¦¦
       ¦+---------+¦
       +-----------+
```

Description
The **vc** command is used to control writing different versions of a file to standard output. However, since Source Code Control System commands ("admin" in topic 1.1.16) provide more efficient and complete control, they should be used instead of **vc**.

The **vc** command copies lines from standard input to standard output. The flags and **keywords** on the command line and control statements in the input modify the resulting output. **vc** replaces user declared **keywords** with their **value** assigned on the command line. These **keywords** can be replaced both in text and in control statements.

Subtopics

1.1.520.1 Control Statements

A control statement is a single line beginning with a control character (the default control character is a **:** (colon)). They provide conditional processing of the input. The allowable types of control statements are:

:if condition

text

:end Writes all the lines between the **:if** statement and the matching **:end** to standard output only if **condition** is true. You can nest **:if-:end** statements, but once a condition is false, all remaining nested **:if-:end** statements are ignored. See "Condition Syntax" in topic 1.1.520.2 for the syntax of conditions and allowable operators.

:dcl keyword [, keyword ...]
Declares **keyword**s. All **keyword**s must be declared.

:asg keyword=value
Assigns **value** to **keyword**. An **:asg** statement takes precedence over keyword assignment on the **vc** command line. A later **:asg** statement overrides all earlier assignments of the associated **keyword**. **keyword**s declared, but not assigned **values** have null values.

:: text Removes the two leading control characters and replaces **keyword**s with their **value**s, and then copies the line to the standard output.

:on

:off Turns on or off **keyword** replacement on all lines.

:ctl char
Changes the control character to **char**.

:msg message
Writes a message to standard error output in the form:

Message(n): **message**

where **n** is number of the input line on which the message appeared.

:err message
Writes an error message to standard error in the form:

```
ERROR: message
ERROR: err statement on line n (vc15)
```

vc stops processing and returns an exit **value** of 1.

1.1.520.2 Condition Syntax

Item	**Statements Allowed**
condition	::=or statement
	::=**not** or statement
or statement	::=and statement
	::=and statement | or statement
and statement	::=expression
	::=expression **&** and statement
expression	::=**(** or statement **)**
	::=value operator value
operator	::=**=** or **!=** or **<** or **>**
value	::=ASCII string
	::=numeric string

The available condition operators and their meanings are as follows:

=	equal
!=	not equal
&	and
|	or
>	greater than
<	less than
( )	used for logical groupings
not	may only occur immediately after the **if**, and when present, inverts the value of the entire condition.

The > and < (greater than and less than symbols) operate only on unsigned integer values; for example: 012 > 12 is false. All other operators take strings as modifiers; for example: **012 != 12** is true. The precedence of the operators, from highest to lowest precedence, is as follows:

= != > < all of equal precedence

&

|

Parentheses can be used, of course, to alter the order of precedence.

Values must be separated from operators or parentheses by at least one blank or tab.

1.1.520.3 Keyword Replacement

A **keyword** must begin and end with the same control character used in control statements. A **keyword** may be up to nine alphanumeric characters, where the first character must be alphabetic. Keyword **values** must be ASCII strings. A numeric keyword **value** is an unsigned string of digits. **values** cannot contain tabs or spaces.

Examples of **keyword=value** assignments are:

```
numlines=4
prog=acctg
pass4=yes
```

The **vc** command removes all control characters and **keyword**s from input text lines marked with two control characters as it writes the text to standard output. To prevent a control character from being interpreted, precede it with a backslash, as in the following example:

```
::the :prog: program includes several of the following\:
```

The keyword **:prog:** is replaced by its **value**, but the **\:** is passed to standard output as **:** (colon).

Input lines beginning with a \ (backslash) followed by a control character are not control lines, and are copied to standard output without the backslash. **vc** writes lines beginning with a backslash and no following control character without any changes (including the initial backslash).

Flags

-a	Replaces **keyword**s surrounded by control characters with their assigned **value** in all text lines (not just those beginning with two control characters).
-cchar	Uses **char** as the control character.
-s	Does not display the warning messages normally displayed to standard error.
-t	Ignores all characters from the beginning of a line up to and including the first tab character for detecting a control statement. If **vc** finds a control character, it ignores all characters up to and including the tab.

Related Information

See the following command: "admin" in topic 1.1.16.

1.1.521 vgrind

Purpose
Grinds (formats) nice listings of programs.

Syntax

```
          +------------+   +------------+   +----------+   +--------------+
vgrind ---¦ +--------+ +---¦            +---¦          +---¦              +---
          +-¦ -f -sn +-+   +- -hheader -+   +- -dfile -+   +- -llanguage -+
            ¦ -  -t  ¦
            ¦ -n -x  ¦
            ¦    -W  ¦
            +--------+
```

Description

The **vgrind** command formats the program sources, which are arguments, in a nice style using **troff**. Comments placed in italics, keywords in bold face, and the name of the current function is listed down the margin of each page as it is encountered.

The **vgrind** command runs in two basic modes, filter mode or regular mode. In filter mode **vgrind** acts as a filter in a manner similar to **tbl**. The standard input is passed directly to the standard output except for lines bracketed by the **troff**-like macros:

.vS	- starts processing
.vE	- ends processing

These lines are formatted as described above. The output from this filter can be passed to **troff** for output. There need be no particular ordering with **eqn** or **tbl**.

In regular mode the **vgrind** command accepts input files, processes them, and passes them to **troff** for output.

In both modes the **vgrind** command passes any lines beginning with a decimal point without conversion.

Flags

-f	Forces filter mode.
-	Forces input to be taken from standard input (default if **-f** is specified).
-d	Specifies an alternate language definitions file (default is /usr/lib/vgrindefs).
-h	Specifies a particular header to put on every output page (default is the file name).
-l	Specifies the language to use. Currently known are CSH (**-lcsh**) (which is the default), PASCAL (**-lp**), MODEL (**-lm**), C (**-lc**), SHELL (**-lsh**), RATFOR (**-lr**), MODULA2 (**-lmod2**), YACC (**-lyacc**), LISP (**-lisp**), and ICON (**-lI**).

-n	Forces no bolding of keyword.
-s	Specifies a point size to use on output (exactly the same as the argument of a .ps).
-t	Similar to the same option in **troff** causing formatted text to go to the standard output.
-x	Outputs the index file in a "pretty" format. The index file itself is produced whenever **vgrind** is run with a file called **index** in the current directory. The index of function definitions can then be run off by giving **vgrind** the **-x** option and the file **index** as argument.
-W	Forces output to the (wide) Versatec printer rather than the (narrow) Varian printer.

Files

index	File where source for index is created.
/usr/lib/tmac/tmac.vgrind	Macro package.
/usr/lib/vfontedpr	Preprocessor.
/usr/lib/vgrindefs	Language descriptions.

1.1.522 vi, vedit, view

Purpose
Edits files with a full screen display.

Syntax

```
+-------+   +------------+   +----------------+   +----------------+   +-----
¦ vi    +---¦ +--------+ +---¦      +--------+ +---¦ +---- + -----+ +---¦
¦ view  ¦   +-¦ -l     +-+   +- -r -¦        +-+   +-¦            +-+   +---
¦ vedit ¦     ¦ -t tag ¦¦           +- file -+       +- + subcmd -+
+-------+    ¦¦ -R     ¦¦                                                   +---
             ¦¦ -wnum  ¦¦
             ¦¦ -ynum  ¦¦
             ¦+--------+¦
             +----------+
```

Description

The **vi** command is a display editor based on an underlying line editor (**ex**). The **view** command is a read-only version of **vi**. In it, the **readonly** option is set to protect files during browsing. The **vedit** command is a version of **vi** intended for beginners. In it, the **report** option is set to 1, and the **showmode** and **novice** options are set. Since **novice** is set, it is a line editor. For more information on these options, see "Setting Options" in topic 1.1.522.2.

The **file** parameter specifies the file or files to be edited. If you supply more than one **file** on the command line, **vi** edits each file in the order specified.

When you use **vi**, changes you make to a file are reflected in your display. The position of the cursor on the display indicates its position within the file. The subcommands effect the file at the cursor position.

The following list provides the maximum limits of the **vi** editor. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the character limits can be reduced by as much as 50%, depending on the character code set being used.

    1024 characters per lin
    256 characters per global command lis
    128 characters in the previous inserted and deleted tex
    100 characters in a shell escape comman
    63 characters in a string-valued optio
    30 characters in a tag nam
    250,000 lines of 1024 characters per line silently enforce
    128 map macros with 2048 characters total. Each macro has a maximu
    of 100 characters.
    100 characters per each **map** macro subcommand (or "rhs").

If **vi** is run in a non-interactive manner (that is, with standard input redirected from a file), the command runs as though it were invoked as **ex** rather than **vi**.

Subtopics

1.1.522.1 Editing States

The **vi** editor has the following operational states:

command state	This is the initial state. Any subcommand can be entered (except commands that can only be used in the input state). When subcommands and the other states end, they return to this state. Pressing the **ESC** key cancels a partial command.
input state	Entered by the **a**, **A**, **i**, **I**, **o**, **O**, **c**, **C**, **s**, **S**, and **R** subcommands. After entering one of these commands, you can enter text into the editing buffer at the current cursor position. To return to the command state, press **ESC** for normal exit or press INTERRUPT (**Ctrl-C**) to end abnormally.
last line state	Some subcommands (subcommands with the prefix **:**, **/**, **?**, **!obj**, or **!!**) read input on a line displayed at the bottom of the screen. When you enter the initial character, **vi** places the cursor at the bottom of the screen, where you enter the remaining characters of the command. Press the **Enter** key to perform the subcommand and INTERRUPT (**Ctrl-C**) to cancel it.

1.1.522.2 Setting Options

The **vi** editor allows you to customize options so that you can use the editor for a specific task. Use the **set** command to set or change an option. To view the current setting of options, enter **:set all** while in **vi** command state. Some options are set to a string or a number value. Other options are simply turned on or off. To change an option that is set to a value, enter a command in the form **:set option=value**. To toggle an option that can be set to on or off, enter a line of the form **:set option** to set it on or **:set nooption** to set it off.

Options set by the **set** command last only for the current editing session. To have certain options set each time **vi** is started, use **EXINIT** variables or add an **.exrc** file to your home directory. All **set** command options and abbreviations must be entered in ASCII characters. See the *Text Formatting Guide* for details.

Options can be abbreviated in a **set** command. The following table lists some of the most commonly-used options, abbreviations, and descriptions:

Table 1-8. Commonly Used vi Options

Option	**Abbreviation**	**Description**
autoindent	**ai**	Indents automatically in text mode to the indentation on the previous line by using the spacing between tab stops specified by the **shiftwidth** option. The default is **noai**. To back the cursor up to the previous tab stop, type **Ctrl-D**. This option is not in effect for global commands.
autoprint	**ap**	Prints the current line after any command that changes the editing buffer. The default is **ap**. This option applies only to the last command in a sequence of commands on a single line, and is not in effect for global commands.
autowrite	**aw**	Writes the editing buffer to the file automatically before the **:n**, **:ta**, **Ctrl-^**, and **!** subcommands if the editing buffer has been changed since the last **write** command. The default is **noaw**.
beautifying text	**bf**	Prevents user from entering control characters (except for tab, newline, and formfeed) in the editing buffer during text entry. The default is **nobf**. This option does not apply to command input.
directory	**dir=**	Displays the directory that contains the editing buffer. The default is **dir=/tmp**.
edcompatible	**ed**	Retains global (**g**) and confirm (**c**) subcommand suffixes during multiple substitutions and causes the read (r) suffix to work like the **r** subcommand. The default is **noed**.
errorbells	**eb**	Error messages are preceded by a bell. If possible, **vi** places the error message in reverse video (white lettering on a black background)

		instead of ringing a bell. The default is **noeb**. This may not work on PS/2 machines.
flash	**fl**	Flashes error messages. The default is **flash**. If nofl is set, it always beeps on errors, even if terminal can flash. This may not work on PS/2 machines.
hardtabs	**ht=**	Tells **vi** the distance between the hardware tab stops on your display. The default is **ht=8**.
ignorecase	**ic**	Ignores distinction between uppercase and lowercase while searching for regular expressions. The default is **noic**.
lisp	**lisp**	Removes the special meaning of (), {}, [[and]] and enables the = (formatted print) operator for S-expressions, so you can edit LISP programs. The default is **nolisp**.
list	**list**	Displays text with tabs and the end of lines marked. Tabs are displayed as **andI** and the end of lines as **$**. The default is **nolist**.
mesg	**mesg**	Causes write permission to be turned off to the terminal while you are in visual mode, if **nomesg** is set. The default is **mesg**.
magic	**magic**	Treats the characters ., [, and * as special characters in scans. In off mode, only the **( )** and **$** retain special meanings; however, special meaning of other characters can still be evoked by preceding the character with a \. The default is **magic**.
modeline	**modeline**	Runs editor command lines found in the first five and the last five lines of the file. An editor command line may be anywhere in a line. To be recognized as a command line, it must contain a space or a tab followed by the string **ex:** or **vi:**. The command line is ended by a second **:**. The editor tries to interpret any data between the first and second **:** as editor commands. The default is **nomodeline**.
novice	**novice**	A feature of **viedit** which makes it a line editor. This option cannot be set in **vi** The default is **nonovice**.
number	**nu**	Displays lines prefixed with their line numbers. The default is **nonu**.
optimize	**opt**	Speeds up the operation of terminals that lack cursor-addressing. The default is **noopt**.
paragraphs	**para=**	Defines to **vi** macro names that start paragraphs. The default is **para=IPLPPPQPP LIpplpipnpbp**. Single letter **nroff** macros, such as **.P** must include the space as a quoted character if re-specifying a paragraph.

prompt	**prompt**	Command mode input is prompted with a colon (:). The default is **prompt**.
readonly	readonly	Sets permanent read mode. The default is is **noreadonly**.
redraw	**re**	Simulates a smart work station on a dumb work station. The default is **nore**.
remap	**remap**	If on, macros are repeatedly tried until they are changed. For example, if **o** is mapped to **O** and **O** is mapped to **I**, and then **remap** is set, **o** maps to **I**. But if **noremap** is set, it maps to **O**. The default is **remap**.
report	**report=**	Sets the number of repetitions of a command before a message is displayed. For subcommands that can produce a number of messages, such as global subcommands, the messages are displayed when the command is completed. The default is **report=5**.
scroll	**Pscroll=**	Displays the default number of lines to be scrolled when the user scrolls up or down. The default is **scroll=12**. Changing this with the **set** option has no effect. This feature can be set by entering **n CTRL-d** during the command mode. **CTRL-u** and **CTRL-d** both respond to this change.
sections	**sect=**	Defines to **vi** macro names that start sections. The default is **sect=NHSHH HUuhsh+c**. Single letter **nroff** macros, such as **.P** must include the space as a quoted character if re-specifying a paragraph.
shell	**sh=**	Defines the shell for **!** or **:!** commands. The default is **sh=/bin/sh**.
shiftwidth	**sw=**	Sets the distance for the software tab stops used by **autoindent**, the shift commands (> and <), and the input commands (**Ctrl-D** and **Ctrl-T**) to allow the editor to indent text and move back to a previous indentation. The default is **sw=8**.
showmode	**smd**	If **smd** is turned on, whenever you are in input mode, it is indicated on the bottom line. The default is **noshowmode**. (**smd** if you invoked **vi** as **viedit**.)
showmatch	**sm**	Shows the matching open parenthesis (or open bracket { as you type the close parenthesis) or close bracket }. The default is **nosm**.
slowopen	**slow**	Postpones updating the display during inserts. The default is **slow**.
tabstop	**ts=**	Sets distance between tab stops when a file is displayed. The default is **ts=8**.

taglength	**tl**	Tags are not significant beyond this many characters. A value of zero (the default) means that all characters are significant.
tags	**tags**	A list of files to be used as tag files for the **tag** command. The tag files are searched sequentially for the requested tag. The default is **tags = tags /usr/lib/tags**. Thus, by default, a file named **tags** in the current directory, and a file named **tags** in **/usr/lib** are searched. **/usr/lib/tags** is a master file for the entire system.
term	**term=**	Sets the kind of work station you are using. The default is **term=$TERM** where **$TERM** is the value of the shell variable **TERM**. You cannot set **TERM** directly while you are in **vi**. Set it at the shell level or quit **vi** to **ex** command mode. (:Q), type **set term** and re-enter **vi** (**:vi**).
terse	**terse**	Allows **vi** to display the short form of messages. The default is **noterse**.
timeout	**to**	Sets a time limit of one second on entry of characters. This limit allows the characters in a macro to be entered and processed as separate characters when **timeout** is set. To resume use of the macro, set **notimeout**. The default is **to** (**timeout**).
warn	**warn**	Displays a warning message before the **!** subcommand executes a shell command if this is the first time you have issued a shell command after a given set of changes have been made in the editing buffer but not written to a file. The default is **warn**.
window	**wi=**	Sets the number of lines displayed in one window of text. The default is dependent on the baud rate at which you are operating: 600 baud or less / 8 lines, 1200 baud / 16 lines, higher speeds / full screen minus one.
wrapmargin	**wm=**	Sets the margin for automatic wordwrapping from one line to the next. The default is **wm=0**. A value of zero indicates no wordwrapping.
wrapscan	**ws**	Allows string searches to wrap from the end of the editing buffer to the beginning. The default is **ws**.
writeany	**wa**	Turns off the checks usually made before a **write** command. The default is **nowa**.

1.1.522.3 VEDIT

vedit works much like **vi** but it is for beginning users. When you enter the input state (via the **a**, **A**, **i**, **I**, **o**, **O**, **c**, **C**, **s**, **S,** and **R** subcommands), the bottom right hand portion of the screen verifies this.

The **vedit** command has the following main subcommands: **showmode**, **novice** and**report** (the **vedit** default for **report** is 1).

Note: When you are in **vedit** and enter **:set**, you see the options that are different from **vi**.

1.1.522.4 Defining Macros

If you use a subcommand or sequence of subcommands frequently, you can create a macro that issues the subcommand or sequence when you call a macro. To create a macro, enter the sequence of subcommands into an editing buffer named with an ASCII character. When used, **a - z** overlay the contents of the buffer; **A - Z** append text to the previous contents of the buffer, allowing the building of a macro piece by piece.

To invoke the macro, enter @**x** where **x** is the letter name of the buffer. Enter @@ to repeat the last macro you invoked.

1.1.522.5 Mapping Keys

You can use the **map** command to set a keystroke to a subcommand or a sequence of subcommands. To set a key mapping, enter

```
  :map key subcommand
```

where **key** (lhs) is the key to which you want to assign a subcommand or sequence of subcommands and **subcommand** (rhs) is the subcommand or sequence of subcommands. (The **key** variable can contain Japanese characters.) For example, to set @ to delete lines, enter:

```
  :map  @  dd
```

In this example, @ is the key to which the subcommand is assigned and **dd** is the subcommand.

In the next example, a subcommand sequence is mapped to a key:

```
  :map * {>>
```

The ***** is the key to which the subcommand sequence is assigned and **{>>** is the subcommand sequence. The **{** moves the cursor to the beginning of the paragraph and the **>>** indents the paragraph to the next shiftwidth.

To display the list of the current key mappings (while you are in command mode), enter the command **:map**. You can also remove a key mapping. To remove a key mapping, enter **:unmap string** or **:unmap! string**, where **string** is the string used after the **:map** command to set the key and subcommand sequence. For example to remove key mapping for the previous example:

```
  :unmap *
```

If function keys are defined for your terminal, they can be put in a **map** or **unmap** command by typing **Ctrl-V**, then pressing the desired key. Keys that are infrequently used in editing are useful to define, such as a **Shift**, **Ctrl**, or **Alt** function key with another key or one of the function keys **F0 - F12**.

1.1.522.6 Keeping a Customized Change

When you customize **vi** from the **vi** command line, the customized editor is in effect until you exit the editor. If you want to keep your assignments so you can reuse them, you must put the commands in the file **.exrc**. The editor reads this file each time you call it. When you type the commands in a file, do not type the : (colon) before each command. The : is only required if you are in the editor. Here is an example of an **.exrc** file:

```
  set ai aw
  set wm=5
  map  @  dd
```

Flags

-l	Enters **vi** in **LISP** mode. In this mode, **vi** indents appropriately for LISP code and the **(**,**)**, **{**, **}**, **[[**, and **]]** subcommands are modified to act appropriately for LISP.
-r file	Recovers **file** after an editor or system crash. If you do not specify a **file** name, **vi** displays a list of all saved files.
-R	Sets the **readonly** option to protect the **file** against overwriting.
-t tag	Edits the file containing the **tag** and positions the editor at its definition.
-wnum	Sets the default window size to **num**. This is useful when you use the editor over a low speed line.
+subcmd	Performs the **ex** subcommand before editing begins. If you do not specify **subcmd**, the cursor is placed on the last line of the file.
-ynum	Allocates **num** number of lines at startup.

Subcommands

In the following lists, **<ESC>** stands for pressing the ESCAPE key instead of pressing the **Enter** key.

1.1.522.7 General Subcommand Syntax

[**named_buffer**] [**operator**] [**number**] **object**

Surrounding square brackets indicate optional items:

[**named_buffer**]	A temporary text storage area in memory.
[**operator**]	Specifies the subcommand or action; tells **vi** what to do.
[**number**]	A whole decimal value that specifies either the extent of the action, or a line address.
object	Specifies what to act on. This can be a text object (a character, word, sentence, paragraph, section, character string) or a text position (a line, position in the current line, screen position).

1.1.522.8 Counts before Subcommands

You may prefix many subcommands with a number. **vi** interprets this number in one of the following ways:

1. Go to line **number**:

   ```
   5G
   10z
   ```

2. Go to column **number**:

   ```
   25|
   ```

3. Scroll **number** lines:

   ```
   10Ctrl-D
   10Ctrl-U
   ```

1.1.522.9 Subcommands for Moving within the File

There are many commands that you can use to move within a file. They can be entered while **vi** is in the command state.

Movements within a Line

or **h**	Moves the cursor one character to the left.
or **j**	Moves the cursor down one line (but it remains in the same column).
or **k**	Moves the cursor up one line (but it remains in the same column).
or **l**	Moves the cursor one character to the right.

Character Positioning within a Line

^	Moves the cursor to the first nonblank character.
0	Moves the cursor to the beginning of the line.
$	Moves the cursor to the end of the line.
fx	Moves the cursor to the next **x** character.
Fx	Moves the cursor to the last **x** character.
tx	Moves the cursor to one column before the next **x** character.
Tx	Moves the cursor to one column after the last **x** character.
;	Repeat the last **f**, **F**, **t**, or **T** subcommand.
,	Repeat the last **f**, **F**, **t**, or **T** subcommand in the opposition direction.
num|	Moves the cursor to the specified column.

Words, Sentences, Paragraphs

l	Moves the cursor to the next character (includes punctuation as words).
w	Moves the cursor to the next word (includes punctuation as words).
b	Moves the cursor to the previous word (includes punctuation as words).
e	Moves the cursor to the end of the word (includes punctuation as words).
W	Moves the cursor to the next word (ignores punctuation).
B	Moves the cursor to the previous word (ignores punctuation).
E	Moves the cursor to the end of the word (includes punctuation as part of the current word).

Line Positioning

H	Moves the cursor to the top line on the screen.
L	Moves the cursor to the last line on the screen.
M	Moves the cursor to the middle line on the screen.
+	Moves the cursor to the next line at its first nonblank character.
-	Moves the cursor to the previous line at its first nonblank character.
Enter	Moves the cursor to the next line at its first nonblank character.

Scrolling

Ctrl-U	Scrolls up one half screen.
Ctrl-D	Scrolls down one half screen.
Ctrl-F	Scrolls forward one screen.
Ctrl-B	Scrolls backward one screen.

Searching for Patterns

[num]G	Places the cursor at line number **num** or to the last line if **num** is not specified.
/pattern	Places the cursor at the next line containing **pattern**.
?pattern	Places the cursor at the next previous line containing **pattern**.
n	Repeats last search for **pattern** in the same direction.
N	Repeats last search for **pattern** in the opposite direction.
/pattern/+num	Places the cursor at the **num**th line after the line matching **pattern**.
?pattern?-num	Places the cursor at the **num**th line before the line matching **pattern**.
%	Finds the parentheses or brace that matches the one at the current cursor position.

The **pattern** can contain Japanese characters.

Moving to Sentences, Paragraphs, or Sections

]]	Places the cursor at next section (or function if you are in the LISP mode).
[[	Places the cursor at previous section (or function if you are in the LISP mode).
(	Places the cursor at the beginning of the previous sentence (or the previous s-expression if you are in the LISP mode).

)	Places the cursor at the beginning of the next sentence (or the next s-expression if you are in the LISP mode).
{	Places the cursor at the beginning of the previous paragraph (or at the next list if you are in the LISP mode).
}	Places the cursor at the beginning of the next the paragraph, at the next list if you are in the LISP mode.

Marking and Returning

``	Moves the cursor to the previous location off current line.
''	Moves cursor to the beginning of the line containing the previous location off the current line.
mx	Marks the current position with letter **x**.
"x	Moves cursor to mark **x**.
'x	Moves cursor to the beginning of the line containing mark **x**.

The **x** variable must be an ASCII character.

Adjusting the Screen

Ctrl-L	Clears and redraws the screen.
Ctrl-R	Redraws the screen and eliminates blank lines marked with a @.
z	Redraws the screen with the current line at the top of the screen.
z-	Redraws the screen with the current line at the bottom of the screen.
z.	Redraws the screen with the current line at the center of the screen.
/pattern/z-	Redraws the screen with the line containing **pattern** at the bottom. The **pattern** can contain Japanese characters.
znum.	Makes the window **num** lines long.
Ctrl-E	Scrolls the window down 1 line.
Ctrl-Y	Scrolls the window up 1 line.

1.1.522.10 Subcommands for Editing

Use the following subcommands to edit your text. Those subcommands that do not have an ***** (asterisk) following them enter in the input state. You return to the command state by pressing the **Esc** key. These subcommands affect the text relative to the current cursor position.

Editing the File

atext	Inserts **text** after the cursor.
Atext	Adds **text** to the end of the line.
C	Changes rest of line (**c$**).
numcc	Changes lines.
numce	Changes text to the end of **n** words.
ch	Changes characters before cursor.
numcl	Changes characters (**s**).
numcw	Changes words from cursor position; can be used to change word endings, etc.
numcwtext	Changes words to **text**.
<Enter>	Split line at cursor position.
numdd *	Deletes lines.
numde *	Deletes text to the end of **n** words.
dh *	Deletes characters before the cursor (**X**).
numdl *	Deletes characters.
numdw *	Deletes words from cursor position; can be used to delete incorrect word endings.
itext	Inserts **text** before the cursor. The **text** can contain Japanese characters.
Itext	Inserts **text** before the first nonblank character in the line. The **text** can contain Japanese characters.
J *	Joins lines.
o	Adds an empty line below the current line.
O	Adds an empty line above the current line.
numrx *	Replaces the current character with **x**. (Commands followed by ***** do not enter the input state.)
Rtext	Overwrites characters with **text**.
nums *	Substitutes characters (**cl**).

S *	Substitutes lines (**cc**).
u *	Undoes the previous change.
numx *	Deletes characters (**dl**).
X *	Deletes characters before cursor (**dh**).
numyh *	Yanks characters before cursor into the **undo** buffer.
numyl *	Yanks characters (s) into the **undo** buffer.
numyw *	Yanks words from cursor position into the **undo** buffer.
numyy *	Yanks lines into the **undo** buffer.
<< *	Shifts one line to the left.
<L	Shifts all lines from the cursor to the end of the screen to the left.
>> *	Shifts one line to the right.
>L	Shifts all lines from the cursor to the end of the screen to the right.
~	Changes letters at cursor to opposite case.
! *	Indents for LISP.

Corrections during Insert
Use the following commands only while in the input state; they have different meanings in the command state.

Ctrl-H	Erases last character.
Ctrl-W	Erases last word.
\	Quotes the erase and kill characters.
<ESC>	Ends insertion, back to command state.
Ctrl-?	Interrupts, terminates insert or **Ctrl-D**.
Ctrl-D	Goes back to previous **autoindent** stop.
^Ctrl-D	Ends **autoindent** for this line only.
0Ctrl-D	Moves cursor back to left margin.
Ctrl-V	Enters nonprinting character.

Moving Text

p	Puts back text in the **undo** buffer after the cursor.
P	Puts back text in the **undo** buffer before the cursor.
"xp	Puts back text from the buffer **x**.

"xd	Deletes text into the buffer **x**.
y	Places the object that follows (for example, **w** for word) in the **undo** buffer.
"xy	Places the object that follows in the **x** buffer.
Y	Places the line in the **undo** buffer.

The **x** variable must be an ASCII character.

Restoring and Repeating Changes

u	Undoes the last change.
U	Restores the current line.
.	Repeats the last change.
"n p	Retrieves the **n**th last delete.

1.1.522.11 Interrupting, Cancelling, and Exiting vi

Q	Enter **ex** editor in command state.
ZZ	Exits **vi**, saving changes.
:q	Quits **vi**. If you have changed the contents of the editing buffer, **vi** displays a warning message and does not quit.
:q!	Quits **vi**, discarding the editing buffer.
:sh	Runs a shell. You can return to **vi** by pressing **Ctrl-D**.
:!cmd	Runs **cmd**, then returns.
:!!	Repeats the last **:!cmd** command.
n!!cmd	Executes shell command **cmd** and replaces the number of lines specified by **n** with the output of **cmd**. If **n** is not specified, the default is 1. If **cmd** expects standard input, the lines specified are used as input. Thus the command **!sort** can sort a paragraph.
n!obj cmd	Executes shell command **cmd** and replaces **n** with output of **cmd**. If **n** is not specified, the default is 1. If **cmd** expects standard input, the lines or **obj** specified is used as input.
<ESC>	Ends insert or ends an incomplete subcommand.
Ctrl-L	Redisplays a screen.
Ctrl-R	Redisplays the screen if **Ctrl-L** is the key.
Ctrl-?	Interrupts a subcommand.

1.1.522.12 File Manipulation

:e file	Edits **file**.
:e!	Re-edits the current file and discards all changes.
:e +file	Edits **file** starting at the end.
:e +num	Edits **file** starting at line **num**.
:e #	Edits the alternate file. The alternate file is usually the the previous current file name. However if changes are pending on the current file when a new file is called, the new file becomes the alternate file. This subcommand is the same as **Ctrl and**.
:n	Edits next file in the list entered on the command line.
:n files	Specifies new list of files to edit. The file names can contain Japanese characters.
:r file	Reads the file into the editing buffer by adding new lines below the current cursor position.
:r !cmd	Runs the shell command **cmd** and places its output in the file by adding new lines below the current cursor position.
:ta tag	Edits a file containing **tag** at the location of **tag**. The **tag** must contain ASCII characters.
:w	Writes the editing buffer contents to the original file.
:w file	Writes the editing buffer contents to the named **file**.
:w! file	Overwrites **file** with the editing buffer contents.
Ctrl-G	Shows current file name and line.
Ctrl and	Edits the alternate file. The alternate file is usually the the previous current file name. However if changes are pending on the current file when a new file is called, the new file becomes the alternate file. This subcommand is the same as **:e #**.
Ctrl]	Finds the word at the cursor in the tags file and edits the indicated file, placing the cursor at the tag. With **noaw** set, this only moves within the current file if changes have been made to the edit buffer. A warning message is posted if the desired tag is in another file.

Related Information

See the following commands: ctags, "ed, red" in topic 1.1.147 and "ex" in topic 1.1.158.

1.1.523 vipw

Purpose
Edits the password file.

Syntax

vipw ---¦

Description

The **vipw** command edits the password file while setting the appropriate locks and does any necessary processing after the password file is unlocked. If the password file is already being edited, you are told to try again later. The **vi** editor is used unless the environment variable **EDITOR** indicates an alternate editor. If it cannot find **vi**, it tries to find the **ed** editor. The user is notified that **ed** is used and what the command is to get out of **ed**. The **vipw** command performs consistency checks on the password entry for **root** and does not allow a password file with an incorrect root entry to be installed. Only ASCII characters can be entered in the password file.

Files

/etc/ptmp
/etc/passwd
/etc/passwd.pag
/etc/passwd.dir

Related Information

See the following commands: "passwd, chfn, chsh" in topic 1.1.312, "adduser, users" in topic 1.1.15, and "mkpasswd" in topic 1.1.272.

1.1.524 vmh

Purpose
Invokes a visual interface for use with MH commands.

Syntax

```
       +- -prompt (vmh) --+   +----- -vmhproc msh -----+
vmh ---¦                  +---¦        one of          +---¦
       +- -prompt string -+   ¦ +-------------------+ ¦
                              +-¦ -vmhproc cmdstring +-+
                                ¦ -novmhproc         ¦
                                +-------------------+

vmh --- -help ---¦
```

Note: This command does not have MBCS support.

Description

The **vmh** command is used to invoke a visual interface for use with MH commands. This command is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The **vmh** command implements the server side of the MH window management protocol and maintains a split-screen interface to any program which implements the client side of the protocol.

The **vmh** command prompts for commands and sends them to the client side of the protocol. If the command produces a window of more than one screen's worth of output, **vmh** prompts the user for a subcommand.

Note: The **vmh** command can never page more than one screen ahead of itself. For example, if you have a screen size of 10 and your text is 100 lines long, typing "50g" from line 1 gives you lines 10-20 (the next screen).

Subcommands

SPACE	Advances to the next screen.
[*num***] ENTER**	Advances the specified number of lines. The default is one line.
[*num***] y**	Goes back the specified number of lines. The default is one line.
[*num***] d**	Advances ten times the specified number of lines. The default for *num*is 1, for a total of ten lines.
[*num***] u**	Goes back ten times the specified number of lines. The default for *num* is 1, for a total of ten lines.
[*num***] g**	Goes to the specified line.
[*num***] G**	Goes to the end of the window. If *num* is specified, this command acts like **g**.
Ctrl-L	Refreshes the screen.

h	Displays a help message.
q	Ends output.

Flags

-help	Displays help information for the command.
-novmhproc	Runs the default **vmproc** directly, without the window management protocol.
-prompt *string*	Uses the specified string as the prompt.
-vmhproc *cmdstring*	Specifies the program which implements the client side of the window management protocol. The default is **msh**.

Profile Entries

Path:	Specifies your **mhpath**.
mshproc:	Specifies the program used for the MH shell.

Files

$HOME	The user's profile.
$HOME/.mh_profile	The MH user profile.

Related Information

See the MH command "msh" in topic 1.1.281.

See the **mh-profile** file in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.525 vrm2rtfont

Purpose
Converts a standard AIX font file to RT PC rtx font format.

Syntax

vrm2rtfont --- **aixfile** --- **rtxfile** ---¦

Description
Before any of the standard fonts provided with the AIX Operating System can be used with the X-Windows program, they must be converted to the X-Windows' **rtx** font file-format. This command takes an AIX font file as input and converts the file to **rtx** format. This format can be used with X-Windows on PS/2 displays.

Examples
To convert the AIX font file **/etc/vtm/itl1.9x20** to an **rtx** font file:

```
  vrm2rtfont /etc/vtm/itl1.9x20  /usr/lpp/fonts/Itl14.500
```

Note how the output file name conforms to the **rtx** naming convention.

Files

/etc/vtm	Contains AIX fonts.
/usr/lpp/fonts	Contains fonts for other programs.

1.1.526 vs

Purpose
Compiles C, VS Pascal and VS FORTRAN programs.

Syntax

```
      +--------------------+   +--------+   +-----------+
vs ---¦ +----------------+ +---¦ +----+ +---¦ +-------+ +--- filename ---¦
      +-¦ -c      -oname +-+   +-¦ -v +-+   +-¦ d+    +-+               ¦
        ¦ -C      -O     ¦¦      ¦ -# ¦¦      ¦ ename ¦   +-------------+
       ¦¦ -Dname  -p     ¦¦     ¦+----+¦      ¦ g+    ¦
       ¦¦ -E      -P     ¦¦     +------+      ¦ l+    ¦
       ¦¦ -g      -Q!    ¦¦                   ¦ lname ¦
       ¦¦ -G      -S     ¦¦                   ¦ o1+   ¦
       ¦¦ -h      -w     ¦¦                   ¦ o2+   ¦
       ¦¦ -Idir   -X     ¦¦                   ¦ o3+   ¦
       ¦¦ -lkey   -z     ¦¦                   ¦ o4+   ¦
       ¦¦ -ldir          ¦¦                   ¦ v+    ¦
       ¦+----------------+¦                   ¦ w-    ¦
       +------------------+                   +-------+
```

Note: This command is for the PS/2 only.

Description
The **vs** command runs the C, VS Pascal and VS FORTRAN compilers and the assembler. Its syntax is similar to the **cc** command. It accepts files containing C source code, FORTRAN source code, VS Pascal source code, assembler source code, or object code and changes them into a form that the computer system can run. **vs** compiles and assembles source files and then links them with any specified object files, in the order listed on the command line. It puts the resulting executable program in a file named **a.out**.

The **vs** command runs the following programs:

cpp The macro preprocessor.

vsc The first pass of the C compiler.

vspascal
The first pass of the VS Pascal compiler.

vsfort The first pass of the VS FORTRAN compiler.

vspass2
The second pass of the compilers (code generator).

vspass3
The third pass of the compilers (code formatter).

as The assembler.

ld The linkage editor.

On the following page is a schematic of the suffixes accepted and generated by the **vs** command.

```
  .c     .h 1     i      .f          .p        .s      .o    .a
```

```
   ¦      ¦       ¦       ¦          ¦        ¦      ¦    ¦
   +------+       ¦       ¦          ¦        ¦      ¦    ¦
                  ¦       ¦          ¦        ¦      ¦    ¦
                  ¦       ¦          ¦        ¦      ¦    ¦
 +---+            ¦       ¦          ¦        ¦      ¦    ¦
 ¦cpp¦            ¦       ¦          ¦        ¦      ¦    ¦
 +---+            ¦       ¦          ¦        ¦      ¦    ¦
                  ¦       ¦          ¦        ¦      ¦    ¦
                  ¦       ¦          ¦        ¦      ¦    ¦
  .i              ¦       ¦          ¦        ¦      ¦    ¦
   +--------------+       ¦          ¦        ¦      ¦    ¦
                          ¦          ¦        ¦      ¦    ¦
                                                     ¦    ¦
 +---+                +------+  +--------+  +--+     ¦    ¦
 ¦vsc¦                ¦vsfort¦  ¦vspascal¦  ¦as¦     ¦    ¦
 +---+                +------+  +--------+  +--+     ¦    ¦
   ¦                      ¦          ¦        ¦      ¦    ¦
   +---------------------------------+        ¦      ¦    ¦
                          ¦                   ¦      ¦    ¦
                                              ¦      ¦    ¦
                                              ¦      ¦    ¦
                         .i                   ¦      ¦    ¦
                                              ¦      ¦    ¦
                          ¦                   ¦      ¦    ¦
                                              ¦      ¦    ¦
                      +-------+               ¦      ¦    ¦
                      ¦vspass2¦               ¦      ¦    ¦
                      +-------+               ¦      ¦    ¦
                          ¦                   ¦      ¦    ¦
                                              ¦      ¦    ¦
                                              ¦      ¦    ¦
                         .j                   ¦      ¦    ¦
                                              ¦      ¦    ¦
                          ¦                   ¦      ¦    ¦
                                              ¦      ¦    ¦
                      +-------+               ¦      ¦    ¦
                      ¦vspass3¦               ¦      ¦    ¦
                      +-------+               ¦      ¦    ¦
                            +-----------------+      ¦    ¦
                                                     ¦    ¦
                                                     ¦    ¦
                           .o                        ¦    ¦
                            +-----------------------------+
                            ¦


                           +--+
1  If -h option is set.    ¦ld¦
                           +--+


                           a.out (executable)
```

You can replace any or all of these passes with your own versions by editing the **/bin/vs** script.

Subtopics
1.1.526.1 Input File Types
1.1.526.2 Ordinary Operation

1.1.526.1 Input File Types

The **vs** command recognizes and accepts as input the following file types:

file.c	The name of a C language source file should end with **.c**.
file.f	The name of a VS FORTRAN source program should end with **.f**.
file.p	The name of a VS Pascal source file should end with **.p**.
file.s	The name of an assembly language source program should end with **.s**.
file.i	The name of a file that contains preprocessed C source code ends in **.i**.
file.o	The name of an object file ends in **.o**. The **vs** command sends these files to the **ld** command.
file.a	The name of an archive library should end with **.a**.

Flags

The **vs** command recognizes several flags. In addition, flags intended to modify the action of the linkage editor (**ld**), the assembler (**as**), or the preprocessor (**cpp**) may also appear on the **vs** command line. **vs** sends any flags it does not recognize to these commands for processing. The following list includes the most commonly used **cpp** flags (**-D**, **-I**), and **ld** flags (**-l**, **-L**, **-o**). See "as" in topic 1.1.25, "cpp" in topic 1.1.94, and "ld" in topic 1.1.226 for a complete list of additional flags.

Note: If you use the **-l** flag, it must be the last entry on the command line, following any file parameters.

1.1.526.2 Ordinary Operation

-c	Does not send the completed object file to the **ld** command. With this flag, the output of **vs** is a **.o** file for each **.c** or **.s** file.
-C	Copies source file output comments to output file. If omitted, the **cpp** command removes all comments except those found on **cpp** directive lines.
-Dname[=def]	Defines **name** as in a **#define** directive. The default **def** is 1.
-E	Runs the named C source file through only the preprocessor and writes the result to standard output.
-g	Produces additional information for use with the **dbx** command (the symbolic debugger).
-G	Indicates that global variables are volatile. The optimizer makes fewer transformations when you specify this flag. To make a particular variable volatile, add the "volatile" specification to its declaration.
-h	Treats files with the suffix **.h** in the same way as files with the suffix **.c**.
-Idir	Looks first in **dir**, then looks in the directories on the standard list for **#include** files with names that do not begin with / (slash).
-lkey	Searches the specified library file, where **key** selects the file **libkey.a**. **ld** searches for this file in the directory specified by an **-L** flag, then in **/lib** and **/usr/lib**. The **ld** command searches library files in the order in which you list them on the command line.
-Ldir	Looks in **dir** for files specified by **-l** keys. If it does not find the file in **dir**, **ld** searches the standard directories.
-oname	Assigns **name** rather than **a.out** to the output file.
-O	Sends compiler output to the code optimizers.
-p	Prepares the program so that the **prof** command can generate an execution profile. The compiler produces code that counts the number of times each routine is called. If programs are sent to **ld**, the compiler replaces the startup routine with one that calls the **monitor** subroutine at the start (see *AIX Operating System Technical Reference* for a discussion of this subroutine), and writes a **mon.out** file when the program ends normally.
-P	Sends the specified C source file to the macro preprocessor and stores the output in a **.i** file.
-Q!	Disables inlining.
-S	Compiles the specified C programs, storing assembly

language output in a **.s** file.

-w Prevents printing of warning messages.

-X Produces an assembler listing. This is stored in a file that has the same name as the assembler source file but with the extension **.lst** instead of **.s**.

-z Uses the **libm.a** version, or a version specified by the user, of the following transcendental functions:

acos	asin	atan	atan2	cos	exp
log	log10	sin	sqrt	tan	

If this flag is not used, the compiler generates inline instructions for the 80387 math co-processor. For more information on **libm.a**, see **math.h** in *AIX Operating System Technical Reference*.

1.1.526.3 Debugging

-v	Displays the trace as with **-#** and invokes the programs.
-#	Displays a trace of the actions to be taken (for example, invoking the preprocessor), without actually invoking any programs.

1.1.526.4 Extended Compiler Functions

The following flags are accepted by the C, VS FORTRAN and VS Pascal compilers. See the appropriate compiler **User's Guide** for additional compiler-specific flags.

d+	Produces additional information for use with the **dis** command (the disassembler).
ename	Produces an error listing and writes it to the file **name**.
g+	Equivalent to the **-g** option.
l+	Produces a source listing and writes it to standard output.
lname	Produces a source listing and writes it to the file **name**.
o1+	Uses optimization level 1.
o2+	Uses optimization level 2.
o3+	Uses optimization level 3.
o4+	Uses optimization level 4, equivalent to the **-O** option.
v+	Generates information on the progress compilation.
w-	Equivalent to the **-w** option.

Examples

1. To compile and link a VS FORTRAN program, creating an executable **a.out** file:

   ```
   vs pgm.f
   ```

2. To compile a VS Pascal program, producing an object file to be linked later:

   ```
   vs -c pgm.p
   ```

   This compiles **pgm.p** and produces an object file named **pgm.o**.

3. To view the output of the macro preprocessor:

   ```
   vs -P -C pgm.c
   ```

   This creates a file named **pgm.i** that contains the preprocessed program text including comments. **vs** passes the **-P** and **-C** flags to the preprocessor. See "cpp" in topic 1.1.94 for more details about them.

4. To predefine macro identifiers:

   ```
   vs -DBUFFERSIZE=512 -DDEBUG pgm.c
   ```

   This assigns **BUFFERSIZE** the value **512** and **DEBUG** the value **1** before preprocessing. **vs** passes the **-D** flag to the preprocessor.

5. To use **#include** files located in nonstandard directories:

```
vs -I/u/tom/include pgm.c
```

This looks in the directory that contains **pgm.c** for the **#include** files with names enclosed in double quotes (**" "**), then in **/u/tom/include**, and then in the standard directories. It looks in **/u/tom/include** for **#include** file names enclosed in angle brackets (**< >**), then in the standard directories. **vs** passes the **-I** flag to the preprocessor.

6. To optimize the object code and produce an assembler listing:

```
vs -S -O pgm.c
```

This uses the optimizing compiler (**-O** is minus, capital oh), and produces an assembler listing in a file named **pgm.s** (**-S**).

Files

file.c	C source file.
file.s	Assembler file.
file.f	VS FORTRAN source file.
file.p	VS Pascal source file.
file.o	Object file.
a.out	Linked output.
/lib/cpp	Preprocessor.
/lib/vsc	C compiler first pass.
/lib/vsfort	VS FORTRAN compiler first pass.
/lib/vspascal	VS Pascal compiler first pass.
/lib/vspass2	Compiler second pass.
/lib/vspass3	Compiler third pass.
/bin/as	Assembler.
/bin/dis	Disassembler.
/bin/ld	Linkage editor.
/lib/crt1.o	Runtime startoff.
/lib/mcrt1.o	Runtime startoff for profiling.
/lib/libc.a	Standard library.
/lib/libm.a	Standard math library.
/lib/libvsfor.a	VS FORTRAN runtime library.
/lib/libvssys.a	VS Pascal and VS FORTRAN runtime library.
/usr/lib/msg/vscctmsg.inc	C compiler message file.
/usr/lib/msg/vsdismsg.inc	Disassembler message file.
/usr/lib/msg/vsfctmsg.inc	VS FORTRAN compiler message file.
/usr/lib/msg/vsfrtmsg.inc	VS FORTRAN runtime message file.
/usr/lib/msg/vspctmsg.inc	VS Pascal compiler message file.
/usr/lib/msg/vsprtmsg.inc	VS Pascal runtime message file.
/usr/include	Standard directory for #include files.
/usr/tmp/vs*	Temporary.

Related Information

See the following commands: "as" in topic 1.1.25, "cc" in topic 1.1.52, "ld" in topic 1.1.226, and "cpp" in topic 1.1.94.

1.1.527 vucp

Purpose
Receives NETDATA files.

Syntax

```
       +--------+   +------+   +------------------+
vucp ---¦ +----+ +---¦      +---¦        +--------+ +---¦
       +-¦ -b +-+   +- -r -+   +- file -¦        +-+
         +----+                         +- dest -+

       +------+   +--------+
vucp ---¦      ¦   ¦        +---¦
       +- -q -+   +- file -+
```

Note: This command is for the System/370 only.

Description
The **vucp** command is used to interpret and translate (if required) a NETDATA file. The result is directed to **dest**. The actual transmitted spool file is left unchanged in your **netfile** directory. This allows you to process the file using different translate tables (this can be helpful if an error occurs during the reading in of the file). Because the file is left in the **netfile** directory, you are responsible for deleting that file. You can do this, for example, by using the AIX/370 **rm** command.

If a file was sent via the **uvcp** command, the specified translation option is included in the transmitted file. This option determines **vucp** command processing, unless explicitly overridden by you through the use of the translation options. The conditions are:

The translation table specified via the environment variable **NLIN** is used to translate the file from EBCDIC to ASCII.

The **-b** option was used by the **uvcp** command. No translation occurs.

If acknowledgement was requested and the **vucp** command is invoked without the **-q** option, an acknowledgement is returned to the sender.

Defaults for the vucp Command
The defaults for the **vucp** command are:

Translation to ASCII by using the translation table specified in you environment variable **NLIN**.

If the file was transmitted using the **uvcp** command, the specified translate option is included in the transmitted file and determines the translation default.

Note: If you have selected a language that supports multibyte characters, the environment variable **NLIN** is not used. Translation to one of the supported file or network code sets is done by checking the current locale. For information on supported code sets, see "iconv" in topic 1.1.204.

No overwrite of an existing file with the same name

Standard input is used, if the file parameter is omitted

Standard output is used, if the **dest** parameter is omitted.

Restrictions

The NETDATA protocol allows multiple files to be included in on transmission. The **vucp** command only reads the first file of such a transmission. The **uvcp** sends multiple files one by one.

If you try to use the **vucp** command for a non-NETDATA file, an error message such as the following is displayed:

```
UVC107 vucp: INVALID INPUT FORMAT
```

You must interpret the correct format of the file.

Flags

-b
Specifies that a binary file is being received. New-line characters are not inserted in the output file and no translation is performed.

-r
Specifies that the newly created output file is to replace any existing file with the same name. Unless this option is specified, processing stops and you receive this error message if a file with the same name as the output file exists:

```
UVC201 vucp: FILE file name ALREADY EXISTS, SPECIFY -r
```

The existing file is not overwritten.

file
Identifies the file name of the file to be received. If a qualified path name is specified for the file, it is used. Otherwise, the current working directory is searched. If the file is not found there, the **$HOME/netfile** directory is searched. You can have more than one file waiting and you must specify which file you want. You must have read access to the file.

dest
Specifies where the received file is to be placed. If **dest** is a directory, it is used to store the file with the file name contained in the transferred file's data record. If **dest** is an absolute path name, the directory and the file name contained in that path name is used. If **dest** is not specified, the file is directed to the standard output.

-q
Specifies that the file name, the sender, the node, and the date and time of sending is displayed. The type of file is also displayed if the file was sent via the **uvcp** command. There are two types of files:

text	file was translated using the default translate table.
binary	file was not translated or was translated using a translate table other than the default table.

When using this option, the file is not actually received; therefore, acknowledgement (if requested) is not sent.

Files

/usr/lib/nls/nlin Directory containing the EBCDIC/ASCII translation facility.

Related Information

See the following commands: "uvcp" in topic 1.1.517, "iconv" in topic 1.1.204, and "rdrdaemon" in topic 1.1.361.

1.1.528 w

Purpose
Prints who is on the system and what they are doing.

Syntax

```
     +-------------------------+
w ---¦   +----+   +--------+   +---¦
     +---¦ -h +---¦        +---+
         ¦ -l ¦ ¦ +- user -
       ¦ ¦ -s ¦ ¦
       ¦ +----+ ¦
       +--------+
```

Description
The **w** command prints a summary of the current activity on the system, including what each user is doing. The heading line shows the current time of day, how long the system has been up, the number of users logged into the system, and the load averages. The load average numbers give the number of jobs in the run queue averaged over 1, 5 and 15 minutes.

The fields output are: the users login name, the name of the tty the user is on, the time of day the user logged on, the number of minutes since the user last typed anything, the CPU time used by all processes and their children on that terminal, the CPU time used by the currently active processes, the name and arguments of the current process.

If **user** is included, the output is restricted to that user.

Flags

-h	Suppresses the heading.
-s	Specifies short output form. In the short form, the tty is abbreviated, the login time and cpu times are left off, as are the arguments to commands.
-l	Specifies the long output, which is the default.

Files

/etc/utmp
/dev/kmem

Related Information

See the following commands: "who" in topic 1.1.537 and "ps" in topic 1.1.337.

1.1.529 wall

Purpose
Writes a message to all logged-in users.

Syntax

```
        +------+
wall ---¦      +---¦
        +- -L -+
```

Description

The **wall** command reads a message from standard input until it reaches an end-of-file character. It then sends the message to all logged-in users preceded by the following heading:

  Broadcast Message from **user (/machinename/dev/tty##) Day Mon dd hh:mm**

To override any protections other users have set up, you must be operating with superuser authority. Typically, root uses **wall** to warn all users of an impending system shutdown.

If you are writing a message to all logged-in users on a system on which users have selected different character code sets for displaying text in different languages, the message should be written using ASCII characters so that it displays consistently for all users.

Flag

-L	Message is not to be broadcast to users logged in to remote cluster sites.

Files

/dev/tty*	Default terminal devices.

Related Information

See the following commands: "mesg" in topic 1.1.262, "su" in topic 1.1.449, and "write" in topic 1.1.541.

1.1.530 wc

Purpose
Counts the number of lines, words and characters in a file.

Syntax

```
      +----- -cwl ------+   +--------+
wc ---¦     one of      +---¦        +---¦
      ¦ +-------------+ ¦   +- file -+
      +-¦ -c -cl -cwl +-+         ¦
        ¦ -l -cw      ¦      +------+
        ¦ -w -lw      ¦
        +-------------+
```

Description

The **wc** command counts the number of lines, words, or characters in **file** or in the standard input if you do not specify any **file**s. It writes the results to standard output. It also keeps a total count for all named files. A word is defined as a string of characters delimited by spaces, tabs, or new-line characters. **wc** counts lines, words, and characters by default.

When you specify more than one **file** on the command line, **wc** displays the name of the file along with the counts.

Flags

-c Counts bytes.
-l Counts lines.
-w Counts words.

Examples

1. To display the line, word, and character counts of a file:

   ```
   wc chap1
   ```

   This displays the number of lines, words, and characters in the file **chap1**.

2. To display only character and word counts:

   ```
   wc -cw chap*
   ```

   This displays the number of characters and words in each file whose name starts with **chap**, and displays the totals.

1.1.531 what

Purpose
Displays identifying information in files.

Syntax

```
        +------+
what ---¦      +-- file --¦
        +- -s -+        ¦
                 +------+
```

Description
The **what** command searches the named **file**s for all occurrences of the pattern that **get** substitutes for the %**Z**% keyletter (see "Identification Keywords" in topic 1.1.186.2). By convention, the value substituted is **@(#)**. **what** writes to standard output whatever follows the pattern up to but not including the first double quotation mark (**"**), greater than symbol (>), new-line character, backslash (\), or null character.

The **what** command is intended for use in conjunction with the **get** command, which automatically inserts the identifying information. You can also use **what** on files where the information is inserted manually.

Flags

-s Searches for only the first occurrence of **@(#)**.

Examples

Suppose that the file **test.c** contains a C program that includes the line:

```
  char ident[ ] = "@(#)Test Program";
```

If you compile **test.c** to produce **test.o** and **a.out**, the command:

```
  what test.c test.o a.out
```

displays:

```
  test.c:
          Test Program
  test.o:
          Test Program
  a.out:
          Test Program
```

Related Information

See the following commands: "get" in topic 1.1.186, and "sccshelp" in topic 1.1.411.

See the **sccsfile** file in *AIX Operating System Technical Reference*.

See the discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

1.1.532 whatis

Purpose
Describes a command.

Syntax

```
whatis --- command ---¦
```

Description

The **whatis** command looks up a given command and displays the header line from the manual section. You can then run the **man** command to get more information. If the header line starts **name(section**), you can enter **man section name** to get the documentation for it. For example, entering **whatis ed** displays the header line for the **ed** command; entering **man ed** displays the manual page.

Note: The **whatis** command is the same as using the **man** command with the **-f** option.

Files

/usr/man/whatis	Data base.

Related Information

See the following commands: "man" in topic 1.1.258 and "catman" in topic 1.1.50.

1.1.533 whatnow

Purpose
Invokes a prompting interface for draft disposition.

Syntax

```
           +-----------------------------------------------------------------
           ¦         one of
           ¦   +----------------+
whatnow ---¦   ¦ file            ¦
           ¦ +-¦ -nodraftfolder +---------------------------------------------
           ¦ ¦ +----------------+
           +-¦                   one of
             ¦ +-------------------------------------+   +------- cur -------+
             +-¦ -draftfolder +folder -draftmessage +---¦      one of       +-
               ¦ -draftfolder +folder                 ¦   ¦ +----------------+ ¦
               ¦ -draftmessage                        ¦   +-¦ num      cur  +-+
               +-------------------------------------+     ¦ sequence .      ¦
                                                            ¦ first    next   ¦
                                                            ¦ prev     last   ¦
                                                            +----------------+

    +-----------------+   +- -prompt "What Now?" -+
 ---¦     one of      +---¦                        +---¦
    ¦ +-------------+ ¦   +--- -prompt string ----+
    +-¦ -editor cmd +-+
      ¦ -noedit     ¦
      +-------------+

whatnow --- -help ---¦
```

Note: This command does not have MBCS support.

Description

The **whatnow** command is the default program used by **comp**, **dist**, **forw**, and **repl** to prompt you for the disposition of messages. **whatnow** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

By default, **whatnow** invokes an editor and places the current draft message (or **file**) in the editing session. When you exit the editing session, **whatnow** prompts: **What now?** You can specify any of the **whatnow** subcommands, or you can press **Enter** to see a list of the subcommands. These subcommands enable you to re-edit the message, direct the disposition of the message, or end the processing of the **whatnow** command. The **whatnow** command continues to prompt you for subcommands until you specify **quit**.

Subcommands

display [*flags***]**	Displays the message being acted upon (redistributed or replied to). For *flags*, you can specify any flag that is valid for the command serving as the **lproc**. (You can set a default **lproc:** entry in **$HOME/.mh_profile**.) If you specify any flags that are not valid for **lproc**, **whatnow** does not pass the path name of

	the draft to the **lproc**.
edit [*cmdstring***]**	Re-edits the message using the specified editor. You can specify the editor and any valid flags to that editor. If you do not specify an editor, **whatnow** selects a default editor according to the information supplied in the MH profiles. You can define a default editor for re-editing in **$HOME/.mh_profile**. If an editor is not defined for re-editing in your **.mh_profile**, **whatnow** invokes the editor used in the previous editing session.
list [*flags***]**	Displays the draft. For *flags*, you can specify any flag that is valid for the command serving as the **lproc**. (You can set a default **lproc:** entry in **$HOME/.mh_profile**.) If you specify any flags that are not valid for **lproc**, **whatnow** does not pass the path name of the draft to **lproc**.
push [*flags***]**	Sends the message in the background. You can specify any valid flag for the **send** command.
quit [-delete]	Ends the **whatnow** session. If you specify **-delete**, **whatnow** deletes the draft. Otherwise, **whatnow** stores the draft.
refile [*flags***] +***folder*	Files the draft in the specified folder and supplies a new draft having the previously specified form. For *flags*, you can specify any flag that is valid for the command serving as the **fileproc**. (You can set a default **fileproc:** entry in **$HOME/.mh_profile**.)
send [*flags***]**	Sends the message. You can specify any valid flags for the **send** command.
whom [*flags***]**	Displays the addresses to which the message would be sent. You can specify any valid flags for the **whom** command.

Flags

-draftfolder +*folder* Places the draft message in the specified folder. If you do not specify this flag, **whatnow** selects a default draft folder according to the information supplied in the MH profiles. You can define a default draft folder in **$HOME/.mh_profile**. If **-draftfolder +***folder* is followed by *msg*, *msg*represents the **-draftmessage** attribute.

-draftmessage *msg* Specifies the draft message. You can use one of the following message references as *msg*:

num	*sequence*	**first**
prev	**cur**	**.**
next	**last**	

The default draft message is **cur**.

-editor *cmd*	Specifies that *cmd* is the initial editor for composing or revising the message. If you do not specify this flag, **whatnow** selects a default editor or suppresses the initial edit, according to the information supplied in the MH profiles. You can define a default initial editor in **$HOME/.mh_profile**.
-help	Displays help information for the command.
-nodraftfolder	Places the draft in the file **user_mh_directory/draft**.
-noedit	Suppresses the initial edit.
-prompt *string*	Uses *string* as the prompt. The default string is **"What now?"**

Profile Entries

Draft-Folder:	Sets your default folder for drafts.
Editor:	Sets your default initial editor.
fileproc:	Specifies the program used to refile messages.
lasteditor-next:	Specifies the editor used after exiting **lasteditor**.
lproc:	Specifies the program used to list the contents of a message.
Path:	Specifies your **user_mh_directory**.
sendproc:	Specifies the program used to send messages.
whomproc:	Specifies the program used to determine the users to whom a message would be sent.

Files

.profile	The user's profile.
$HOME/.mh_profile	MH user profile.
user_mh_directory/draft	Draft file.

Related Information

See other MH commands: "comp" in topic 1.1.85, "dist" in topic 1.1.131, "forw" in topic 1.1.174, "prompter" in topic 1.1.334, "refile" in topic 1.1.366, "repl" in topic 1.1.369, "rmm" in topic 1.1.380, "scan" in topic 1.1.409, "send" in topic 1.1.416, "whom" in topic 1.1.539.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.534 where

Purpose
Displays file storage locations.

Syntax

```
         +--------+  +-----+
where ---¦ +---+  +--¦     +--- file ---¦
         +-¦-v +--+  +- - -+          ¦
           ¦-p ¦¦           +---------+
          ¦+---+¦
          +-----+
```

Description

The **where** command takes a list of files and prints each name, status information, and a list of the sites or CPU types where the file is stored. There are three types of file systems with different output for each:

1. For files in non-replicated file systems, a single site name is listed.

2. Without the **-v** (verbose) option, files in system-replicated file systems show site types where the files are stored plus any additional primary or backbone sites. With the **-v** option, all storage sites are listed vertically under the file information.

3. Files in user-replicated file systems always have storage sites displayed vertically.

The first column of output lists status information about the file. The format is:

```
  [nr|sr|ur] [m|r|e|?|x]
```

where the flags have the following meanings:

nr	File stored in a non-replicated file system.
sr	File stored in a system-replicated file system.
ur	File stored in a user-replicated file system.
m	File exists and is modifiable (subject to permissions).
r	File exists and is read-only (subject to permissions).
e	File exists but no copy of it is available.
?	File may exist but is unavailable due to unmounted file system or unavailable storage site.
x	File does not exist. This information is returned only if the **-p** flag is used.

The next column of output is the file name, which the **where** command converts to an absolute path if necessary. When a file does not exist or is unavailable and contains symbolic links, **where** returns a name different than the one specified even if it is given as an absolute path. Under these circumstances, **where** prints an expanded path name using the contents of the symbolic links; that is, it is an absolute path name where no available component of the path is a symbolic link. For example, if **/u** is a symbolic link to **/uy**, **/uy/user** is returned for **/u/user.**

The file name can have a **[SYM]** after it if the final component of the file path is a symbolic link. Symbolic links in the last component are not followed. Also, a **[MP]** can follow the path name if the named directory is the mount point for a file system.

For system-replicated or non-replicated files, the last column of output is the comma-separated list of sites and CPU types. For user-replicated files or system-replicated files with the **-v** flag on, the sites are printed one per line below the file name. Information about sites that are not in the current partition or file systems that are not mounted is marked with a trailing question mark.

Device files indicate the storage sites for their inodes, not necessarily where the the device is attached. See "ls, lf, lr" in topic 1.1.252 for more information.

Note: The **where** command is sensitive to incorrect information in the **/etc/fsmap** file, particularly disk flags.

Flags

-p Lists potential storage sites for files. The **where** command permits you to specify non-existent files only if you use this option.

-v Verbose option; besides listing sites, displays site number and, for replicated file systems, indicates whether a site is primary (pr) or backbone (bb).

- Signals that the next argument is a file name beginning with a -.

Files

/etc/site Contains site number/site name correspondences.
/etc/fsmap Searched for information on file systems of site which down.
/etc/fstore Contains fstore/cpu name correspondences.

Related Information

See the following commands: "ls, lf, lr" in topic 1.1.252, "store" in topic 1.1.443, and "chfstore" in topic 1.1.64.

See **fsmap**, **fstore**, **fs**, **site**, **dustatd** and **symlink** in the *AIX Operating System Technical Reference.*

1.1.535 whereis

Purpose
Locates source, binary and/or manual for program.

Syntax

```
           +--------+   +---------------------+
whereis ---¦ +----+ +---¦   +--------+        +--- name ---¦
           +-¦ -s +-+   +---¦ -S dir +--- -f -+
             ¦ -b ¦         ¦ -B dir ¦
            ¦¦ -m ¦¦      ¦ ¦ -M dir ¦ ¦
            ¦¦ -u ¦¦      ¦ +--------+ ¦
            ¦+----+¦      +------------+
            +------+
```

Description

The **whereis** command locates source/binary and manuals sections for specified files. The supplied names are first stripped of leading path name components and any (single) trailing extension of the form ".ext", for example ".c". Prefixes of "s." resulting from use of source code control are also dealt with. The **whereis** command then attempts to locate the desired program in a list of standard places. If any of the **-b, -s,** or **-m** flags are given then the **whereis** command searches only for binaries, sources or manual sections respectively (or any two thereof). The **-u** flag may be used to search for unusual entries. A file is said to be unusual if it does not have one entry of each requested type. Thus "whereis -m -u *" asks for those files in the current directory which have no documentation.

Finally, the **-B, -M,** and **-S** flags may be used to change or otherwise limit the places where **whereis** searches. The **-f** file flag is used to terminate the last such directory list and signal the start of file names.

Note: Since the program uses **chdir** to run faster, path names given with the **-M, -S** and **-B** must be full; they must begin with a /.

Examples
The following finds all the files in **/usr/bin** which are not documented in **/usr/man/man1** with source in **/usr/src/cmd**:

```
cd /usr/bin

whereis -u -M /usr/man/man1 -S /usr/src/cmd -f *
```

Files

/usr/src/*
/usr/{doc,man}/*
/lib, /etc, /usg, /usr/{lib,bin,ucb,lpp}

1.1.536 which

Purpose
Locates an executable program using "path."

Syntax

```
which --- file ---¦
                  ¦
       +--------+
```

Description

Each argument is expanded if it is aliased, and the **which** command follows the user's path to locate the first user executable program(s) matching the given file(s). If hidden directories are encountered, they are displayed.

Note: If a given file is not found, **which** exits silently.

Examples

With PATH = :/bin:/usr/bin:/usr/ucb, the command:

```
        which login who which
```

produces:

```
       no login in . /bin /usr/bin /usr/ucb
       /bin/who
       /usr/ucb/which
```

Files

$HOME/.cshrc

Related Information

See the following command: "csh" in topic 1.1.100.

1.1.537 who

Purpose

Identifies the users currently logged in.

Syntax

```
       +----- -s ------+   +- /etc/utmp -+
who ---¦ +--- -a ----+ +---¦             +---¦
       +-¦ +-------+ +-+   +--- file ----+
         +-¦ -b -s +-+
           ¦ -d -t ¦
          ¦¦ -l -T ¦¦
          ¦¦ -p -u ¦¦
          ¦¦ -r -L ¦¦
          ¦¦ -A -H ¦¦
          ¦¦    -q ¦¦
          ¦+-------+¦
          +---------+

who am i ---¦
```

Description

The **who** command with no flags writes to standard output the login name, work station name, and date and time of login for all users currently on the system. **who am i** gives this information only for you.

With flags, **who** can also display the elapsed time since line activity occurred, the process-ID of the command interpreter (shell), logins, logouts, restarts, and changes to the system clock, as well as other processes generated by the **init** process.

The general format of the output of **who** is as follows:

name [**state**] **line time activity pid[location**] [**exit**][**sitename**]

where:

name is the user's login name.
state indicates whether the line is readable by everyone (see the **-T** flag on 1.1.537).
line is the name of the line as found in the directory **/dev**.
time is the time that user logged in.
activity is the hours and minutes since activity last occurred on that user's line. A dot (.) here indicates line activity within the last minute. If the line has been quiet more than 24 hours or has not been used since the last system startup, the entry is marked old.
pid is the process-ID of the user's shell.
location is the location associated with this line as found in file **/etc/ports**.

This file can contain information about where the work station or terminal is located, the telephone number of the dataset, the type of work station if direct-connected, and other related information. For users logged in using **rlogin** or **telnet** (see the *TCP/IP Users' Guide*) the location field also shows the name of the remote host.
exit is the exit status of ended processes (see the **-d** flag on the following page).

sitename is the name of the site to which the user is logged on.

To obtain its information, **who** normally examines **/etc/utmp**. If you specify another **file**, **who** examines the named **file** instead. This **file** is usually **/usr/adm/wtmp**, which contains the history of all logins since the file was last created or **/etc/.ilog**, which contains the history of invalid logins. Only someone operating with superuser authority or a member of the system group can examine **/etc/.ilog**.

Flags

-A	Display user's login name.
-a	Prints information from **/etc/utmp** or the named file in a verbose form.
-b	Indicates the time and date of the most recent system startup. The **LANG** and **LC_TIME** environment variables control the format of the login time and date.
-d	Displays all processes that have expired without being regenerated by **init**. The **exit** field appears for dead processes and contains the termination and exit values (as returned by **wait**) of the dead process. (This flag is useful for determining why a process ended.)
-H	Displays a header above each column.
-l	Lists only work stations or terminals not in use. The **name** field is **LOGIN** in such cases. Other fields are the same as for the **-u** option.
-L	Prints information about the local site only. This option also suppresses the display of the site name for each user.
-p	Lists any active process that is currently active and has been previously generated by **init**.
-q	Displays a compact list of logged on users similar to the **users** command.
-r	Indicates the current **run-level** of the **init** process. Following the run level and date information are three fields that indicate the current state, the number of times that state was previously entered, and the previous state.
-s	Lists only the **name**, **line**, **time** and **host computer** fields. (This is the default; thus, **who** and **who -s** are equivalent.) The **LANG** and **LC_TIME** environment variables control the format of the time.
-t	Indicates the last change to the system clock by the superuser using the **date** command. The **LANG** and **LC_TIME** environment variables control the format of the time.

-T	Displays the **state** of the work station line and indicates who can write to that work station as follows: **+** writable by anyone **-** writable only by the superuser or its owner **?** bad line encountered.
-u	Displays the user name, work station name, login time, line activity, process-ID of each current user, and either the comment from **/etc/inittab**, or the originating host's name, if the user used either **telnet** or **rlogin** to log on to the local host. The **LANG** and **LC_TIME** environment variables control the format of the login time.

Examples

1. To display information about who is using the system:

   ```
   who
   ```

   This lists the user name, work station name, and login time of all users currently using the system.

2. To display your user name:

   ```
   who am i
   ```

   This displays the user name you typed when you logged in, the name of the work station you are using, and the time you logged in. Your login user name may be different from your current user name if you have used the **su** command.

3. To display a history of logins, logouts, system startups, and system shutdowns:

   ```
   who-as  /usr/adm/wtmp
   ```

Files

/etc/utmp	Contains user and accounting information for system users.
/usr/adm/wtmp	History of all logins on system.
/etc/ports	System terminal port definitions.
/etc/.ilog	History of invalid login attempts.

Related Information

See the following commands: "date" in topic 1.1.110, "init, telinit" in topic 1.1.208, "login" in topic 1.1.241, "mesg" in topic 1.1.262, and "su" in topic 1.1.449.

See the **wait** system call and the **ports** and **utmp** files in *AIX Operating System Technical Reference*.

See "Introduction to International Character Support" in *Managing the AIX Operating System*.

1.1.538 whoami

Purpose
Print the effective current user ID.

Syntax

```
whoami ---¦
```

Description

The **whoami** command prints who you are. It works even if you used **su** to change your user name, while **who am i** does not since it uses **/etc/utmp**.

Files

/etc/passwd	Name data base.

Related Information

See the following command: "who" in topic 1.1.537.

1.1.539 whom

Purpose
Lists and verifies the addresses of the proposed recipients of a message.

Syntax

```
       +- -alias /usr/lib/mh/MailAliases -+   +-- -nocheck --+
whom ---¦                                  +---¦    one of    +---
       +---------- -alias file -----------+   ¦ +----------+ ¦
                                 ¦             +-¦ -check   +-+
                 +---------------+               ¦ -nocheck ¦
                                                 +----------+

    +--------------------------------------------------------------+
    ¦         one of                                               ¦
    ¦   +---------------+                                          ¦
 ---¦   ¦ -draft        ¦                                          +---¦
    ¦   ¦ -nodraftfolder¦                                          ¦
    ¦ +-¦ file          +--------------------------------------+ ¦
    ¦ ¦ +---------------+                                      ¦ ¦
    +-¦               one of                                   +-+
      ¦ +-------------------------------------+   +--- cur ---+ ¦
      +-¦ -draftfolder +folder -draftmessage +---¦  one of   +-+
        ¦ -draftfolder +folder                ¦   ¦ +-------+ ¦
        ¦ -draftmessage                       ¦   +-¦ num   +-+
        +-------------------------------------+     ¦ first ¦
                                                    ¦ prev  ¦
                                                    ¦ cur   ¦
                                                    ¦ .     ¦
                                                    ¦ next  ¦
                                                    ¦ last  ¦
                                                    +-------+

whom --- -help ---¦
```

Note: This command does not have MBCS support.

Description

The **whom** command is used to expand the headers of a message into a set of addresses. The **whom** command is also used to verify that those addresses are valid. **whom** is part of the Message Handling (MH) package and can be used with other MH and AIX commands.

The message can reside in a draft folder or in a file. You can use one of the **-draft**, **-draftfolder**, **-draftmessage**, or **nodraftfolder** flags or the **file** argument to specify where the message resides.

If you want to verify the addresses, you must specify the **-check** flag.

Flags

-alias *file*	Specifies that *file* is a mail alias file to be searched for aliases. The default alias file is **/usr/lib/mh/MailAliases**.
-check	Checks to see if the addresses are valid. This flag should be used only if the program used to post

messages is **/usr/lib/mh/post** (as specified in **$HOME/.mh_profile**).

-draft	Uses the header information in the file **user_mh_directory/draft** if it exists.
-draftfolder +*folder*	Uses the header information of the draft message in the specified folder. If you do not specify this flag, **whom** selects a default draft folder according to the information supplied in the MH profiles. You can define a default draft folder in **$HOME/.mh_profile**. If **-draftfolder +***folder* is followed by *msg:*, *msg* represents the **-draftmessage** attribute.
-draftmessage *msg*	Uses the header information in the specified draft message. You can use one of the following message references when specifying *msg*:

num	*sequence*	**first**
prev	**cur**	**.**
next	**last**	

The default draft message is **cur**.

-help	Displays help information for the command.
-nocheck	Does not check to see if the addresses are valid. This is the default.
-nodraftfolder	Undoes the last occurrence of **-draftfolder +***folder*

Profile Entries

Draft-Folder:	Sets your default folder for drafts.
postproc:	Specifies the program used to post messages.

Files

$HOME/.mh_profile	The MH user profile.

Related Information

See other MH commands: "ali" in topic 1.1.17, "post" in topic 1.1.320, "whatnow" in topic 1.1.533.

See the **mh-alias**, **mh-format**, **mh-mail**, and **mh-profile** files in *AIX Operating System Technical Reference*.

See "Overview of the Message Handling Package" in *Managing the AIX Operating System*.

1.1.540 window

Purpose

Window environment.

Syntax

```
          +--------+   +-----------------+   +--------------+
window ---¦ +----+ +---¦                 +---¦              +---¦
          +-¦ -t +-+   +- -e escape-char -+   +- -c command -+
            ¦ -f ¦
           ¦¦ -d ¦¦
           ¦+----+¦
           +------+
```

Note: This command does not have MBCS support.

Description

The **window** command implements a window environment on ASCII terminals and PS/2s not on 3270 terminals.

A window is a rectangular portion of the physical terminal screen associated with a set of processes. Its size and position can be changed by the user any time. Processes communicate with their window in the same way they normally interact with a terminal--through their standard input, output, and diagnostic file descriptors. The window program handles the details of redirecting input and output to and from the windows. At any one time, only one window can receive input from the keyboard, but all windows can simultaneously send output to the display.

Windows can overlap and are framed as necessary. Each window is named by one of the digits "1" to "9". This one-character identifier, as well as a user-definable label string, are displayed with the window on the top edge of its frame. A window can be designated to be in the **foreground**, in which case it is always on top of all normal, non-foreground windows, and can be covered only by other foreground windows. A window need not be completely within the edges of the terminal screen. Thus a large window (possibly larger than the screen) may be positioned to show only a portion of its full size.

Each window has a cursor and a set of control functions. Most intelligent terminal operations such as line and character deletion and insertion are supported. Display modes such as underlining and reverse video are available if they are supported by the terminal. In addition, similar to terminals with multiple pages of memory, each window has a text buffer which can have more lines than the window itself.

Flags

When **window** starts up, the commands (see long commands below) contained in the file **.windowrc** in the user's home directory are executed. If the file does not exist, two equal-sized windows spanning the terminal screen are created by default.

-t Turn on terse mode (see **terse** command below).

-f Fast. Don't perform any startup action.

-d	Ignore **.windowrc.** and create the two default windows instead.
-e escape-char	Set the escape character to **escape-char. Escape-char** can be a single character, or in the form **^X** where **X** is any character, meaning control**X**.
-c command	Execute the **string** command as a long command (see below) before doing anything else.

Process Environment

With each newly created window, a shell program is spawned with its process environment tailored to that window. Its standard input, output, and diagnostic file descriptors are bound to one end of either a pseudo-terminal (**pty** special files in *AIX Technical Reference Vol II*) or a UNIX domain socket (**socketpair** *AIX Technical Reference Vol I*). If a pseudo-terminal is used, then its special characters and modes (see **stty**) are copied from the physical terminal. The TERMINFO variable points to the **terminfo** entry tailored to this window which is created in the user's directory. The **terminfo** entry contains the window's characteristics and information from the physical terminal such as the existence of underline, reverse video, and other display modes and codes produced by any terminal function keys. In addition, the window size attributes of the pseudo-terminal are set to reflect the size of this window, and updated whenever it is changed by the user. In particular, the editor **vi** uses this information to redraw its display.

Note: Processes can be run only from windows that use pseudo-terminals (**pty**). Windows that use **socketpairs** can be used only to display information, such as error messages.

Processing

During normal execution, **window** can be one of two states: conversation mode and command mode. In conversation mode, the terminal's real cursor is placed at the cursor position of a particular window--called the current window--and input from the keyboard is sent to the process in that window. The current window is always on top of all other windows, except those in foreground. In addition, it is set apart by highlighting its identifier and label in reverse video.

Typing **window's** escape character (normally ^P) in conversation mode switches it into command mode. In command mode, the top line of the terminal screen becomes the command prompt window, and **window** interprets input from the keyboard as commands to manipulate windows.

There are two types of commands: short commands are usually one or two key strokes; long commands are strings either typed by the user in the command window (see the ":" command below), or read from a file see **source** below).

Short Window

Below, # represents one of the digits "1" to "9" corresponding to the windows 1 to 9. ^**X** means control-**X**, where **X** is any character. In particular, ^^ is control-^. **Escape** is the escape key, or ^].

#	Select window # as the current window and return to conversation

mode.

% # Select window # but stay in command mode.

^^ Select the previous window and return to conversation mode. This is useful for toggling between two windows.

escape Return to conversation mode.

^P Return to conversation mode and write ^P to the current window. Thus, typing two ^P's in conversation mode sends one to the current window. If the **window** escape is changed to some other character, that character takes the place of ^P here.

? List a short summary of commands

^L Redraw the screen.

q Exit **window**. Confirmation is requested.

^Z Suspend **window**.

w Create a new window. The user is prompted for the position of the upper left and lower right corners of the window. The cursor is placed on the screen and the keys "h", "j", "k", and "l" move the cursor left, down, up, and right, respectively. The keys "H", "J", "K", and "L" move the cursor to the respective limits of the screen. Typing a number before the movement keys repeats the movement that number of times. Return enters the cursor position as the upper left corner of the window. The lower right corner is entered in the same manner. During this process, the placement of the new window is indicated by a rectangular box drawn on the screen, corresponding to where the new window is framed. Typing escape at any point cancels this command.

This window becomes the current window, and is given the first available ID. The default buffer size is used (see **nline** command below.)

Only fully visible windows can be created this way.

c# Close window #. The process in the window is sent the hangup signal (see **kill**). The **csh** command should handle this signal correctly and cause no problems.

m# Move window # to another location. A box in the shape of the window is drawn on the screen to indicate the new position of the window, and the same keys as those for the **w** command are used to position the box. The window can be moved partially off-screen.

M# Move window # to its previous position.

s# Change the size of window #. The user is prompted to enter the new lower right corner of the window. A box is drawn to indicate the new window size. The same keys used in **w** and **m** are used to enter the position.

S# Change window # to its previous size.

^Y Scroll the current window up by one line.

^E	Scroll the window down by one line.
^U	Scroll the window up by half the window size.
^D	Scroll the window down by half the window size.
^B	Scroll the window up by the full window size.
^F	Scroll the current window down by the full window size.
h	Move the cursor of the current window left by one column.
j	Move the cursor of the current window down by one line.
k	Move the cursor of the current window up by one line.
l	Move the cursor of the current window right by one column.
^S	Stop output in the current window.
^Q	Start output in the current window.
:	Enter a line to be executed as long commands. Normal line editing characters (erase character, erase word, erase line) are supported.

Long Window

Long commands are a sequence of statements parsed much like a programming language, with a syntax similar to that of C. Numeric and string expressions and variables are supported, as well as conditional statements.

There are two data types: string and number. A string is a sequence of letters of digits beginning with a letter. "_" and "." are considered letters. Alternately, non-alphanumeric characters can be included in strings by quoting them in " " or escaping them with \\. In addition, the \\ sequences of C are supported, both inside and outside quotes. (That is "\\n" is a new line, and "\\r" a carriage return). These are also legal strings: abcde01234, "&#$^*&# ", ab"$#"cd, ab\\$\\#cd, "**/usr/ucb/window**".

A number is an integer value in one of three forms: a decimal number, an octal number preceded by "0", or a hexadecimal number preceded by "0x" or "0X". The natural machine integer size is used (the signed integer type of the C compiler). As in C, a non-zero number represents a Boolean true.

The character "#" begins a comment which terminates at the end of the line.

A statement is either a conditional or an expression. Expression statements are terminated with a new line or ";". To continue an expression on the nest line, terminate the first line with "\\".

Conditional Statement

Window has a single control structure: the fully bracketed if statement in the form.

```
     if <expr> then
```

```
        <statement>
    elsif <expr> then
        <statement>
    else
        <statement>
    endif
```

The **else** and **elsif** parts are optional, and the latter can be repeated any number of times. **<Expr>** must be numeric.

Expressions

Expressions in **window** are similar to those on the C language, with most C operators supported on numeric operands. In addition, some are overloaded to operate on strings.

When an expression is used as a statement, its value is discarded after evaluation. Therefore, only expressions with side effects (assignments and function calls) are useful as statements.

Single valued (no arrays) variables are supported, of both numeric and string values. Some variables are predefined (see "Predefined Variables").

The operators in order of increasing precedence:

<expr1> = <expr2>
Assignment. The variable of name **expr1**, which must be string valued, is assigned the result of **expr2**. Returns the value of **expr2**.

<expr1> ? <expr2> : <expr3>
Returns the value of **expr2** if **expr1** evalutes true (non-zero numeric value) : returns the value of **expr3** otherwise. Only one of **expr2** and **expr3** is evaluated. **Expr1** must be numeric.

<expr1> || <expr2>
Logical or. Numeric values only. Short circuit evaluation is supported (that is, if **expr1** evaluates true, then **expr2** is not evaluated).

<expr1> && <expr2>
Logical and with short circuit evaluation. Numeric values only.

<expr1> | <expr2>
Bitwise or. Numeric values only.

<expr1> ^ <expr2>
Bitwise exclusive or. Numeric values only.

<expr1> & <expr2>
Bitwise and. Numeric values only.

<expr1> == <expr2>, <expr1> != <expr2>
Comparison (equal and not equal, respectively). The Boolean result (either 1 or 0) of the comparison is returned. The operands can be numeric or string valued. One string operand forces the other to be converted to a string in necessary.

<expr1> < <expr2>, <expr1> > <expr2>, <expr1> <= <expr2>, <expr1> >=

<expr2>
Less than, greater than, less than or equal to, greater than or equal to. Both numeric and string values, with automatic conversion as above.

<expr1> << <expr2>, <expr1> >> <expr2>
If both operands are numbers, **expr1** is bit shifted left (or right) by **expr2** bits. If **expr1** is a string, then its first (or last) **expr2** characters are returns (if **expr2** is also a string, then its length is used in place of its value).

<expr1> + <expr2>, <expr1> - <expr2>
Additions and subtractions of numbers. For "+", if one argument is a string, then the other is converted to a string, and the result is the concatenation of the two strings.

<expr1> * <expr2>, <expr1> / <expr2>, <expr1> % <expr2>
Multiplication, division, modulo. Numbers only.

-<expr>, °<expr>, !<expr>, $<expr>, $?<expr>
The first three are unary minus, bitwise complement and logical compliment on numbers only. The operator, "$", takes **expr** and returns the value of the variable of that name. If **expr** is numeric with value **n** and it appears within an alias macro (see below), then it refers to the nth argument of the alias invocation. "$?" tests for the existence of the variable **expr**, and returns 1 if it exists or 0 otherwise.

<expr>(<arglist>)
Function call. **expr** must be a string that is the unique prefix of the name of a built-in **window** function or the full name of a user defined alias macro. In the case of a built-in function, **argkist** can be in one of two forms:

```
<expr1>, <expr2>, . . .
argname1 = <expr1>, argname2 = <expr2>, . . .
```

The two forms can in fact be intermixed, but the result is unpredictable. Most arguments can be omitted; default values are supplied for them. The **argnames** can be unique prefixes of the argument names. The commas separating arguments are used only to disambiguate, and can usually be omitted.

Only the first argument form is valid for user defined aliases. Aliases are defined using the **alias** built-in function (see below). Arguments are accessed via a variant of the variable mechanism "$" operator above).

Most functions return value, but some are used for side effect only and so must be used as statements. When a function or an alias is used as a statement, the parenthesis surrounding the argument list may be omitted. Aliases return no value.

Built-in Function

The arguments are listed by name in their natural order. Optional arguments are in square brackets ("[ ]"). Arguments that have no names are in angle brackets ("<>").

alias([<string>], [<string-list>])
If no argument is given, all currently defined alias macros are

listed. Otherwise, **string** is defined as an alias, with expansion **string-list**. The previous definition of **string**, if any, is returned. Default for **string-list**is no change.

close(<window-list>)
Close the windows specified in **window-list**. If **window-list** is the word **all**, than all windows are closed. No value is returned.

cursormodes([modes])
Set the window cursor to **modes**. **Modes** is the bitwise or of the mode bits defined as the variables **m ul** (underline), **m rev** (reverse video), **m blk** (blinking), and **m grp** (graphics, terminal dependent). Return value is the previous modes. Default is no change. For example, cursor(Sm_rev|Sm_blk) sets the window cursors to blinking reverse video.

echo([window] [<string-list>])
Write the lists of strings, **string-list**, to **window**, separated by spaces and terminated with a new line. The strings are only displayed in the window; the processes in the window are not involved (see **write** below). No value is returned. Default is the current window.

escape([escapec])
Set the escape character to **escapec**. Returns the old escape character as a one character string. Default is no change. **escapec** can be a string of a single character, or in the form ^X, meaning control -**X**.

foreground([window], [flag])
Move **window** in or out of foreground. **Flag** can be one of **on**, **off**, **yes**, **no**, **true**, or **false**, with obvious meanings, or it can be a numeric expression, in which case a non-zero value is true. Returns the old foreground flag as a number. Default for **window** is the current window, default for **flag** is no change.

label([window], [label])
Set the label of **window** to **label**. Returns the old label as a string. Default for **window** is the current window, default for **label** is no change. To turn off a label, set it to an empty string (" ").

list()
No arguments. List the identifiers and labels of all windows. No value is returned.

nline([nline])
Set the default buffer size to **nline**. Initially, it is 48 lines. Returns the old default buffer size. Default is no change. Using a very large buffer can slow the program down considerably.

select([window])
Make **window** the current window. The previous current window is returned. Default is no change.

shell([<string-list>])
Set the default window shell program to **string-list**. Returns the first string in the old shell setting. Default is no

change. Initially, the default shell is taken form the environment variable **SHELL**.

source(filename)
Read and execute the long commands in **filename**. Returns -1 if the file cannot be read, 0 otherwise.

terse([flag])
Set terse mode to **flag**. In terse mode, the command window stays hidden even in command mode, and errors are reported by sounding the terminal's bell. **Flag** can take on the same values as in foreground above. Returns the old terse flag. Default is no change.

unalias(alias)
Undefine **alias**. Returns -1 if **alias** does not exist, 0 otherwise.

unset(variable)
Undefine **variable**. Returns -1 if **variable** does not exist, 0 otherwise.

variable()
No arguments. List all variables. No value is returned.

window(row column nrow ncol
[nline] [label] [pty] [frame] [mapnl] [shell])

Open a window with upper left corner at **row**, **column** and size **nrow**, **ncol**. If **nline** is specified, then that many lines are allocated for the text buffer. Otherwise the default buffer size is used. Default values for **row**, **column**, **nrow**, and **ncol** are, respectively, the upper, left-most, lower, or right-most extremes of the screen. **Frame, pty**, and **mapnl** are flag values interpreted in the same way as the argument to **foreground** (see above); they mean, respectively, put a frame around this window (default true), allocate pseudo-terminal for this window rather than socketpair (default true), and map new line characters in this window to carriage return and line feed (default true if socketpair is used, false otherwise). **Shell** is a list of strings that are used as the shell program to place on the window (default is the program specified by **shell**, see below). The created window's identifier is returned as a number.

write([window], [<string-list>])
Send the list of strings, **string-list**, to **window**, separated by spaces but not terminated with a new line. The strings are actually given to the window as input. No value is returned. Default is the current window.

Predefined Variables

These variables are for information only. Redefining them does not affect the internal operation of **window**.

baud The baud rate as a number between 50 and 38400.

modes The display modes (reverse video, underline, blinking, graphics) supported by the physical terminal. The value of **modes** is the bitwise or of some of the one bit values, **m blk**, **m grp**, **m rev**, and

m ul (see below). These values are useful in setting the window cursors modes (see **cursormodes** above).

m_blk	The blinking mode bit.
m_grp	The graphics mode bit.
m_rev	The reverse video mode bit.
m_ul	The underline mode bit.
ncol	The number of columns on the physical screen.
nrow	The number of rows on the physical screen.
term	The terminal type. The standard name, found in the second name field of the terminal's **terminfo** entry, is used.

Files

$HOME/.windowrc	Startup command file.

1.1.541 write

1.1.542 write

Purpose
Sends messages to other users on the system.

Syntax

```
         +------+                     +------------+
write ---¦      +--- user --- line ---¦            +---¦
         +- -L -+                     +- sitename -+
```

Description
The **write** command is used to **converse** with another logged-in **user**. That is, each user alternately sends and receives short messages from the other's terminal. Long messages can be sent by first putting the complete message in a file and then redirecting that file as input to the **write** command.

For another **user** to receive your message, that user must be logged in and must not have refused message permission. When a person you are trying to reach is not logged in, you get the message **user not logged in**. When the person you are trying to reach has refused message permission, you get the message **write: permission denied**.

When you run the **write** command, it immediately sends the following message, along with an attention-getting sound (the ASCII BEL character) to the person whose login name you entered.

  Message from **yourid** on **yoursitename** (tty**nn**) [date]...

After successful connection, **write** then sends two ASCII BEL characters to your terminal to alert you that whatever you enter now is being sent, one line at a time, to the other **user**. Sending continues until you press **Ctrl-D**, at which point **write** sends an end-of-text character to the other terminal and exits. At this point, the other **user** can respond if desired by sending a **write** message back. To carry on an exchange of several messages, you can use the following convention. After entering your first message, do not type **Ctrl-D** to exit **write**. Instead terminate each message with a signal such as o (over) to alert the other person to reply. Each user can alternate sending messages, not typing **Ctrl-D** until the conversation is finished.

When you **write** to a **user** logged in at more than one terminal on your site, **write** uses the first login instance found in file **/etc/utmp** as the message delivery point, and you get the message:

  userid is logged in more than once on **sitename**.
  You are connected to line1.
  Other locations are line2.

You can contact this **user** at another terminal by specifying the **line**. **line** indicates to which terminal (**tty00**, for example) the message should be sent.

When your machine is operating as a site in a TCF cluster, you can converse with users logged into sites anywhere on the cluster. If the **user** you specify is not logged into your local site, write connects to the user at the lowest-numbered site on the cluster where that user is logged in. Specify **sitename** if you want to **write** to the user at a specific site.

To specify a **line** without a **user**, or a **sitename** without either **line** or **user**, use a dash (-) in place of the omitted **line** or **user**. You cannot omit both **line** and **user**.

Permission to **write** to another **user** is granted or denied by the other **user** with the **mesg** command. Some commands deny message permission while they are running to prevent interference with their output. A user with superuser authority can **write** to any terminal regardless of the terminal's message permission.

If you are sending a message to another user on a system on which users may have selected different character code sets for displaying text in different languages, the message should be written in either the same code set used by the recipient or in ASCII characters.

Flag

-L	Write to user only if that user is logged into the local site. If the **-L** flag is specified, the sitename argument cannot also be specified.

Examples

1. To write a message to a user who is logged in:

   ```
   write billie
   I need to see you!  Meet me in the computer room at 12:30.
   Ctrl-D
   ```

   If your user ID is **kirk** and you are using terminals **tty3** on site alpha, **billie**'s terminal displays:

   ```
   Message from kirk on alpha (tty3) [current date] ...
   I need to see you!  Meet me in the computer room at 12:30.
   EOF
   ```

2. To hold a conversation:

   ```
   write billie
   Meet me in the computer room at 12:30.
   (o)
   ```

   This starts the conversation. The **(o)** at the end stands for "over". It tells **billie** that you are waiting for a response. **Do not** press **Ctrl-D** if you wish to continue.

   Now **billie** replies by typing:

   ```
   write kirk
   I'm running tests at 12:30.  Can we meet at 3?
   (o)
   ```

   And you might respond:

   ```
   OK--the computer room at 3.
   (oo)
   ```

   The **(oo)** stands for "over and out," telling **Billie** that you have nothing more to say. If **Billie** is also finished (**oo**), you both press

Ctrl-D to end the conversation.

3. To write someone a prepared message:

```
write jay <message.text
```

   This writes the contents of the file **message.text** to **Jay's** terminal.

4. To write to the person using a certain terminal:

```
write - console
The printer in building 998 has jammed.
Please send help.
Ctrl-D
```

   This writes the message to the person logged in at the terminal **/dev/console** on your site.

Files

/etc/utmp	Contains user and accounting information for the **who**, **write**, and **login** commands.

Related Information

See the following commands: "mesg" in topic 1.1.262, "nroff, troff" in topic 1.1.301, "pr" in topic 1.1.322, "sh, Rsh" in topic 1.1.420, "wall" in topic 1.1.529 and "who" in topic 1.1.537.

1.1.543 wump

Purpose
Plays the game Hunt the Wumpus.

Syntax

```
/usr/games/wump ---¦
```

Description

A wumpus is a creature living in a cave with many rooms interconnected by tunnels. You move among the rooms trying to shoot the wumpus with an arrow and trying to avoid being eaten by the wumpus or falling into Bottomless Pits. There are also Super Bats that may pick you up and drop you in some randomly selected room. For moving among the rooms and shooting arrows, the **wump** command asks appropriate questions and follows your instructions.

After either you kill the wumpus, the wumpus eats you, or you fall into a Bottomless Pit, the **wump** command asks if you want a new game. To quit the game at any time, press the interrupt key. To quit the game on a new line, press the end-of-file key (**Ctrl-D**).

1.1.544 xargs

Purpose
Constructs argument lists and runs commands.

Syntax

```
         +--------+   +------------------+   +-- -s470 --+   +------+
xargs ---¦ one of +---+-- -nnum ---------+---¦           +---¦      +---
         ¦ +----+ ¦   ¦        +---1---+ ¦   +- -ssize --+   +- -x -+
         +-¦ -p +-+   +-- -l --¦       +-+
           ¦ -t ¦              +- num -+
           +----+

    +----- -e -------+   +----------------+   +--- echo ----+
 ---¦ +---- -e ----+ +---¦ +---- -i -----+ +---¦             +---¦
    +-¦            +-+   +-¦             +-+   +- cmdstring -+
      +- -eeofstr -+       +- -ireplstr -+
```

Description
The **xargs** command runs a command line. It constructs the command line by combining **cmdstring**, a string containing a command and its flags or parameters, with additional arguments read from standard input. It runs **cmdstring** as many times as necessary to process all input arguments. The default **cmdstring** is **echo**.

Arguments read from standard input are character strings delimited by one or more blanks, tabs, or new-line characters. You can imbed a blank or a tab in arguments by preceding it with a \ (backslash) or by quoting it. The **xargs** command reads characters enclosed in single or double quotes as literals and removes the delimiting quotes. It always discards empty lines.

Flags

-e[eofstr]	Sets the logical end-of-file string to **eofstr**. The **xargs** command reads standard input until it encounters either an end-of-file character or the logical **EOF** string. If you do not specify the **-e** flag, the default **eofstr** is _ (the underline character). If you specify **-e** with no **eofstr**, **xargs** interprets the underline character as a literal character rather than as an end-of-file marker.
-i[replstr]	Takes an entire line as a single argument and inserts it in each instance of **replstr** found in **cmdstring**. A maximum of five arguments in **cmdstring** may each contain one or more instances of **replstr**. The **xargs** command discards blanks and tabs at the beginning of each line. The argument constructed may not be larger than 255 characters. The default **replstr** is **{}**. This flag also turns on the **-x** flag.
-l[num]	Runs **cmdstring** with the specified **num** of non-empty argument lines read from standard input. The last invocation of **cmdstring** can have fewer argument lines if fewer than **num** remain. A line ends with the first new-line character unless the last character of the line is a blank or a tab. A trailing blank or tab indicates a continuation through the next non-empty line. The default **num** is 1. This flag turns on the **-x** flag.

-nnum Executes **cmdstring** using as many standard input arguments as possible, up to a maximum of **num**. **xargs** uses fewer arguments if their total size is greater than the number of characters specified by the **-ssize** flag described below. It also uses fewer arguments for the last invocation if fewer than **num** arguments remain. When **-x** is present, each **num** argument must fit the **size** limitation specified by **-x**.

-p Asks whether to run **cmdstring**. It displays the constructed command line, followed by a **?...** prompt. Press **y** to run the **cmdstring**. Any other response causes **xargs** to skip that particular invocation of **cmdstring**. You are asked about each invocation.

-ssize Sets the maximum total size of each argument list. **size** must be a positive integer less than or equal to 470. The default **size** is 470 characters. The character count for **size** includes one extra character for each argument and the number of characters in the command name.

-t Echoes the **cmdstring** and each constructed argument list to file descriptor 2 (usually standard error).

-x Stops running **xargs** if any argument list is greater than the number of characters specified by the **-ssize**. This flag is turned on if you specify either the **-i** or **-l** flags. If you do not specify **-i**, **-l**, or **-n**, the total length of all arguments must be within the **size** limit.

Notes:

1. If you have selected a language (through the **LANG** environment variable) that supports multibyte characters, the character limits cited in the **-i** and **-s** descriptions can be reduced by as much as 50%, depending on the character code set being used.

2. The **xargs** command ends if it cannot run **cmdstring** or if it receives a return code of -1. When **cmdstring** calls a shell procedure, the shell procedure should explicitly **exit** with an appropriate value to avoid accidentally returning -1. (See "sh, Rsh" in topic 1.1.420.)

Examples

1. To use a command on files whose names are listed in a file:

```
xargs  lint  -a  <cfiles1  *
```

   If **cfiles** contains the text:

```
main.c  readit.c
gettoken.c
putobj.c
```

   then **xargs** constructs and runs the command:

```
lint  -a  main.c  readit.c  gettoken.c  putobj.c
```

   Each shell command line can be up to 470 characters long. If **cfiles** contains more file names than fit on a single line, **xargs** runs the **lint** command with the file names that fit. It then constructs and runs another **lint** command using the remaining file names. Depending on the names listed in **cfiles**, the commands might look like:

```
lint  -a  main.c  readit.c  gettoken.c...
lint  -a  getisx.c  getprp.c  getpid.c...
lint  -a  fltadd.c  fltmult.c  fltdiv.c...
```

This is not quite the same as running **lint** once with all the file names. The **lint** command checks cross-references between files. However, in this example it cannot check between **main.c** and **fltadd.c**, or between any two files listed on separate command lines.

For this reason you may want to run the command only if all the file names fit on one line. Tell **xargs** this by using the **-x** flag:

```
xargs  -x  lint  -a  <cfiles
```

If all the file names in **cfiles** do not fit on one command line, **xargs** displays an error message.

2. To construct commands that contain a certain number of file names:

```
xargs  -t  -n2  diff  <<end
starting  chap1  concepts  chap2  writing
chap3
end
```

This constructs and runs **diff** commands that contain two file names each (**-n2**):

```
diff  starting  chap1
diff  concepts  chap2
diff  writing   chap3
```

The **-t** flag tells **xargs** to display each command before running it so that you can see what is happening. The **<<end** and **end** define a "Here Document," which uses the text entered before the **end** line as standard input for the **xargs** command. For more details, see "Inline Input Documents" in topic 1.1.420.19.

3. To insert file names into the middle of commands:

```
ls  |  xargs  -t  -i  mv  '{}''{}.old'
```

This renames all files in the current directory by adding **.old** to the end of each name. The **-i** tells **xargs** to insert each line of the **ls** directory listing where a **{}** appears. If the current directory contains the files **chap1**, **chap2**, and **chap3**, then this constructs the commands:

```
mv  chap1  chap1.old
mv  chap2  chap2.old
mv  chap3  chap3.old
```

4. To run a command on files that you select individually:

```
ls  |  xargs  -p  -n1  ar  r  lib.a
```

This allows you to select files to add to the library **lib.a**. The **-p** flag tells **xargs** to display each **ar** command it constructs and ask if you want to run it. Type **y** and press **Enter** to run the command. Press **Enter** alone if you do not want to run it.

Related Information

See the following command: "sh, Rsh" in topic 1.1.420.

1.1.545 xsend, xget, enroll

Purpose
Implements a secure communication.

Syntax

xsend ---¦

xget ---¦

enroll ---¦

Description

These commands implement a secure communication channel; it is like **mail**, but no one can read the messages except the intended recipient. The method embodies a public-key cryptosystem using knapsacks.

To receive messages, use the **enroll** command; it asks you for a password that you must subsequently quote in order to receive secret mail.

To receive secret mail, use **xget**. It asks for your password, then gives you the messages.

To send secret mail, use **xsend** in the same manner as the ordinary mail command. (However, it accepts only one target.) A message announcing the receipt of secret mail is also sent by ordinary mail.

Note: Secret mail should be integrated with ordinary mail. The announcement of secret mail makes traffic analysis possible.

Files

N/.smail/*key	Keys.
N/.smail/* [0-9]	Messages.

Related Information

See the following command: "mail, Mail" in topic 1.1.253.

1.1.546 xstr

Purpose
Maintains a file "**strings**".

Syntax

```
        +--------+   +--------+
xstr ---¦ +----+ +---¦        +---¦
        +-¦ -c +-+   +- file -+
          ¦    ¦¦
         ¦+----+¦
         +------+
```

Note: This command does not have MBCS support.

Description

The **xstr** command maintains a file **strings** into which strings in component parts of a large program are hashed. These strings are replaced with references to this common area. This serves to implement shared constant strings, most useful if they are also read-only.

The command **xstr -c name** extracts the strings from the C source in name, replacing string references by expressions of the form (**&xstr[number]**) for some number. An appropriate declaration of **xstr** is prepended to the file. The resulting C text is placed in the file **x.c**, to then be compiled. The strings from this file are placed in the **strings** data base if they are not there already. Repeated strings and strings which are suffices of existing strings do not cause changes to the data base.

After all components of a large program have been compiled, a file **xs.c** declaring the common **xstr** space can be created by a command of the form **xstr**. This **xs.c** file should then be compiled and loaded with the rest of the program. If possible, the array can be made read-only (shared) saving space and swap overhead.

The **xstr** command can also be used on a single file. A command **xstr name** creates files **x.c** and **xs.c** as before, without using or affecting any **strings** file in the same directory.

It may be useful to run **xstr** after the C preprocessor if any macro definitions yield strings or if there is conditional code which contains strings which may not, in fact, be needed. **xstr** reads from its standard input when the argument '-' is given. An appropriate command sequence for running **xstr** after the C preprocessor is:

```
  cc -E name.c ¦ xstr -c -
  cc -c x.c
  mv x.o name.o
```

The **xstr** command does not touch the file **strings** unless new items are added, thus **make** can avoid remaking **xs.o** unless truly necessary.

Note: If a string is a suffix of another string in the data base but the shorter string is seen first by the **xstr** command, both strings are placed in the data base, when just placing the longer one there is sufficient.

Files

strings	Data base of strings.
x.c	Massaged C source.
xs.c	C source for definition of array xstr.
/tmp/xs*	Temp file when xstr name doesn't touch **strings**.

Related Information

See the following command: "mkstr" in topic 1.1.273.

1.1.547 yacc

Purpose
Generates an LR parsing program from input consisting of a context-free grammar specification.

Syntax

```
       +-----------+
yacc ---¦ +-------+ +-- grammar --¦
       +-¦ -v -d +-+
         ¦ -l -s ¦
        ¦¦ -t    ¦¦
        ¦+-------+¦
        +---------+
```

Note: This command does not have MBCS support.

Description

The **yacc** command converts a context-free grammar into a set of tables for a simple automaton that executes an LR(1) parsing algorithm. The grammar can be ambiguous; specified precedence rules are used to break ambiguities.

You must compile the output file, **y.tab.c**, with a C Language compiler to produce a function **yyparse**. This function must be loaded with the lexical analyzer function **yylex**, as well as **main** and **yyerror**, an error-handling routine (you must provide these routines). The **lex** command is useful for creating lexical analyzers usable by **yacc**.

For more detailed discussion of **yacc** and its operations, see *AIX Operating System Programming Tools and Interfaces*.

Flags

-d Produces the file **y.tab.h**. This contains the **#define** statements that associate the **yacc**-assigned token codes with your token names. This allows source files other than **y.tab.c** to access the token codes by including this header file.

-l Does not include any **#line** constructs in **y.tab.c**. Use this only after the grammar and associated actions are fully debugged.

-s Breaks the **yyparse** function into several smaller functions. Since its size is somewhat proportional to that of the grammar, it is possible for **yyparse** to become too large to compile, optimize, or execute efficiently.

-t Compiles run-time debugging code. By default, this code is not included when **y.tab.c** is compiled. However, the run-time debugging code is under the control of **YYDEBUG**, a global variable for the **cc** command preprocessor. If **YYDEBUG** has a nonzero value, the C compiler (**cc**) includes the debugging code, whether the **-t** flag was used. Without compiling this code, **yyparse** has a faster operating speed.

-v Prepares the file **y.output**. It contains a readable description of the parsing tables and a report on conflicts generated by grammar ambiguities.

Files

y.output	
y.tab.c	
y.tab.h	Definitions for token names.
yacc.tmp,	
yacc.debug	Temporary file.
yacc.acts	Temporary file.
/usr/lib/yaccpar	Parser prototype for C programs.

Related Information

See the following command: "lex" in topic 1.1.229.

See the description of **yacc** in *AIX Operating System Programming Tools and Interfaces*.

1.1.548 yes

Purpose
Be repetitively affirmative.

Syntax

```
yes --- phrase ---¦
```

Description

The **yes** command repeatedly outputs "y", or if **phrase** is given, that is output repeatedly. Termination is by keyboard INTR character (see "stty, STTY" in topic 1.1.447).

1.1.549 ypcat

Purpose
Displays the values in a network information service (NIS) data base.

Syntax

```
        +--------+   +----------------+
ypcat ---¦ +----+ +---¦                +--- mname ---¦
        +-¦ -k +-+   +- -d domainname -+
          ¦ -t ¦
         ¦+----+¦
         +------+
        +------+
ypcat ---¦      +---¦
        +- -x -+
```

Note: This command does not have MBCS support.

Description

The **ypcat** command displays values in an NIS map specified by the **mname** parameter, which can be either a map name or a map nickname. Since the **ypcat** command uses the network information service, you do not specify a NIS server.

For example, to look at the network-wide password data base, **passwd.byname** (with the nickname **passwd**), enter:

```
ypcat passwd
```

Flags

-k	Displays the keys for maps in which the values are null or the key is not part of the value. (None of the maps derived from files that have an ASCII version in **/etc** fall into this class.)
-t	Inhibits translation of **mname** to **mapname**. For example, **ypcat -t passwd** fails because there is no map named **passwd**, whereas **ypcat passwd** is translated to **ypcat passwd.byname**.
-d domainname	Specifies a domain other than the default domain. The default domain is returned by **domainname**.
-x	Displays the map nickname table. The **-x** flag lists the nicknames (**mnames**) and indicates the **mapname** associated with each nickname.

Related Information

See the following commands: "ypserv, ypbind" in topic 1.1.555, "ypmatch" in topic 1.1.551, and "domainname" in topic 1.1.133.

1.1.550 ypinit

Purpose

Builds and installs NIS data base.

Syntax

```
                  +------- -m -------+
/etc/yp/ypinit ---¦                  +---¦
                  +- -s master_name -+
```

Note: This command does not have MBCS support.

Description

ypinit sets up an NIS data base on an NIS server for the current NIS domain. The domain must have been set earlier using the **domainname** command. You must be the superuser to run it. It asks a few, self-explanatory questions, and reports success or failure to the terminal.

It sets up a master server using the simple model in which that server is master to all maps in the data base. This is the way to bootstrap the NIS system; later if you want you can change the association of maps to masters. All data bases are built from scratch, either from information available to the program at runtime, or from the ASCII data base files in **/etc**. These files are listed below under **Files**. All such files should be in their "traditional" form, rather than the abbreviated form used on client machines.

An NIS data base on a slave server is set up by copying an existing data base from a running server. The **master name** argument should be the hostname of NIS server (either the master server for all the maps, or a server on which the data base is up-to-date and stable).

ypinit creates its map files in the directory **/local/yp/<domain>**. Furthermore, the file **/etc/yp/<domain>/MASTER** is created if this NIS host is the master server. This file is created so that in a TCF cluster, only the one host which will be serving as the master server will start up **ypserv** and **rpc.yppasswdd**.

Flags

-m	Indicates that the local host is to be the NIS master.
-s	Set up a slave data base.

Files

/etc/passwd
/etc/group
/etc/hosts
/etc/networks
/etc/services
/etc/protocols
/etc/yp/<domain>/MASTER

Related Information

See the following commands: "makedbm" in topic 1.1.255, "yppush" in topic 1.1.554, "ypxfr" in topic 1.1.558, "ypserv, ypbind" in topic 1.1.555, and "domainname" in topic 1.1.133.

1.1.551 ypmatch

Purpose
Displays the value of one or more keys from an NIS map.

Syntax

```
           +-------------+   +--------+
ypmatch ---¦             +---¦ +----+ +--- key --- mname ---¦
           +- -d domain -+   +-¦ -k +-+         ¦
                               ¦ -t ¦   +-------+
                              ¦+----+¦
                              +------+
           +------+
ypmatch ---¦      +---¦
           +- -x -+
```

Note: This command does not have MBCS support.

Description

The **ypmatch** command displays the values associated with one or more keys from the NIS map specified by the **mname** parameter, which can be either a map name or a map nickname. Multiple keys can be specified; the same map is searched for all. The keys must be exact values. No pattern matching is available. If a key is not matched, a message is displayed.

Flags

-d domain	Specifies a domain other than the default domain.
-k	Displays the keys, followed by a colon (:) and the value of the key.
-t	Inhibits translation of nicknames to map names. For example, **ypmatch -t zippy passwd** fails because there is no map named **passwd**, whereas **ypmatch zippy passwd** is translated to **ypmatchp zippy passwd.byname**.
-x	Displays the map nickname table, which lists the nicknames (**mnames**) and indicates the map name associated with each nickname.

Related Information

See the following command: "ypcat" in topic 1.1.549.

1.1.552 yppasswd

Purpose

Changes login password in NIS.

Syntax

```
            +--------+
yppasswd ---¦        +---¦
            +- name -+
```

Note: This command does not have MBCS support.

Description

The **yppasswd** command changes (or installs) a network password associated with the user name (your own name by default) in the NIS. The NIS password can be different from the one on your own machine.

The **yppasswd** command prompts for the old NIS password and then for the new one. You must enter the old password correctly for the change to take effect. The new password must be typed twice to avoid mistakes. If you enter your old password incorrectly, you are not notified until after you have entered your new password.

New passwords must contain at least four ASCII characters if they use a sufficiently rich alphabet, and at least six ASCII characters if monocase. These rules are relaxed if you are insistent enough. Only the owner of the name or the superuser can change a password; in either case, you must prove you know the old password.

The **rpc.yppasswdd** daemon must be running on your NIS server for the new password to take effect.

Related Information

See the following command: "passwd, chfn, chsh" in topic 1.1.312.

See "Configuring the Network Information Services on Your System" - "Starting the **yppasswd** Daemon," in *Managing the AIX Operating System*.

1.1.553 yppoll

Purpose

Discovers what version of an NIS map is at an NIS server host.

Syntax

```
                  +-----------+   +-------------+   +-----------+
/etc/yp/yppoll ---¦           +---¦             +---¦           +---¦
                  +- -h host -+   +- -d domain -+   +- mapname -+
```

Note: This command does not have MBCS support.

Description

The **yppoll** command asks a **ypserv** process what the order number is and which host is the master NIS server for the named map. If the server is a v.1 NIS protocol server, **yppoll** uses the older protocol to communicated with it. In this case, it also uses the older diagnostic messages in case of failure. If host is omitted, the local host is used.

Flags

-h host	Ask the **ypserv** process at **host** about the map parameters. If **host** isn't specified, the NIS server for the local host is used. That is, the default is the one returned by **ypwhich**. Host may be either a name or an internet address of form xx.yy.zz.
-d domain	Use **domain** instead of the default domain.

Related Information

The following command: "ypserv, ypbind" in topic 1.1.555.

1.1.554 yppush

Purpose
Forces propagation of a changed NIS map.

Syntax

```
                  +-------------+   +------+
/etc/yp/yppush ---¦             +---¦      +--- mapname ---¦
                  +- -d domain -+   +- -v -+
```

Note: This command does not have MBCS support.

Description

The **yppush** command copies a new version of an NIS map from the master NIS server to the slave NIS servers. It is normally run only on the master NIS server by the **Makefile** in **/usr/etc/yp/** after the master data bases are changed. It first constructs a list of NIS server hosts by reading the NIS map **ypservers** within the domain. Keys within the map **ypservers** are the ASCII names of the machines on which the NIS servers run.

A "transfer map" request is sent to the NIS server at each host, along with the information needed by the transfer agent (the program which actually moves the map) to call back the **yppush**. When the attempt has completed (successfully or not), and the transfer agent has sent **yppush** a status message, the results may be printed to **stdout**. Messages are also printed when a transfer is not possible; for instance when the request message is undeliverable, or when the timeout period on responses has expired.

Flags

-d	Specifies a domain
-v	Verbose; causes messages to be printed when each server is called, and for each response. If this flag is omitted, only error messages are printed.

Files

/etc/yp/domainname/ypservers.{dir,pag}

Related Information

See the following commands: "ypserv, ypbind" in topic 1.1.555, "ypxfr" in topic 1.1.558.

1.1.555 ypserv, ypbind

Purpose
NIS server and binder processes.

Syntax

/etc/ypserv ---¦

/etc/ypbind ---¦

Note: This command does not have MBCS support.

Description

NIS provides a simple network lookup service consisting of data bases and processes. The data bases are **dbm** files in a directory tree rooted at **/etc/yp**. These files are described in **ypfiles**. The processes are **/etc/ypserv**, the NIS data base lookup server, and **/etc/ypbind**, the NIS binder. The programmatic interface to NIS is described in **ypclnt**. Administrative tools are described in **yppush**, **ypxfr**, **yppoll**, **ypwhich**, and **ypset**. Tools to see the contents of NIS maps are described in **ypcat**, and **ypmatch**. Data base generation and maintenance tools are described in **ypinit** and **makedbm**.

Both **ypserv** and **ypbind** are daemon processes typically activated at system startup time from **/local/local.init.dir/Singl2multi**. **ypserv** runs only on NIS server machines with a complete NIS data base. **ypbind** runs on all machines using NIS, both NIS servers and clients.

The **ypserv** daemon's primary function is to look up information in its local data base of NIS maps. The operations performed by **ypserv** are defined for the implementer by the NIS protocol specification, and for the programmer by the header file **<rpcsvc/yp_prot.h>**. Communication to and from **ypserv** is by means of RPC calls. Lookup functions are described in **ypclnt**, and are supplied as C-callable functions in **/lib/libc** There are four lookup functions, all of which are performed on a specified map within some NIS domain: Match, Get_first, Get_next, and Get_all. The Match operation takes a key, and returns the associated value. The Get_first operation returns the first key-value pair from the map, and Get_next can be used to enumerate the remainder. Get_all ships the entire map to the requester as the response to a single RPC request.

Two other functions supply information about the map, rather than map entries; Get_order_number, and Get_master_name. In fact, both order number and master name exist in the map as key-value pairs, but the server does not return either through the normal lookup functions. (If you examine the map with **makedbm**, however, they are visible.) Other functions are used within the NIS subsystem itself, and are not of general interest to NIS clients. They include Do_you_serve_this_domain?, Transfer_map, and Reinitialize_internal_state.

The function of **ypbind** is to remember information that lets client processes on a single node communicate with some **ypserv** process. **ypbind** must run on every machine which has NIS client processes; **ypserv** may or may not be running on the same node, but must be running somewhere on the network.

The information **ypbind** remembers is called a **binding** -- the association of a domain name with the internet address of the NIS server, and the port on that host at which the **ypserv** process is listening for service requests. The process of binding is driven by client requests. As a request for an unbound domain comes in, the **ypbind** process broadcasts on the net trying to find a **ypserv** process that serves maps within that domain. Since the binding is established by broadcasting, there must be at least one **ypserv** process on every net. Once a domain is bound by a particular **ypbind**, that same binding is given to every client process on the node. The **ypbind** process on the local node or a remote node may be queried for the binding of a particular domain by using the **ypwhich** command.

Bindings are verified before they are given out to a client process. If **ypbind** is unable to speak to the **ypserv** process it's bound to, it marks the domain as unbound, tells the client process that the domain is unbound, and tries to bind the domain once again. Requests received for an unbound domain fail immediately. In general, a bound domain is marked as unbound when the node running **ypserv** crashes or gets overloaded. In such a case, **ypbind** binds to any NIS server (typically one that is less-heavily loaded) available on the net.

The **ypbind** command also accepts requests to set its binding for a particular domain. The request is usually generated by the NIS subsystem itself, **ypset** is a command to access the Set_domain facility.

Files

If the file **/usr/etc/yp/ypserv.log** exists when **ypserv** starts up, log information is written to this when error conditions arise.

Related Information

See the following commands: "ypmatch" in topic 1.1.551, "yppush" in topic 1.1.554, "ypwhich" in topic 1.1.557, and "yppoll" in topic 1.1.553.

1.1.556 ypset

Purpose

Points **ypbind** at a particular server.

Syntax

```
                    +-----------------+
/etc/yp/ypset ---¦ +-------------+ +--- server ---¦
                    +-¦ -V1          +-+
                      ¦ -h hos       ¦¦
                     ¦¦ -d domainame ¦¦
                     ¦+--------------+¦
                     +----------------+
```

Note: This command does not have MBCS support.

Description

The **ypset** command tells the **ypbind** command to get network information service for the specified domain from the **ypserv** process running on **server**. If server is down, or isn't running **ypserv**, this is not discovered until a NIS client process tries to get a binding for the domain. At this point, the binding set by **ypset** is tested by **ypbind**. If the binding is invalid, **ypbind** attempts to rebind for the same domain.

The **ypset** command is useful for binding a client node which is not on a broadcast net, or is on a broadcast net which is not running an NIS server host. It also is useful for debugging NIS client applications, for instance, where a NIS map only exists at a single NIS server host.

In cases where several hosts on the local net are supplying network information service, it is possible for **ypbind** to rebind to another host even while you attempt to find out if the **ypset** operation succeeded. That is, you can enter **ypset host1**, and then **ypwhich**, which replies **host2**, which can be confusing. This is a function of the NIS subsystem's attempt to load-balance among the available NIS servers, and occurs when **host1** does not respond to **ypbind** because it is not running **ypserv** (or is overloaded), and **host2**, running **ypserv**, gets the binding.

server indicates the NIS server to bind to, and can be specified as a name or an IP address. If specified as a name, **ypset** attempts to use network information service to resolve the name to an IP address. This works only if the node has a current valid binding for the domain in question. In most cases, **server** should be specified as an IP address.

Flags

-V1 Bind **server** for the v.1 NIS protocol.

If no version is supplied, **ypset**, first attempts to set the domain for the (current) v.2 protocol. If this attempt fails, **ypset**, then attempts to set the domain for the (old) v.1 protocol.

-h host Set **ypbind**'s binding on **host**, instead of locally. **host** can be specified as a name or as an IP address.

-d domain   Use domain, instead of the default domain.

Related Information

See the following commands: "ypwhich" in topic 1.1.557 and "ypserv, ypbind" in topic 1.1.555.

1.1.557 ypwhich

Purpose
Tells which host is the NIS server or map master.

Syntax

```
          +-------------------+ +-----------+ +------------+
ypwhich --¦      +----------+ +-¦ +- -V1 -+ +-¦            +-¦
          +- -d -¦          +-+ +-¦       +-+ +- hostname -+
                 +- domain -+     +- -V2 -+
          +-------------+ +-------------+      +---------+
ypwhich --¦             +-¦             +- -m -¦         +-¦
          +- -t mapname -+ +- -d domain -+      +- mname -+

ypwhich -- -x --¦
```

Note: This command does not have MBCS support.

Description

The **ypwhich** command tells which NIS server supplies network information services to an NIS client, or which is the master for a map. If invoked without arguments, it gives the NIS server for the local machine. If **hostname** is specified, that machine is queried to find out which NIS master it is using.

Flags

-d	Use domain instead of the default domain.
-V1	Which server is serving v.1 NIS protocol-speaking client processes?
-V2	Which server is serving v.2 NIS protocol client processes? If neither version is specified, **ypwhich** attempts to locate the server that supplies the (current) v.2 services. If there is no v.2 server currently bound, **ypwhich** then attempts to locate the server supplying the v.1 services. Since NIS servers and NIS clients are both backward compatible, the user need seldom be concerned about which version is currently in use.
-t mapname	Inhibit nickname translation; useful if there is a mapname identical to a nickname. This is not true of any Sun-supplied map.
-m	Find the master NIS server for a map. No **hostname** can be specified with **-m mname** can be a mapname, or a nickname for a map. When **mname** is omitted, produce a list available maps.
-x	Display the map nickname table. This lists the nicknames (**mnames**) the command knows of, and indicates the **mapname** associated with each nickname.

Related Information

See the following commands: "rpcinfo" in topic 1.1.384, "ypset" in topic 1.1.556, and "ypserv, ypbind" in topic 1.1.555.

1.1.558 ypxfr

Purpose
Transfers an NIS map from some NIS server.

Syntax

```
                 +-----------+   +-------------+   +--------+
/etc/yp/ypxfr ---¦           +---¦             +---¦ +----+ +---
                 +- -h host -+   +- -d domain -+   +-¦ -f +-+
                                                     ¦ -c ¦¦
                                                    ¦+----+¦
                                                    +------+

    +-------------------------+
 ---¦                         +--- map ---¦
    +- -C tid prof ipadd part -+
```

Note: This command does not have MBCS support.

Description

The **ypxfr** command moves an NIS map to the local host by making use of normal network information service. It creates a temporary map in the directory **/etc/yp/domain** (which must already exist), fills it by enumerating the map's entries, fetches the map parameters (master and order number), and loads them. It then deletes any old versions of the map and moves the temporary map to the real mapname.

If the **ypxfr** command is run interactively, it writes its output to the terminal. However, if it is invoked without a controlling terminal, and if the log file **/etc/yp/ypxfr.log** exists, it appends all its output to that file. Since **ypxfr** is most often run from **/usr/lib/crontab**, or by **ypserv**, you can use the log file to retain a record of what was attempted, and what the results were.

For consistency between servers, **ypxfr** should be run periodically for every map in the NIS data base. Different maps change at different rates; the **services.byname** map may not change for months at a time, for instance, and may therefore be checked only once a day in the wee hours. You may know that **mail.aliases** or **hosts.byname** changes several times per day. In such a case, you may want to check hourly for updates. A **crontab** entry can be used to perform periodic updates automatically. Rather than having a separate **crontab** entry for each map, you can group commands to update several maps in a shell script. Examples (mnemonically named) are in **/etc/yp**: **ypxfr_1perday**, **ypxfr_2perday**, and **ypxfr_1perhour**. They can serve as reasonable first cuts.

Flags

-f	Force the transfer to occur even if the version at the master is not more recent than the local version.
-c	Don't send a "Clear current map" request to the local **ypserv** process. Use this flag if **ypserv** is not running locally at the time you are running **ypxfr**. Otherwise, **ypxfr** complains that it cannot talk to the local **ypserv** and the transfer fails.

-h host Get the map from **host**, regardless of what the map says the master is. If **host** is not specified, **ypxfr** asks the network information service for the name of the master and tries to get the map from there. **host** can be a name or an internet address in the form **aa.bb.cc.dd.**

-d domain Specify a domain other than the default domain.

-C tid prog ipadd port This option is only for use by **ypserv**. When **ypserv** invokes **ypxfr**, it specifies that **ypxfr** should call back a **yppush** process at the host with IP address **ipaddr**, registered as program number **prog**, listening on port **port**, and waiting for a response to transaction **tid**.

Files

/etc/yp/ypxf.log Log file.
/etc/yp/ypxfr_1perday Script to run one transfer per day, for use with **cron**.
/etc/yp/ypxfr_2perday Script to run two transfers per day.
/etc/yp/ypxfr_1perhour Script for hourly transfers of volatile maps.

Related Information

See the following commands: "cron" in topic 1.1.97, "ypserv, ypbind" in topic 1.1.555, and "yppush" in topic 1.1.554.

1.1.559 300, 300s

Purpose
Handles special line-motion functions for DASI 300/300s work stations.

Syntax

```
 one of
+------+   +-- -d3,90,30 --+
¦ 300  +---¦ +-----------+ ¦
¦ 300s ¦   +-¦ +12       +-+
+------+     ¦ -nu       ¦¦
            ¦¦ -dt,len,c ¦¦
            ¦+-----------+¦
           +-------------+
```

Note: This command does not have MBCS support.

Description

Note: If your work station has a **PLOT** switch, make sure this switch is turned **on** before using this command.

The **300** command reads standard input, processes its input for printing on the DASI 300, GSI 300, or DTC 300 work stations, and writes to standard output. The **300s** command performs the same functions for the DASI 300s, GSI 300s, and DTC 300s. They convert the input files' motion control characters for half-line forward, half-line reverse, and full-line reverse into motion commands recognized by these work stations.

You can use the **300** and **300s** commands to draw Greek characters and other special symbols that require more than one vertical line, and it allows you to use 12-pitch text. For a discussion of special symbols and Greek characters supported by **300**, see "greek" in topic 1.1.192.

The **nroff** command can be used with the **300** command to format text. **300** must be used if you use special delays or formatting options. You can either pipe from **nroff** to **300** or use the **-T300** flag with **nroff** to specify the ***printing device***. The movement control of the **300** command usually produces better aligned output than **nroff -T300**.

When using **nroff**, the **-s** flag or **.rd** requests are required for inserting paper manually or changing fonts in the middle of a document. In these cases, you must press the line feed key to continue printing.

Using the **300** command with the **neqn** command gives you the best display of your equations. You can use the following sequence to display equations:

```
  neqn file... | nroff | 300
```

Note: Some special characters cannot be correctly printed in column 1 because the print head cannot be moved to the left from that position.

If your output contains Greek characters or reverse line feeds, use a friction-feed platen instead of a forms tractor. A forms tractor slips when reversing direction.

Flags

-dt,len,c Controls output delay factors. The default setting is **-d3,90,30**. DASI 300 is too slow to handle very long lines, too many tab characters, or long strings with no blanks and no identical characters. One null character is inserted in a line for every set of **t** tabs, and for every contiguous string of **c** nonblank, nontab characters. When a line is longer than **len** bytes, several nulls (the line length divided by 20, plus one) are inserted at the end of that line. In all three cases, the nulls delay the output enough to avoid a problem. Items can be omitted from the end of the list, implying the default values. Entering zero for **t** results in insertion of two null bytes per tab, while entering zero for **c** results in insertion of two null bytes per character.

When printing C Language programs, using **-d0,1** helps adjust for the many indentation levels. When printing files like **/etc/passwd**, using **-d3,30,5** helps print it properly.

This flag affects carriage return and line feed delays. The **stty** parameters **nl0 cr2** or **nl0 cr3** are recommended for most uses.

-num Controls the size of half-line spacing. The default half-line values (which are exact half-lines) of **num** are:

10-pitch, 6 lines-per-inch, num=4
12-pitch, 8 lines-per-inch, num=3
12-pitch, 6 lines-per-inch, num=4

You can use other values for **num** to change the appearance of subscripts and superscripts. For example, **-2** makes **nroff** half-lines act like quarter-lines.

+12 Uses 12-pitch, 6 lines-per-inch text. The DASI 300 normally allows only two combinations: 10-pitch, 6 lines per inch, or 12-pitch, 8 lines per inch. To use the 12-pitch, 6 lines-per-inch combination, set the PITCH switch to **12** and use the **+12** flag on the command line.

Related Information

See the following commands: "450" in topic 1.1.561, "eqn, neqn, checkeq" in topic 1.1.152, "graph" in topic 1.1.191, "nroff, troff" in topic 1.1.301, "tbl" in topic 1.1.463, and "tplot" in topic 1.1.477.

See the **Greek** miscellaneous facility in *AIX Operating System Technical Reference*.

1.1.560 4014

Purpose
Formats a full page 66-line screen display for a Tektronix 4014 work station.

Syntax

```
        +------------+   +-------- -p66 --------+   +--------+
4014 ---¦ +--------+ +---¦           +-- l ---+ +---¦        +---¦
        +-¦ -t     +-+   +- -p num --¦ one of +-+   +- file -+
          ¦ -c num ¦¦                ¦ +---+  ¦
         ¦¦ -n     ¦¦                +-¦ l +--+
         ¦+--------+¦                  ¦ i ¦
         +----------+                  +---+
```

Note: This command does not have MBCS support.

Description
The **4014** command reads a **file** (standard input by default) and writes a 66-line page display to standard output. It also divides the screen into a specified number of columns, adding an eight space page offset when it uses the default single column format. It interprets tabs, spaces, backspaces, and TELETYPE Model 37 half-line and reverse-line sequences correctly. At the end of each page, **4014** waits for a line feed from the keyboard before continuing. While the **4014** command is waiting, you can send commands to the shell by entering !**AIX-cmd**, where **AIX-cmd** is a AIX command.

Flags

-cnum	Divides the screen into **num** columns and waits after the last column. The default is a single, full page-width column.
-n	Starts displaying at the current cursor position and does not erase the screen.
-pnuml	
-pnumi	Sets page length to **num** lines (**l**, the default) or to **num** inches (**i**).
-t	Does not wait between pages.

Related Information

See the following commands: "pr" in topic 1.1.322, "tc" in topic 1.1.464, and "troff" in topic 1.1.302.2.

1.1.561 450

Purpose
Handles special line-motion functions for the DASI 450 work station.

Syntax

450 ---¦

Note: This command does not have MBCS support.

Description
The **450** command reads standard input, processes its data for output on a DASI 450 or an equivalent work station (such as the DIABLO 1620 or Xerox 1700. It converts half-line forward, half-line reverse, and full-line reverse motions to the correct vertical motions on standard output. It attempts to draw Greek characters and other special symbols in the same manner as the **300** command vertical line space. See "greek" in topic 1.1.192 for a list of symbols supported by the **450** command.

Use **450** with the **nroff -s** flag or **.rd** requests when you need to insert paper manually or change fonts in the middle of a document. Instead of using the return key in these cases, you must use a the line feed key to get any response. In many cases you can use **nroff -T450** instead of the **450** command. However, you must use **450** if you require special delays or options. In a few cases, using **450** may produce better aligned output. You can pipe the output of the **neqn** command to **450** to print equations neatly.

Notes:

1. Make sure the **PLOT** switch is turned on before using this command. Also, the **SPACING** switch should be in the desired position, either 10- or 12-pitch. For either setting, vertical spacing is 6 lines per inch unless changed to 8 lines per inch by an escape sequence.

2. Some special characters cannot be correctly printed in column 1 because the print head cannot be moved to the left from that position.

3. If your output contains Greek characters or reverse linefeeds, use a friction-feed platen instead of a forms tractor. A forms tractor tends to slip when reversing direction.

Flag

-f	Permits the use of ETX/ACK protocol with 1200 bps printers. You cannot use **450** with this flag in a pipeline or if you redirect its output. Instead it must drive the printer directly.

Related Information

See the following commands: "300, 300s" in topic 1.1.559, "eqn, neqn, checkeq" in topic 1.1.152, "graph" in topic 1.1.191, "greek" in topic 1.1.192, "nroff, troff" in topic 1.1.301, "tabs" in topic 1.1.459, "tbl" in topic 1.1.463, "tplot" in topic 1.1.477, and "troff" in topic 1.1.302.2.

See the **Greek** miscellaneous facility in *AIX Operating System Technical Reference*.

A.0 Appendix A. AIX Device Table

AIX standard devices are special files. Table A-1 lists and describes some special files. For more detailed information on special files and those that are AIX/370 specific, see *AIX Operating System Technical Reference*.

Table A-1. AIX Standard Devices (Special Files)

Special File	**Description**
appltrace	Application trace pseudo device driver
console	Console device
error	Error-logging interface
fd[*num*]	Diskette drive, block device
hd[*num*]	Fixed disk drive, block device
hft	High function terminal
kmem	Kernel memory image
lp[*num*]	Line printer
mem	Memory image
null	The null device
nvram	Non-volatile memory image
osm	System message interface
prf	AIX Operating System profiler
rfd[*num*]	Diskette drive, raw device
rhd[*num*]	Fixed disk drive, raw device
rmt[*num*]	Streaming tape
termio	General terminal interface
tty[*num*]	Controlling terminal interface
unixtrace	Kernel trace event pseudo device driver

B.0 Appendix B. Task Index

This index groups commands by task. Each command listing includes a command, a page reference, and a description of the command. Commands are grouped under the following tasks:

Subtopics

B.1 Managing the System

Subtopics

B.1.1 Installing and Maintaining Programs

install	1.1.209	Installs a command.
installp	1.1.212	Installs a Licensed Program Product (LPP).
make	1.1.254	Maintains up-to-date versions of programs.
mdrc	1.1.261	Allows you to reinstall a user-created minidisk after you have reinstalled AIX.
savecore	1.1.408	Saves a core dump of the operating system.
updatep	1.1.499	Updates one or more Licensed Program Products (LPPs).

B.1.2 Configuring the System

adduser	1.1.15	Adds, deletes and changes user and group information.
clear	1.1.73	Clears the screen.
config	1.1.88	Builds kernel configuration modules from system configuration files.
devices	1.1.119	Adds, deletes, changes and displays device information.
defkey	1.1.115	Defines keyboard key assignments.
display	1.1.130	Sets colors and fonts on the current virtual terminal.
dmft	1.1.122	Writes a label, a vtoc and a boot block on a 370 disk.
env, printenv	1.1.151	Sets the environment for execution of a command.
getty	1.1.189	Sets the characteristics of ports.
hftinit	1.1.199	Initialize the default keyboard map and display model.
help	1.1.198	Provides information to new users.
init	1.1.208	Initializes the system.
keyboard	1.1.220	Controls the delay and repetition rates of the keyboard.
loadserver	1.1.237	Local site load daemon.
locator	1.1.238	Controls the sample rate of the locator.
logger	1.1.240	Provides a program interface to the syslog system log module.
minidisks	1.1.266	Adds, deletes, changes and displays minidisks.
mknod	1.1.271	Creates a special file.
more	1.1.277	Acts as the perusal filter for screen viewing.
netparams	1.1.287	Sets cluster networking parameters.
clusterstart, clusterstop	1.1.75	Enables/disables cluster communication.
newkernel	1.1.291	Configures, builds and installs new AIX kernels.
osconfig	1.1.308	Installs AIX device drivers.
pagesize	1.1.311	Prints system page size.
pdisable, phold	1.1.314	Kills the logger running on the specified port.
pstart, penable, pshare, pdelay	1.1.338	Enables or reports the availability of login ports.
rc	1.1.354	Performs normal startup initialization.
rup	1.1.400	Displays the host status of local machines when NFS is installed.
sound	1.1.435	Controls the volume and click of the keyboard speaker.
splp	1.1.439	Changes or displays printer driver settings.
stty	1.1.447	Sets, resets, or reports work station operating parameters.
swapon	1.1.451	Specifies additional devices for paging and swapping.
termdef	1.1.468	Queries terminal characteristics.
tset	1.1.485	Initializes terminal for new user.
whatis	1.1.532	Describes a command.

B.1.3 Controlling System Security

adduser	1.1.15	Adds, deletes and changes user and group information.
chgrp	1.1.65	Changes the group ownership of a file or directory.
chmod	1.1.67	Changes permission codes.
chown	1.1.68	Changes the owner of files or directories.
id	1.1.205	Displays the system identity of the user issuing the command.
last	1.1.223	Looks at logins and logouts for information about users.
lock	1.1.239	Requests a password from the user and verifies.
logname	1.1.242	Displays your login name.
login	1.1.241	Allows you to sign on to the system.
makekey	1.1.256	Generates an encryption key.
newgrp	1.1.290	Changes your primary group identification.
passwd	1.1.312	Changes your login password.
pwck	1.1.343	Checks the password and group files for inconsistencies.
pwgmerge	1.1.346	Merges password and group files from mapping tables.
rrestore	1.1.395	Restores a file system dump across the network.
su	1.1.449	Obtains the privileges of another user, including superuser authority.
umask	1.1.490	Displays and sets file-creation permission code mask.
vipw	1.1.523	Edits the password file.
w	1.1.528	Prints who is on the system and what they are doing.

B.1.4 Backing Up and Restoring System Files

backup	1.1.32	Backs up files.
pack	1.1.309	Compresses files.
restore	1.1.371	Copies back files created by the **backup** command.
rmt	1.1.381	Remote magtape protocol module.
tar	1.1.462	Manipulates archives.

B.1.5 Managing File Systems

basename	1.1.35	Returns the base name of a string parameter.
chfstore	1.1.64	Changes file storage attribute for file in system replicated file systems.
chkfstore	1.1.66	Checks fstore values.
chroot	1.1.70	Changes the root directory of a command.
clri	1.1.74	Clears the specified inode.
comlist	1.1.82	Finds recently committed files.
cpio	1.1.93	Copies files into and out of archive storage and directories.
crash	1.1.96	Examines system images.
dcopy	1.1.113	Copies file systems for the best access time.
devnm	1.1.120	Names a device.
df	1.1.121	Reports number of available disk blocks.
dumpfs	1.1.145	Dumps file system information.
env, printenv	1.1.151	Sets the environment for execution of a command.
ff	1.1.163	Lists the file names and statistics for a file system.
fsck, dfsck	1.1.177	Checks file system consistency and interactively repairs the file system.
fsdb	1.1.178	Debugs file systems.
istat	1.1.218	Examines inodes.
link	1.1.232	Performs a link or unlink system call.
makemotd	1.1.257	Makes the system message of the day on multiple sites.
man	1.1.258	Displays manual entries online.
mkfs	1.1.269	Makes a file system.
mknod	1.1.271	Creates a special file.
mount	1.1.278	Makes a file system available for use.
rpc.mountd	1.1.386	Answers AIX Network File System mount requests.
ncheck	1.1.286	Generates path names from i-numbers.
nfsd	1.1.294	Starts the daemons that handle client network file system requests.
primrec	1.1.325	A user-level replication reconciliation procedure.
proto	1.1.335	Constructs a prototype file for a file system.
pwgmap	1.1.345	Maps password and group files into a table.
quot	1.1.349	Summarizes file system ownership.
rdf	1.1.360	Reports free space information on replicated file systems.
rdrdaemon	1.1.361	Handles (File Transfer) spool files in the reader queue.
rdump	1.1.362	File system dump across the network.
recmstr	1.1.364	Recovery daemon.
rmf	1.1.379	Removes a folder.
rpc.sprayd	1.1.392	AIX Network File System daemon.
setmnt	1.1.419	Creates a mount table.
skulker	1.1.428	Cleans up file systems by removing unwanted files.
sync	1.1.453	Updates the superblock and writes buffered files to the fixed disk.
umount, unmount	1.1.491	Unmounts a previously mounted file system, directory, or file system.
where	1.1.534	Displays file storage locations.
whereis	1.1.535	Locates source, binary and/or manual for program.
which	1.1.536	Locates an executable program using "path".
whoami	1.1.538	Prints the effective current user ID.

B.1.6 Analyzing System Activity

ac	1.1.5	Produces a printout of user logins.
dmesg	1.1.132	Collects system diagnostic messages to form an error log.
du	1.1.141	Summarizes disk usage.
errdemon	1.1.154	Starts the error-logging demon.
errpt	1.1.155	Processes a report of logged errors.
errstop	1.1.156	Terminates the error-logging demon.
errupdate	1.1.157	Updates an error report template.
fast, fastsite	1.1.161	Finds the least loaded site in the network.
fuser	1.1.182	Identifies processes using a file or file structure.
loads	1.1.236	Displays the load average of each site in the network.
portmap	1.1.319	Maps program numbers to port numbers to make service calls.
prev	1.1.323	Prints out the environment.
quota	1.1.350	Displays disc usage and limits.
quotacheck	1.1.351	File system quota consistency checker.
quotaon	1.1.352	Turns file system quotas on or off.
time	1.1.471	Times the execution of a command.
trace	1.1.480	Starts the trace function.
trcrpt	1.1.481	Formats a report from the trace log file.
trcstop	1.1.482	Stops the trace function.
trcupdate	1.1.483	Updates trace format templates.

B.1.7 Performing System Accounting Functions

acct/*	1.1.6	Provides accounting shell procedures.
acctcms	1.1.7	Produces command usage summaries from accounting records.
acctcom	1.1.8	Displays selected process accounting record summaries.
acctcon	1.1.9	Performs connect-time accounting.
acctdisk, acctdusg	1.1.10	Performs disk-usage accounting.
acctmerg	1.1.11	Merges total accounting files.
diskusg	1.1.129	Generates disk accounting data by user ID.
fwtmp	1.1.183	Manipulates connect accounting records.
runacct	1.1.398	Runs daily accounting.
sadc	1.1.405	Provides a system activity report package.

B.2 Using the System

Subtopics

B.2.1 Starting and Stopping the System

actmngr	1.1.13	Lets you interact with multiple virtual terminals.
fastboot	1.1.162	Reboots/halts the system without checking the disks.
halt	1.1.195	Stops the processor.
login	1.1.241	Allows you to sign on to the system.
open	1.1.307	Opens a virtual terminal.
passwd	1.1.312	Changes your login password.
rc	1.1.354	Initializes normal system startup.
script	1.1.412	Makes typescript of terminal session.
setmaps	1.1.418	Sets terminal maps.
shutdown	1.1.425	Ends system operation.

B.2.2 Using Shells and Interfaces

actmngr	1.1.13	Lets you interact with multiple virtual terminals.
anet	1.1.18	Runs a command on all sites in the cluster.
commit	1.1.84	Commits a file within a shell procedure.
csh	1.1.100	Interprets commands read from a file or entered at the keyboard.
sh	1.1.420	Interprets commands read from a file or entered at the keyboard.

B.2.3 Displaying System Statistics and Information

date	1.1.110	Displays or sets the date.
devices	1.1.119	Adds, deletes, changes and displays device information.
diskusg	1.1.129	Generates disk accounting data by user ID.
edquota	1.1.150	Edits user quotas.
errpt	1.1.155	Processes a report of logged errors.
errupdate	1.1.157	Updates an error report template.
file	1.1.164	Determines file type.
fuser	1.1.182	Identifies processes using a file or file structure.
groups	1.1.194	Displays your group membership.
help	1.1.411	Provides information for the new user.
id	1.1.205	Displays the system identity of the user issuing the command.
ipcs	1.1.217	Reports inter-process communication facility status.
istat	1.1.218	Examines inodes.
lastcomm	1.1.224	Prints information about commands recorded in accounting file lifetime.
learn	1.1.227	Gives Computer Aides Instruction courses.
lnetstat	1.1.235	Provides network statistics.
logname	1.1.242	Displays your login name.
minidisks	1.1.266	Adds, deletes, changes and displays minidisks.
mkst	1.1.273	Creates files of error messages.
ncheck	1.1.286	Generates path names from i-numbers.
news	1.1.292	Writes system news items to standard output.
od	1.1.304	Writes the contents of storage to the standard output.
ps	1.1.337	Reports process status.
pstart, penable, pshare, pdelay	1.1.338	Enables or reports the availability of login ports.
pwck	1.1.343	Checks the password and group files for inconsistencies.
pwd	1.1.344	Displays the path name of the working directory.
rpc.statd	1.1.393	Returns NFS performance statistics from the kernel.
sact	1.1.404	Displays current SCCS file editing status.
sadc	1.1.405	Provides a system activity report package.
sag	1.1.406	Displays a graph of system activity.
sar	1.1.407	Collects, reports, or saves system activity information.
splp	1.1.439	Changes or displays printer driver settings.
stty	1.1.447	Sets, resets, or reports work station operating parameters.
sum	1.1.450	Displays the checksum and block count of a file.
time	1.1.471	Times the execution of a command.
timex	1.1.472	Times a command and reports process data and system activity.
tty	1.1.488	Writes to standard output the full path name of your work station.
uname	1.1.492	Displays the name of the current operating system.
uustat	1.1.512	Reports the status of and provides rudimentary job control for BNU commands.
who	1.1.537	Identifies the users currently logged in.

B.2.4 Controlling System Processes

ap	1.1.20	Parses and reformats addresses.
apply	1.1.21	Runs a named command on each argument in turn.
apropos	1.1.22	Shows which manual sections contain keywords.
at	1.1.26	Runs commands at a later time
atq	1.1.27	Prints the queue of jobs that are waiting to be run.
atrm	1.1.28	Removes jobs that were created with the at command.
cron	1.1.97	Runs commands automatically.
crontab	1.1.98	Submits a schedule of commands to **cron**.
errdemon	1.1.154	Starts the error-logging demon.
errstop	1.1.156	Terminates the error-logging demon.
kill	1.1.221	Sends a signal to a running process.
killall	1.1.222	Cancels all processes except the calling process.
nice	1.1.296	Runs a command at a different priority.
nohup	1.1.300	Runs a command without hangups and quits.
on	1.1.306	Runs a command at a specific site.
open	1.1.307	Opens a virtual terminal.
printlocal	1.1.327	Displays the <LOCAL> alias of the current process.
printspath	1.1.328	Displays the site path of the current process.
probe	1.1.330	Determines if a site is up.
prober	1.1.331	Asynchronous probe daemon.
qdaemon	1.1.347	Schedules jobs enqueued by the print command.
rpcgen	1.1.383	Compiles a Remote Procedure Call program if NFS is installed.
rpcinfo	1.1.384	Reports a Remote Procedure Call program if NFS is installed.
rusers	1.1.401	Identifies users logged in on local machines if NFS is installed.
rpc.usersd	1.1.390	Displays list of active AIX Network File System users.
rpc.walld	1.1.391	Handles requests for the rwall and shutdown commands.
sleep	1.1.429	Suspends execution for an interval.
tlog	1.1.474	Starts and stops sending of I/O data.
tlogger	1.1.475	Writes data to a log file.

B.2.5 Using Disks and Diskettes

flcopy	1.1.167	Copies diskettes.
format, **fdformat**	1.1.172	Formats diskettes.
mdrc	1.1.261	Allows you to reinstall a user-created minidisk after you have reinstalled AIX.
minidisks	1.1.266	Adds, deletes, changes and displays minidisks.
mount	1.1.278	Makes a file system available for use.
umount, **unmount**	1.1.491	Unmounts a previously mounted file system, directory, or file system.

B.2.6 Using Tape

B.2.7 Working with Work Stations

defkey	1.1.115	Defines keyboard key assignments.
display	1.1.130	Sets colors and fonts on the current virtual terminal.
echo	1.1.146	Writes its arguments to standard output.
hp	1.1.200	Handles special functions for the HP2640- and HP2621-series terminals.
keyboard	1.1.220	Controls the delay and repetition rates of the keyboard.
locator	1.1.238	Controls the sample rate of the locator.
pdisable, phold	1.1.314	Kills the logger running on the specified port.
pstart, penable, pshare, pdelay	1.1.338	Enables or reports the availability of login ports.
stty	1.1.447	Sets, resets, or reports work station operating parameters.
tab	1.1.458	Changes spaces into tabs or tabs into spaces.
tabs	1.1.459	Sets tab stops on work stations.
termdef	1.1.468	Queries terminal characteristics.
tic	1.1.470	Translates **terminfo** files from source to compiled format.
tput	1.1.478	Queries the **terminfo** file.
tty	1.1.488	Writes to standard output the full path name of your work station.
300	1.1.559	Handles special line-motion functions for DASI 300/300s work stations.
4014	1.1.560	Formats a full page 66-line screen display for a Tektronix 4014 work station.
450	1.1.561	Handles special line-motion functions for the DASI 450 work station.

B.3 Working with Files and Directories

Subtopics

B.3.1 Working with Directories

cd	1.1.53	Changes the current directory.
chroot	1.1.70	Changes the root directory of a command.
dircmp	1.1.127	Compares two directories and the contents of their common files.
dosdir	1.1.135	Lists the directory for DOS files.
find	1.1.165	Finds files matching expression.
li	1.1.230	Lists the contents of a directory.
ls, lf, lr	1.1.252	Displays the contents of a directory.
mklo	1.1.270	Creates a directory with empty files for fsck commands.
mkdir	1.1.268	Makes a directory.
mvdir	1.1.283	Moves (renames) a directory.
pwd	1.1.344	Displays the path name of the working directory.
rm	1.1.375	Removes files or directories.
rmdir	1.1.378	Removes a directory.
shlibrpt	1.1.421	Generates a report file that aids in constructing a shared library.
sortbib	1.1.433	Sorts bibliographic data base.
strings	1.1.444	Finds the printable strings in an object or other binary file.

B.3.2 Creating and Editing Files

addbib	1.1.14	Creates a bibliography.
admin	1.1.16	Creates and initializes SCCS files.
catman	1.1.50	Creates the cat files for the manual
checknr	1.1.63	Checks nroff/troff files for errors.
cdc	1.1.54	Changes the comments in a Source Code Control System (SCCS) delta.
colcrt	1.1.78	Provides virtual line feeds.
colpro	1.1.79	Column filter for IBM 4201 Proprinter.
colrm	1.1.80	Extracts columns from a file.
compress	1.1.86	Reduces the size of named files.
diction	1.1.123	Identifies and prints wordy sentences.
domain name	1.1.133	Sets or displays the name of the current NIS domain.
ed	1.1.147	Edits text by line.
edit	1.1.149	Provides a simple line editor for the new user.
ex	1.1.158	Edits lines interactively with screen display.
get	1.1.186	Creates a specified version of a Source Code Control System (SCCS) file.
head	1.1.197	Prints the first few lines of a file or files.
lookbib,indxbi	1.1.244	Makes an inverted index.
makedbm	1.1.255	Makes a NIS dbm file.
mknod	1.1.271	Creates a special file.
prof	1.1.332	Displays program profile data.
refer	1.1.365	Finds and inserts literature references in documents.
rev	1.1.374	Reverses characters in a line of a file.
roffbib	1.1.382	Runs off bibliographic data base.
sed	1.1.415	Provides a stream editor.
spell	1.1.436	Finds spelling errors.
style	1.1.448	Analyzes the surface characteristics of a document.
symorder	1.1.452	Rearranges a name list.
tab	1.1.458	Changes spaces into tabs or tabs into spaces.
ul	1.1.489	Does underlining.
unifdef	1.1.494	Removed "ifdef'ed" lines.
uniq	1.1.495	Deletes repeated lines in a file.
vi	1.1.522	Edits files with a full screen display.
xstr	1.1.546	Maintains a file "strings".
ypcat	1.1.549	Displays the values in an NIS data base.
ypmatch	1.1.551	Prints the value of one or more keys from an NIS map.
yppasswd	1.1.552	Changes login password in NIS.
yppoll	1.1.553	Discovers which version of an NIS map is at an NIS server host.
yppush	1.1.554	Forces propagation of a changed NIS map.
ypserv, ypbind	1.1.555	NIS server and binder processes.
ypset	1.1.556	Points ypbind at a particular server.
ypwhich	1.1.557	Tells which host is the NIS server or map master.
ypxfr	1.1.558	Transfers an NIS map from some NIS server.

B.3.3 Printing and Displaying Files

cat	1.1.49	Concatenates or displays files.
cut	1.1.107	Writes out selected fields from each line of a file.
lp, cancel	1.1.246	Prints a file in a format suitable for sending to a line printer.
lpr	1.1.248	Uses a spooling daemon to print named files.
nl	1.1.297	Numbers lines in a file.
od	1.1.304	Writes the contents of storage to the standard output.
pg	1.1.315	Formats files to the workstation.
piobe	1.1.317	Writes a file to standard output in a format suitable for sending to a line printer.
pr	1.1.322	Writes a file to standard output.
prs	1.1.336	Displays a Source Code Control System (SCCS) file.
print	1.1.326	Enqueues a file.
punbkend	1.1.341	Punch backend (file transfer).
qdaemon	1.1.347	Schedules jobs enqueued by the print command.
splp	1.1.439	Changes or displays printer driver settings.
tail	1.1.460	Writes a file to standard output, beginning at a specified point.
vc	1.1.520	Substitutes assigned values in place of keywords.

B.3.4 Copying and Moving Files

cat	1.1.49	Concatenates or displays files.
cp	1.1.91	Copies files.
dd	1.1.114	Converts and copies a file.
dosread	1.1.136	Copies a DOS file.
doswrite	1.1.137	Copies AIX files to DOS files.
ln	1.1.234	Links files.
mv	1.1.282	Moves files.
uucp	1.1.506	Copies files from one AIX system to another AIX system.
uuto	1.1.513	Copies public files from one AIX system to another AIX system, with local system control of file access.

B.3.5 Deleting Files

del	1.1.116	Deletes files if the request is confirmed.
dosdel	1.1.134	Deletes DOS files.
rm	1.1.375	Removes files or directories.
uniq	1.1.495	Deletes repeated lines in a file.
uucleanup	1.1.504	Deletes selected files older than a specified number of hours from the BNU spool directory or a named directory.

B.3.6 Comparing Files

bdiff	1.1.37	Uses **diff** to find differences in very large files.
cmp	1.1.76	Compares two files.
comm	1.1.83	Selects or rejects lines common to two sorted files.
diff	1.1.124	Compares text files.
diff3	1.1.126	Compares three files.
diffmk	1.1.125	Marks differences between files.
dircmp	1.1.127	Compares two directories and the contents of their common files.
sdiff	1.1.413	Compares two files and displays the differences in a side-by-side format.
sccsdiff	1.1.410	Compares two files and displays the differences in a side-by-side format.

B.3.7 Scanning Files

awk	1.1.29	Finds lines in files matching specified patterns and performs specified actions on them.
bfs	1.1.39	Scans files.
file	1.1.164	Determines file type.
find	1.1.165	Finds files matching expression.
grep	1.1.193	Searches a file for a pattern.
hyphen	1.1.201	Finds hyphenated words.
wc	1.1.530	Counts the number of lines, words and characters in a file.
what	1.1.531	Displays identifying information in files.

B.3.8 Sorting Files

looke	1.1.243	Consults a sorted file and prints all lines that begin with **string**.
lorder	1.1.245	Finds the best order for member files in an object library.
sort	1.1.432	Sorts files.
tsort	1.1.486	Sorts an unordered list of ordered pairs (a topological sort).

B.3.9 Merging and Splitting Files

csplit	1.1.101	Splits files by context.
join	1.1.219	Joins data fields of two files.
paste	1.1.313	Merges the lines of several files or subsequent lines in one file.
sort	1.1.432	Sorts files.
split	1.1.438	Splits files into pieces.

B.3.10 Formatting Text

col	1.1.77	Column filter for IBM 4201 Proprinter.
cw	1.1.108	Prepares constant-width text for **troff**.
deroff	1.1.118	Removes **nroff, troff**, **troff**, **tbl**, and **eqn** constructs from files.
eqn	1.1.152	Formats mathematical text for the **nroff, troff** commands.
fmt	1.1.168	Formats mail messages prior to sending.
greek	1.1.192	Converts output for a TELETYPE Model 37 workstation to output for other workstations.
hp	1.1.200	Handles special functions for the HP2640- and HP2621-series terminals.
mm	1.1.274	Displays or checks documents formatted with memorandum macros.
mmt	1.1.275	Typesets documents, manual pages, view graphs and slides.
newform	1.1.288	Changes the format of a text file.
nl	1.1.297	Numbers lines in a file.
nroff, troff	1.1.301	Formats text for printing devices.
ptx	1.1.340	Generates a permuted index.
tbl	1.1.463	Formats tables for the **nroff, troff** commands.
tc	1.1.464	Simulates phototypesetter output for a Tektronix 4014 workstation.
vrm2rtfont	1.1.525	Converts a standard AIX font file to RT PC rtx font format.
300	1.1.559	Handles special line-motion functions for DASI 300/300s work stations.
4014	1.1.560	Formats a full page 66-line screen display for a Tektronix 4014 work station.
450	1.1.561	Handles special line-motion functions for the DASI 450 work station.

B.3.11 Working with Graphics

spline	1.1.437	Interpolates smooth curve.
tplot	1.1.477	Produces plotting instruction for a particular work station.

B.3.12 Protecting Files with File Permissions

chgrp	1.1.65	Changes the group ownership of a file or directory.
chmod	1.1.67	Changes permission codes.
chown	1.1.68	Changes the owner of files or directories.
groups	1.1.194	Displays your group membership.
li	1.1.230	Lists the contents of a directory.
ls, lf, lr	1.1.252	Displays the contents of a directory.
umask	1.1.490	Displays and sets file-creation permission code mask.

B.3.13 Backing Up and Restoring Files

ar	1.1.23	Maintains portable libraries used by the linkage editor.
backup	1.1.32	Backs up files.
cpio	1.1.93	Copies files into and out of archive storage and directories.
lorder	1.1.245	Finds the best order for member files in an object library.
pack	1.1.309	Compresses files.
restore	1.1.371	Copies back files created by the **backup** command.
shlib2	1.1.422	Creates a shared library.
tar	1.1.462	Manipulates archives.

B.3.14 Using Data Tools

banner	1.1.33	Writes character strings in large letters to standard output.
cal	1.1.47	Displays a calendar.
calendar	1.1.48	Writes reminder messages to standard output.
ctab	1.1.103	Produces a collating table.
echo	1.1.146	Writes its arguments to standard output.
tr	1.1.479	Translates characters.
units	1.1.496	Converts units in one measure to equivalent units in another.

B.3.15 Performing Calculator Functions

bc	1.1.36	Provides an interpreter for arbitrary-precision arithmetic language.
dc	1.1.112	Provides an interactive desk calculator for doing arbitrary-precision integer arithmetic.
factor	1.1.160	Factors a number.

B.3.16 Sending Messages and Notices

mesg	1.1.262	Permits or refuses **write** messages.
news	1.1.292	Writes system news items to standard output.
repl	1.1.369	Replies to a message.
vacation	1.1.518	Returns "I am on vacation" indication.
wall	1.1.529	Writes a message to all logged-in users.
write	1.1.541	Sends messages to other users on the system.

B.3.17 Working with Messages

dspcat	1.1.139	Displays messages within a catalog.
dspmsg	1.1.140	Displays a message from a message catalog.
gencat	1.1.184	Generates a message catalog.
gettext	1.1.188	Extracts message/insert/help descriptions
mkcatdefs	1.1.267	Processes symbolic message identifiers in message text source files.
msgs	1.1.280	Reads system messages.
puttext	1.1.342	Updates an output file that contains message/insert/help descriptions
refile	1.1.366	Files messages in other folders.
runcat	1.1.399	Creates a message catalog using symbolic identifiers.
sortm	1.1.434	Sorts messages.
spost	1.1.440	Delivers a message.

B.3.18 Using Mailboxes

biff	1.1.41	Informs the system to notify user when mail arrives.
bellmail	1.1.38	Sends messages to system users and displays messages from system users.
comsat	1.1.87	Receives reports of incoming mail.
edconfig	1.1.148	Edits values in a sendmail configuration file.
fmt	1.1.168	Formats mail messages prior to sending.
mail	1.1.253	Sends and receives mail.
rmail	1.1.376	Handles remote mail received via UUCP.
sendmail	1.1.417	Routes mail for local or network delivery.

B.3.19 Using the Message Handling (MH) Package

ali	1.1.17	Lists mail aliases and their addresses.
anno	1.1.19	Annotates messages.
burst	1.1.46	Explodes digests into messages.
comp	1.1.85	Composes a message.
conflict	1.1.89	Searches for alias and password conflicts.
dist	1.1.131	Redistributes a message to additional addresses.
dp	1.1.138	Parses and reformats dates.
folder	1.1.170	Selects and lists folders and messages.
forw	1.1.174	Forwards messages.
inc	1.1.206	Incorporates new mail.
install-mh	1.1.211	Initializes the MH environment.
mark	1.1.259	Creates, modifies and displays message sequences.
mhl	1.1.263	Produces formatted listings of messages.
mhmail	1.1.264	Sends or receives mail.
mhpath	1.1.265	Prints full path names of messages and folders.
msgchk	1.1.279	Checks for messages.
msh	1.1.281	Creates a **msh** shell.
next	1.1.293	Shows the next message.
packf	1.1.310	Compresses the contents of a folder into a file.
pick	1.1.316	Selects messages by content and creates and modifies sequences.
post	1.1.320	Delivers a message.
prev	1.1.323	Shows the previous message.
prompter	1.1.334	Invokes a prompting editor.
rcvdist	1.1.355	Sends a copy of incoming messages to additional recipients.
rcvpack	1.1.356	Saves incoming messages in a packed file.
rcvstore	1.1.357	Incorporates new mail from standard input into a folder.
rcvtty	1.1.358	Notifies the user of incoming messages.
refile	1.1.366	Files messages in other folders.
repl	1.1.369	Replies to a message.
rmf	1.1.379	Removes a folder.
rmm	1.1.380	Removes messages.
scan	1.1.409	Produces a one-line-per-message scan listing.
send	1.1.416	Sends a message.
show	1.1.423	Shows messages.
sortm	1.1.434	Sorts messages.
vmh	1.1.524	Invokes a visual interface for use with MH commands.
whatnow	1.1.533	Invokes a prompting interface for draft disposition.
whom	1.1.539	Lists and verifies the addresses of the proposed recipients of a message.

B.3.20 Communicating with Other Systems

connect	1.1.90	Establishes a connection to a remote system.
ct	1.1.102	Dials an attached terminal and issues a login process.
cu	1.1.105	Connects directly or indirectly to another UNIX system.
rwall	1.1.402	Writes to all users over a network when NFS is installed.
sum	1.1.450	Displays the checksum and block count of a file.
sysline.	1.1.456	Displays system status on the status line of a terminal.
syslogd	1.1.457	Logs system messages.
uucheck	1.1.502	Checks for files and directories required by BNU.
uucico	1.1.503	File transport program for the BNU facility.
uucleanup	1.1.504	Deletes selected files older than a specified number of hours from the BNU spool directory or a named directory.
uucp	1.1.506	Copies files from one AIX system to another AIX system.
uulog	1.1.508	Provides information about UUCP and **uux** activities on a system.
uuname	1.1.509	Provides information about other systems accessible to the local system.
uupick	1.1.510	Accepts or rejects files transmitted to a user.
uusched	1.1.511	Schedules work for the BNU file transport program.
uustat	1.1.512	Reports the status of and provides rudimentary job control for BNU commands.
uuto	1.1.513	Copies public files from one AIX system to another AIX system, with local system control of file access.
uutry, Uutry, uukick	1.1.514	Contacts a remote system with debugging turned on.
uux	1.1.515	Runs a command on another AIX system.
uuxqt	1.1.516	Executes remote command requests.

B.4 Developing Programs

Subtopics

B.4.1 Programming in Assembler

as	1.1.25	Assembles a source file.
dis	1.1.128	Produces Assembler language listing from compiler programs.

B.4.2 Programming in C

cb	1.1.51	Puts C source code into a form that is read easily.
cc	1.1.52	Compiles C programs.
cflow	1.1.62	Generates a C flow graph of external references.
cpp	1.1.94	Performs file inclusion and macro substitution on C Language source files.
cxref	1.1.109	Creates a C program cross-reference listing.
factor	1.1.160	Factors a number.
gprof	1.1.190	Produces an execution profile of C, Pascal, or FORTRAN77 programs.
ipcrm	1.1.216	Removes message queue, semaphore set or shared memory identifiers.
m4	1.1.284	Preprocesses files, expanding macro definitions.
lex	1.1.229	Generates a C Language program that matches patterns for simple lexical analysis of an input stream.
lint	1.1.233	Checks C programs for potential problems.
regcmp	1.1.367	Compiles patterns.
tic	1.1.470	Translates **terminfo** files from source to compiled format.
yacc	1.1.547	Generates an LR parsing program from input consisting of a context-free grammar specification.

B.4.3 Programming in Miscellaneous Languages

bc	1.1.36	Provides an interpreter for arbitrary-precision arithmetic language.
bs	1.1.44	Compiles and interprets modest-sized programs.
ctags	1.1.104	Makes a file of tags to help locate objects in source files.
fpr	1.1.175	Prints a FORTRAN file.
m4	1.1.284	Preprocesses files, expanding macro definitions.
struct	1.1.446	Structures FORTRAN programs.
vs	1.1.526	Compiles C, VS Pascal and VS FORTRAN programs.

B.4.4 Programming in Shell

basename	1.1.35	Returns the base name of a string parameter.
cron	1.1.97	Runs commands automatically.
crontab	1.1.98	Submits a schedule of commands to **cron**.
csh	1.1.100	Interprets commands read from a file or entered from the keyboard.
echo	1.1.146	Writes its arguments to standard output.
env, printenv	1.1.151	Sets the environment for execution of a command.
expr	1.1.159	Evaluates arguments as an expression.
find	1.1.165	Finds files matching expression.
getopt	1.1.187	Parses command line flags and parameters.
line	1.1.231	Reads one line from the standard input.
nice	1.1.296	Runs a command at a different priority.
nohup	1.1.300	Runs a command without hangups and quits.
open	1.1.307	Opens a virtual terminal.
sh	1.1.420	Interprets commands read from a file or input from the keyboard.
sleep	1.1.429	Suspends execution for an interval.
tee	1.1.467	Displays the output of a program and copies it into a file.
test	1.1.469	Evaluates conditional expressions.
time	1.1.471	Times the execution of a command.
true	1.1.484	Returns an exit value of zero (true) or non-zero (false).
xargs	1.1.544	Constructs argument lists and runs commands.

B.4.5 Debugging Programs

bugfiler	1.1.45	Intercepts, summarizes and stores bug reports.
crash	1.1.96	Examines system images.
dbx	1.1.111	Provides a tool to debug and run programs under AIX.
dump	1.1.142	Dumps selected parts of an object file.
od	1.1.304	Writes the contents of storage to the standard output.
prof	1.1.332	Displays program profile data.
time	1.1.471	Times the execution of a command.
timex	1.1.472	Times a command and reports process data and system activity.

B.4.6 Managing Source Programs Using the Source Code Control System (SCCS)

admin	1.1.16	Creates and initializes SCCS files.
cdc	1.1.54	Changes the comments in an SCCS delta.
comb	1.1.81	Combines SCCS deltas.
delta	1.1.117	Creates a delta in a SCCS file.
get	1.1.186	Creates a specified version of a SCCS file.
help	1.1.411	Provides information about a SCCS message, command, or certain non-SCCS commands.
prs	1.1.336	Displays a SCCS file.
rmdel	1.1.377	Removes a delta from a SCCS file.
sact	1.1.404	Displays current SCCS file editing status.
sccsdiff	1.1.410	Compares two files and displays the differences in a side-by-side format.
unget	1.1.493	Cancels a previous **get** command.
val	1.1.519	Validates SCCS files.
what	1.1.531	Displays identifying information in files.

B.4.7 Managing Object Files

ar	1.1.23	Maintains portable libraries used by the linkage editor.
as	1.1.25	Assembles a source file.
dump	1.1.142	Dumps selected parts of an object file.
ld	1.1.226	Links object files.
lorder	1.1.245	Finds the best order for member files in an object library.
make	1.1.254	Maintains up-to-date versions of programs.
nm	1.1.298	Displays the symbol table of an object file.
nm -BSD Version	1.1.299	Displays the symbol table of an object file.
prof	1.1.332	Displays program profile data.
size	1.1.427	Displays the section sized of common object files.
strip	1.1.445	Removes symbol and line number information from a common object file.
touch	1.1.476	Updates the access and modification times of a file.
tsort	1.1.486	Sorts an unordered list of ordered pairs (a topological sort).

BACK_1 Glossary

access. To obtain computing services.

access permission. A group of designations that determine who can access a particular AIX file and how the user may access the file.

account. The log in directory and other information that give a user access to the system.

activity manager. A collection of system programs allowing users to manage their activities. Provides the ability to list current activities (Activity List) and to begin, cancel, hide, and activate activities.

All Points Addressable (APA) display. A display that allows each pel to be individually addressed. An APA display allows for images to be displayed that are not made up of images predefined in character boxes. Contrast with **character display**.

allocate. To assign a resource, such as a disk file or a diskette file, to perform a specific task.

alphanumeric character. Consisting of letters, numbers and often other symbols, such as punctuation marks and mathematical symbols.

American National Standard Code for Information Interchange (ASCII). The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

American National Standards Institute (ANSI). An organization sponsored by the Computer and Business Equipment Manufacturers Association for establishing voluntary industry standards.

application. A program or group of programs that directly apply to a particular user problem, such as the Inventory Control, word processing, or the Accounts Receivable application.

application program. A program used to perform an application or part of an application.

argument. Numbers, letters, or words that change the way a command works.

ASCII. See **American National Standard Code for Information Interchange**.

attribute. A characteristic. For example, the attribute for a displayed field could be blinking.

auto carrier return. The system function that places carrier returns automatically within the text and on the display. This is accomplished by moving whole words that exceed the line end zone to the next line.

backend. The program that sends output to a particular device. There are two types of backends: friendly and unfriendly.

background process. (1) A process that does not require operator intervention that can be run by the computer while the work station is used to do other work. (2) A mode of program execution in which the shell does not wait for program completion before prompting the user for another command.

backup copy. A copy, usually of a file or group of files, that is kept in case the original file or files are unintentionally changed or destroyed.

backup diskette. A diskette containing information copied from a fixed disk or from another diskette. It is used in case the original information becomes unusable.

backup format. A compressed file format. When the backup command makes a copy of a file, it writes the file in this format. A file in this format must be restored by the restore command before it can be used.

backup format file. A file in backup format.

bad block. A portion of a disk that can never be used reliably.

base address. The beginning address for resolving symbolic references to locations in storage.

base name. The last element to the right of a full path name. A filename specified without its parent directories.

batch printing. Queueing one or more documents to print as a separate job. The operator can type or revise additional documents at the same time. This is a background process.

batch processing. A processing method in which a program or programs process records with little or no operator action. This is a background process. Contrast with **interactive processing**.

binary. (1) Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. (2) Involving a choice of two conditions, such as on-off or yes-no.

bit. Either of the binary digits 0 or 1 used in computers to store information. Eight bits make a byte. See also **byte**.

block. (1) A group of records that is recorded or processed as a unit. Same as **physical record**. (2) In data communications, a group of records that is recorded, processed, or sent as a unit. (3) A physical block in AIX P/S2 is 4096 bytes long. (4) A logical block in AIX P/S2 is 4096 bytes long.

block file. A file listing the usage of blocks on a disk.

block special file. A special file that provides access to an input or output device and is capable of supporting a file system. See also **character special file**.

bootstrap. A small program that loads larger programs during system initialization. Sometimes referred to as "IPL" (Initial Program Load).

branch. In a computer program an instruction that selects one of two or more alternative sets of instructions. A conditional branch occurs only when a specified condition is met.

breakpoint. A place in a computer program, usually specified by an instruction, where execution may be interrupted by external intervention or by a monitor program.

buffer. (1) A temporary storage unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of storage, temporarily reserved for performing input or output, into which data is read, or from which data is written.

burst pages. On continuous-form paper, pages of output that can be separated at the perforations.

byte. The amount of storage required to represent one character; a byte is 8 bits.

call. (1) To activate a program or procedure at its entry point. Compare with **load**.

callout. An AIX kernel parameter establishing the maximum number of scheduled activities that can be pending simultaneously.

cancel. To end a task before it is completed.

carrier return. (1) In text data, the action causing line ending formatting to be performed at the current cursor location followed by a line advance of the cursor. Equivalent to the carriage return of a typewriter. (2) A keystroke generally indicating the end of a command line.

case sensitive. Able to distinguish between uppercase and lowercase letters.

character. A member of a set of elements that is used for the representation, organization, or control of data. Characters can be letters, digits, punctuation marks, or other symbols.

character display. A display that uses a character generator to display predefined character boxes of images (characters) on the screen. This kind of display cannot address the screen any less than one character box at a time. Contrast with **All Points Addressable display**.

character key. A keyboard key that allows the user to enter the character shown on the key. Compare with **function keys**.

character position. On a display, each location that a character or symbol can occupy.

character set. A group of characters used for a specific reason; for example, the set of characters a printer can print or a keyboard can support.

character special file. A special file that provides access to an input or output device. The character interface is used for devices that do not use block I/O. See also **block special file**.

character string. A sequence of consecutive characters.

character variable. The name of a character data item whose value may be assigned or changed while the program is running.

child. (1) Pertaining to a secured resource, either a file or library, that uses the user list of a parent resource. A child resource can have only one parent resource. (2) In the AIX Operating System, child is a **process** spawned by a parent process that shares attributes of the parent process. Contrast with **parent**.

C language. A general-purpose programming language that is the primary language of the AIX Operating System.

class. Pertaining to the I/O characteristics of a device. AIX devices are classified as block or character.

code. (1) Instructions for the computer. (2) To write instructions for the computer; to **program**. (3) A representation of a condition, such as an error code.

code segment. See **segment**.

collating sequence. The sequence in which characters are ordered within the computer for sorting, combining, or comparing.

color display. A display device capable of displaying more than two colors and the shades produced via the two colors, as opposed to a monochrome display.

column headings. Text appearing near the top of columns of data for the purpose of identifying or titling.

command. A request to perform an operation or run a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command.

command interpreter. A program (such as Bourne or C shell) that sends instructions to the kernel; also called an interface.

command line. The area of the screen where commands are displayed as they are typed.

command line editing keys. Keys for editing the command line.

command programming language. Facility that allows programming by the combination of commands rather than by writing statements in a conventional programming language. See **shell procedure**.

compile. (1) To translate a program written in a high-level programming language into a machine language program. (2) The computer actions required to transform a source file into an executable object file.

compress. (1) To move files and libraries together on disk to create one continuous area of unused space. (2) In data communications, to delete a series of duplicate characters in a character string.

concatenate. (1) To link together. (2) To join two character strings.

condition. An expression in a program or procedure that can be evaluated to a value of either true or false when the program or procedure is running.

configuration. The group of machines, devices, and programs that make up a computer system. See also **system customization**.

configuration file. A file that specifies the characteristics of a system or subsystem, for example, the AIX queueing system.

consistent. Pertaining to a file system, without internal discrepancies.

console. (1) The main AIX display station for a site. (2) A device name associated with the main AIX display station for a site.

constant. A data item with a value that does not change. Contrast with **variable**.

context search. A search through a file whose target is a character string.

control block. A storage area used by a program to hold control information.

control commands. Commands that allow conditional or looping logic flow in shell procedures.

control program. Part of the AIX Operating System system that determines the order in which basic functions should be performed.

controlled cancel. The system action that ends the job step being run and saves any new data already created. The job that is running can continue with the next job step.

crash. An unexpected interruption of computer service, usually due to a serious hardware or software malfunction.

current directory. The directory that is the starting point for relative pathnames, and can be displayed with the **pwd** command.

current line. The line on which the cursor is located.

current working directory. See **current directory**.

cursor. (1) A movable symbol (such as an underline) on a display, used to indicate to the operator where the next typed character will be placed or where the next action will be directed. (2) A marker that indicates the current data access location within a file.

cursor movement keys. The directional keys used to move the cursor.

customize. To describe (to the system) the devices, programs, users, and user defaults for a particular data processing system.

cylinder. All fixed disk or diskette tracks that can be read or written without moving the disk drive or diskette drive read/write mechanism.

daemon. See **daemon process**.

daemon process. A process begun by the root or the root shell that can be stopped only by the root. Daemon processes generally provide services that must be available at all times to more than one task or user, such as sending data to a printer.

data block. See **block**.

data communications. The transmission of data between computers, or remote devices or both (usually over long distance).

data stream. All information (data and control information) transmitted over a data link.

debug. (1) To detect, locate, and correct mistakes in a program. (2) To find the cause of problems detected in software.

default. A value that is used when no alternative is specified by the operator.

default directory. The directory name supplied by the operating system if none is specified.

default drive. The drive name supplied by the operating system if none is specified.

default value. A value stored in the system that is used when no other value is specified.

dependent work station. A work station having little or no standalone capability, that must be connected to a host or server in order to provide any meaningful capability to the user.

device. An electrical or electronic machine that is designed for a specific purpose and that attaches to your computer, for example, a printer, plotter, disk drive, and so forth.

device driver. A program that operates a specific device, such as a printer, disk drive, or display.

device name. A name reserved by the system that refers to a specific device.

diagnostic. Pertaining to the detection and isolation of an error.

diagnostic aid. A tool (procedure, program, reference manual) used to detect and isolate a device or program malfunction or error.

diagnostic routine. A computer program that recognizes, locates, and explains either a fault in equipment or a mistake in a computer program.

directory. A type of file containing the names and controlling information for other files or other directories.

disable. To make nonfunctional.

discipline. Pertaining to the order in which requests are serviced, for example, first-come-first-served (FCFS) or shortest job next (SJN).

disk I/O. Fixed-disk input and output.

diskette. A thin, flexible magnetic plate that is permanently sealed in a protective cover. It can be used to store information copies from the disk or another diskette.

diskette drive. The mechanism used to read and write information on diskettes.

display device. An output unit that gives a visual representation of data.

display screen. The part of the display device that displays information visually.

display station. A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or received from the computer.

dump. (1) To copy the contents of all or part of storage, usually to an output device. (2) Data that has been dumped.

dump diskette. A diskette that contains a dump or is prepared to receive a dump.

dump formatter. Program for analyzing a dump.

EBCDIC. See **extended binary-coded decimal interchange code**.

EBCDIC character. Any one of the symbols included in the 8-bit EBCDIC set.

edit. To modify the form or format of data.

edit buffer. A temporary storage area used by an editor.

editor. A program used to enter and modify programs, text, and other types of documents and data.

emulation. Imitation; for example, when one computer imitates the characteristics of another computer.

enable. To make functional.

enter. To send information to the computer by pressing the **Enter** key.

entry. A single input operation on a work station.

environment. The settings for shell variables and paths set associated with each process. These variables can be modified later by the user.

error-correct backspace. An editing key that performs editing based on a cursor position; the cursor is moved one position toward the beginning of the line, the character at the new cursor location is deleted, and all characters following the cursor are moved one position toward the beginning of the line (to fill the vacancy left by the deleted element).

escape character. A character that suppresses the special meaning of one or more characters that follow.

exit value. A numeric value that a command returns to indicate whether or not the command completed successfully. Some commands return exit values that give other information, such as whether a file exists. Shell programs can test exit values to control branching and looping. Exit values are also called "return codes".

expression. A representation of a value. For example, variables and constants appearing alone or in combination with operators.

extended binary-coded decimal interchange code (EBCDIC). A set of 256 eight-bit characters.

feature. A programming or hardware option, usually available at an extra cost.

field. (1) An area in a record or panel used to contain a particular category of data. (2) The smallest component of a record that can be referred to by a name.

FIFO. See **first-in-first-out**.

file. A collection of related data that is stored and retrieved by an assigned name.

file name. The name used by a program to identify a file. See **label**.

filename. In DOS, that portion of the file name that precedes the extension.

file specification (filespec). The name and location of a file. A file specification consists of a drive specifier, a path name, and a file name.

file system. A collection of files and directories stored on logical and physical devices (such as disks) and logically organized in a hierarchical fashion.

files. An AIX kernel parameter establishing the maximum number of files that can be open simultaneously.

file tree. The complete directory and file structure of a particular node, starting at the root directory. A file tree contains all local and remote mounts performed on minidisks, directories, and files.

filter. A command that reads standard input data, modifies the data, and sends it to standard output.

first-in-first-out (FIFO). A named permanent pipe. A FIFO allows two unrelated processes to exchange information using a pipe connection.

fixed disk. A flat, circular, non-removeable plate with a magnetic surface layer on which data can be stored by magnetic recording.

fixed-disk drive. The mechanism used to read and write information on fixed disk.

flag. A modifier that appears on a command line with the command name that defines the action of the command. Flags in the AIX Operating System almost always are preceded by a dash.

font. A family or assortment of characters of a given size and style.

foreground. A mode of program execution in which the shell waits for the program specified on the command line to complete before returning your prompt.

format. (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, or files. (2) The pattern which determines how data is recorded.

formatted diskette. A diskette on which control information for a particular computer system has been written but which may or may not contain any data.

free list. A list of available space on each file system. This is sometimes called the free-block list.

free-block list. See **free list**.

full install. A complete installation of AIX or other programs.

full path name. The name of any directory or file expressed as a string of directories and files beginning with the root directory.

function. A synonym for procedure. The C language treats a function as a data type that contains executable code and returns a single value to the calling function.

function keys. Keys that request actions but do not display or print characters. Included are the keys that normally produce a printed character, but when used with the code key produce a function instead. Compare with **character key**.

generation. For some remote systems, the translation of configuration information into machine language.

Gid. See **group number**.

global. Pertains to information available to more than one program or subroutine.

global action. An action having general applicability, independent of the context established by any task.

global character. The special characters * and ? that can be used in a file specification to match one or more characters. For example, placing a ? in a file specification means any character can be in that position.

global search. The process of having the system look through a document for specific characters, words, or groups of characters.

global variable. A symbol defined in one program module, but used in other independently assembled program modules.

graphic character. A character that can be displayed or printed.

group name. A name that uniquely identifies a group of users to the system.

group number (Gid). A unique number assigned to a group of related users. The group number can be substituted in commands that take a group name as an argument.

hardware. The equipment, as opposed to the programming, of a computer system.

header. Constant text that is formatted to be in the top margin of one or more pages.

header label. A special set of records on a diskette describing the contents of the diskette.

hexadecimal. Pertaining to a system of numbers to the base sixteen; hexadecimal digits range from 0 (zero) through 9 (nine) and A (ten) through F (fifteen).

hierarchical tree structure. The organization of files on AIX, similar to tree-structured directories, with each file like a small branch of a larger branch that represents the file's parent directory. A directory can also be contained in another higher level directory, with the parent of all directories represented by the tree's root (**root** or **root directory**).

highlight. To emphasize an area on the display by any of several methods, such as brightening the area or reversing the color of characters within the area.

history. A C-shell mechanism that lists previously executed commands. These commands can be re-executed with the ! command.

history file. A file containing a log of system actions and operator responses.

home directory. The directory a user accesses when he logs in. Synonym for *login directory*.

hog factor. In system accounting, an analysis of how many times each command was run, how much processor time and memory it used, and how intensive that use was.

I/O. See **input/output**.

IF expressions. Expressions within a procedure, used to test for a condition.

indirect block. A block containing pointers to other blocks. Indirect blocks can be single-indirect, double-indirect, or triple-indirect.

informational message. A message providing information to the operator, that does not require a response.

initial program load (IPL). The process of loading the system programs and preparing the system to run jobs. See **initialize**.

initialize. To set counters, switches, addresses, or contents of storage to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

inode. The internal structure for managing files in the system. Inodes contain all of the information pertaining to the node, type, owner, and location of a file. A table of inodes is stored near the beginning of a file system.

i-number. A number specifying a particular inode on a file system.

input. Data to be processed.

input device. Physical devices used to provide data to a computer.

input file. A file opened by a program so that the program can read from that file.

input list. A list of variables to which values are assigned from input data.

input redirection. The specification of an input source other than the standard one.

input-output file. A file opened for input and output use.

input/output (I/O). Pertaining to either input, output, or both between a computer and a device.

interactive processing. A processing method in which each system user action causes response from the program or the system. Contrast with **batch processing**.

interface. A shared boundary between two or more entities. An interface might be a hardware component to link two devices together or it might be a portion of storage or registers accessed by two or more computer programs.

interleave factor. Specification of the ratio between contiguous physical blocks (on a fixed-disk) and logically contiguous blocks (as in a file).

interrupt. (1) To temporarily stop a process. (2) In data communications, to take an action at a receiving station that causes the sending station to end a transmission. (3) A signal sent by an I/O device to the processor when an error has occurred or when assistance is needed to complete I/O. An interrupt usually suspends execution of the currently executing program.

interrupt character. A character (**Ctrl-C** on some systems) typed in to cancel a foreground process.

IPL. See **initial program load**.

job. (1) A unit of work to be done by a system. (2) One or more related procedures or programs grouped into a procedure.

job control. A feature that lets the system accept your commands to stop and start processes (jobs) and move them between background and foreground. The commands **ps** and **jobs** report the status of jobs, (each of which is assigned a Process Identification Number or PID to show its process status), and the **kill** command can be used to stop them.

job number. A number assigned to a background process when it is started. The job number is displayed when the process is started and when the **jobs** command is invoked. It can also be used to kill the process.

job queue. A list, on disk, of jobs waiting to be processed by the system.

justify. To print a document with even right and left margins.

K-byte. See **kilobyte**.

kernel. The memory-resident nucleus of the AIX Operating System containing functions needed immediately and frequently. The kernel supervises the input and output, manages and controls the hardware, and schedules the user processes for execution.

kernel parameters. Variables that specify how the kernel allocates certain system resources.

keyboard. An input device consisting of various keys allowing the user to input data, control cursor and pointer locations, and to control the dialog between the user and the display station.

keylock feature. A security feature in which a lock and key can be used to restrict the use of the display station.

keyword. One of the predefined words of a programming language; a reserved word.

keyword argument. One type of variable assignment that can be made on the command line.

kill. An AIX Operating System command that stops a process.

kill character. The character that is used to delete a line of characters entered after the user's prompt.

kilobyte. 1024 bytes.

label. (1) The name in the disk or diskette volume table of contents that identifies a file. See also **file name**. (2) The field of an instruction that assigns a symbolic name to the location at which the instruction begins, or such a symbolic name.

left margin. The area on a page between the left paper edge and the leftmost character position on the page.

left-adjust. The process of aligning lines of text at the left margin or at a tab setting such that the leftmost character in the line or filed is in the leftmost position. Contrast with **right-adjust**.

library. A collection of functions, calls, subroutines, or other data.

licensed program product (LPP). Software programs that remain the property of the manufacturer, for which customers pay a license fee.

line editor. An editor that modifies the contents of a file one line at a time.

linefeed. An ASCII character that causes an output device to move forward one line.

link. A connection between an inode and one or more file names associated with it. Synonym for *UNIX link* or *hard link*.

literal. A symbol or a quantity in a source program that is itself data, rather than a reference to data.

load. (1) To move data or programs into storage. (2) To place a diskette into a diskette drive, or a magazine into a diskette magazine drive.
(3) To insert paper into a printer.

loader. A program that reads run files into main storage, thus preparing them for execution.

local. Pertaining to a device directly connected to your system without the use of a communications line. Contrast with **remote**.

locale. A set of environment variables that determines the language for keyboard input and display/printer output, the character sets for data in files and on networks, the message catalog, the collating sequence, and the time, date, monetary, and numeric conventions. Each site, user, and program operates within a specified locale, which can be changed by resetting the environment variables.

<LOCAL> alias. The <LOCAL> alias can translate into different strings on different cluster sites for different processes. When <LOCAL> is the first component of the destination name for a symbolic link, it is replaced with its alias string, normally */sitename*.

<LOCAL> file system. The part of the root file system hierarchy comprising system directories and files (such as the **/etc/motd** "message of the day" file) defined uniquely on a particular computer in the cluster.

These files are not replicated. The name of the <LOCAL> file system appears in response to the **site-l** command.

log. To record; for example, to log all messages on the system printer. A list of this type is called a log, such as an error log.

log in. To begin a session at a display station.

log in shell. The program, or command interpreter, started for a user at log in.

log off. To end a session at a display station.

log out. To end a session at a display station.

logical device. A file for conducting input or output with a physical device.

login directory. See home directory.

login shell. The program, or command interpreter, started for a user at log in.

loop. A sequence of instructions performed repeatedly until an ending condition is reached.

LPP. See **licensed program product**.

mailbox. An area designated for storage of mail messages directed to a specific system user.

main program. A primary or control program. See also **program**.

main storage. The part of the processing unit where programs are run.

maintenance system. A special version of the AIX Operating System which is loaded from diskette and used to perform system management tasks.

major device number. A system identification number for each device or type of device.

mapped files. Files on the fixed-disk that are accessed as if they are in memory.

mask. A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

matrix. An array arranged in rows and columns.

maxproc. An AIX kernel parameter establishing the maximum number of processes that can be run simultaneously by a user.

MBCS. See **multibyte character set**.

memory. Storage on electronic chips. Examples of memory are random access memory, read only memory, or registers. See **storage**.

menu. A displayed list of items from which an operator can make a selection.

message. (1) A response from the system to inform the operator of a condition which may affect further processing of a current program. (2) A communication sent from a person or program to another person or program.

message catalog. A file of messages in which each message has a number, and related messages are grouped into sets. AIX supplies different system message catalogs for different locales.

minidisk. A logical division of a fixed disk.

minor device number. A number used to specify various types of information about a particular device, for example, to distinguish among several printers of the same type.

mode word. An inode field that describes the type and state of the inode.

modem. See **modulator-demodulator**.

modulation. Changing the frequency or size of one signal by using the frequency or size of another signal.

modulator-demodulator (modem). A device that converts data from the computer to a signal that can be transmitted on a communications line, and converts the signal received to data for the computer.

module. (1) A discrete programming unit that usually performs a specific task or set of tasks. Modules are subroutines and calling programs that are assembled separately, then linked to make a complete program. (2) See **load module**.

mount. To make accessible to a file system or file tree. AIX allows local file and directory mounts.

multibyte character set (MBCS). MBCS is a system of encoding characters in computer data. It allows the set of represented data to be large enough to accommodate the Japanese language character sets.

multiprogramming. The processing of two or more programs at the same time, on the same logical system.

multivolume file. A diskette file occupying more than one diskette.

multi-user environment. A computer system that provides terminals and keyboards for more than one user at the same time.

nest. To incorporate a structure or structures of some kind into a structure of the same kind. For example, to nest one loop (the nested loop) within another loop (the nesting loop); to nest one subroutine (the nested subroutine) within another subroutine (the nesting subroutine).

network. A collection of computers that can communicate with each other. A network can consist of several interconnected computers or one computer with a number of remote terminals connected to it. Any of a variety of communication media can be used, such as RS232, Ethernet, PC Net, etc.

new-line character. A control character that causes the print or display position to move to the first position on the next line.

node. An individual system connected to a network.

null. Having no value, containing nothing.

null character (NUL). The character hex 00, used to represent the absence of a printed or displayed character.

object code. Machine-executable instruction, usually generated by a compiler from source code written in a higher level language. consists of directly executable machine code. For programs that must be linked, object code consists of relocatable machine code.

octal. A base eight numbering system.

open. To make a file available to a program for processing.

operating system. Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

operation. A specific action (such as move, add, multiply, load) that the computer performs when requested.

operator. A symbol representing an operation to be done.

output. The result of processing data.

output devices. Physical devices used by a computer to present data to a user.

output file. A file that is opened by a program so that the program can write to that file.

output redirection. The specification of an output destination other than the standard one.

override. (1) A parameter or value that replaces a previous parameter or value. (2) To replace a parameter or value.

overwrite. To write output into a storage or file space that is already occupied by data.

owner. The user who has the highest level of access authority to a data object or action, as defined by the object or action.

pad. To fill unused positions in a field with dummy data, usually zeros or blanks.

page. A block of instructions, data, or both.

page space minidisk. The area on a fixed disk that temporarily stores instructions or data currently being run. See also **minidisk**.

pagination. The process of adjusting text to fit within margins and/or page boundaries.

paging. The action of transferring instructions, data, or both between real storage and external page storage.

parallel processing. The condition in which multiple tasks are being performed simultaneously within the same activity.

parameter. Information that the user supplies to a panel, command, or function.

parent. Pertaining to a secured resource, either a file or library, whose user list is shared with one or more other files or libraries. Contrast with **child**.

parent directory. The directory one level above the current directory.

partition. See **minidisk**.

password. A string of characters that, when entered along with a user identification, allows an operator to sign on to the system.

password security. A program product option that helps prevent the unauthorized use of a display station, by checking the password entered by each operator at sign-on.

path name. The sequential list of directory names(s) that identify the location of a particular directory, and directory names(s), and file name that identify the location of a particular file in the file hierarchy. The path name is displayed in response to the **pwd** (print working directory) command. Each file has a full path name, beginning with / (root directory) and ending with the file's name. The file's relative path name does not begin with /.

pattern-matching character. Special characters such as * or ? that can be used in search patterns or in a file specification to match one or more characters. For example, placing a ? in a file specification means any character can be in that position. Pattern-matching characters are also called wildcards.

permission code. A three-digit octal code, or a nine-letter alphabetic code, indicating the access permissions. The access permissions are read, write, and execute.

permission field. One of the three-character fields within the permissions column of a directory listing indicating the read, write, and run permissions for the file or directory owner, group, and all others.

phase. One of several stages file system checking and repair performed by the **fsck** command.

physical device. See **device**.

physical file. An indexed file containing data for which one or more alternative indexes have been created.

physical record. (1) A group of records recorded or processed as a unit. Same as **block**. (2) A unit of data moved into or out of the computer.

PID. See **process ID**.

pipe. To direct the data so that the output from one process becomes the input to another process.

pipeline. A direct, one-way connection between two or more processes.

pitch. A unit of width of typewriter type, based on the number of times a letter can be set in a linear inch. For example, 10-pitch type has 10 characters per inch.

platen. The support mechanism for paper on a printer, commonly cylindrical, against which printing mechanisms strike to produce an impression.

pointer. A logical connection between physical blocks. A link to something else, an address.

port. (1) To make the programming changes necessary to allow a program that runs on one type of computer to run on another type of computer. (2) An access point for data input to or data output from a computer system. See **connector**.

position. The location of a character in a series, as in a record, a displayed message, or a computer printout.

positional parameter. A shell facility for assigning values from the command line to variables in a program.

print queue. A file containing a list of the names of files waiting to be printed.

printout. Information from the computer produced by a printer.

priority. The relative ranking of items. For example, a job with high priority in the job queue will be run before one with medium or low priority.

priority number. A number that establishes the relative priority of printer requests.

privileged user. The account with superuser authority.

problem determination. The process of identifying why the system is not working. Often this process identifies programs, equipment, data communications facilities, or user errors as the source of the problem.

problem determination procedure. A prescribed sequence of steps aimed at recovery from, or circumvention of, problem conditions.

procedure. See **shell procedure**.

process. A program now running. A *foreground* process executes as soon as you type in the command line and completes before returning the system prompt to accept your next command. You can start one or more *background processes* to run independently while you type in a separate command for another process to run in the foreground.

process accounting. An analysis of the use each process makes of the processing unit, memory, and I/O resources.

process ID (PID). A unique number assigned to a process that is running.

profile. (1) A file containing customized settings for a system or user. (2) Data describing the significant features of a user, program, or device.

program. A set of instructions for the computer to interpret and execute.

prompt. A displayed request for information or operator action.

propagation time. The time necessary for a signal to travel from one point on a communications line to another.

qdaemon. The daemon process that maintains a list of outstanding jobs and sends them to the specified device at the appropriate time.

queue. A line or list formed by items waiting to be processed.

queued message. A message from the system that is added to a list of messages stored in a file for viewing by the user at a later time. This is in contrast to a message that is sent directly to the screen for the user to see immediately.

quit. A key, command, or action that tells the system to return to a previous state or stop a process.

quote. To mask the special meaning of certain characters; to cause them to be taken literally.

random access. An access mode in which records can be read from, written to, or removed from a file in any order.

readonly. Pertaining to file system mounting, a condition that allows data to be read, but not modified.

recovery procedure. (1) An action performed by the operator when an error message appears on the display screen. Usually, this action permits the program to continue or permits the operator to run the next job. (2) The method of returning the system to the point where a major system error occurred and running the recent critical jobs again.

redirect. To divert data from a process to a file or device to which it would not normally go.

reference count. In an inode, a record of the total number of directory entries that refer to the inode.

relational expression. A logical statement describing the relationship (such as greater than or equal) of two arithmetic expressions or data items.

relational operator. The reserved words or symbols used to express a relational condition or a relational expression.

relative address. An address specified relative to the address of a symbol. When a program is relocated, the addresses themselves will change, but the specification of relative addresses remains the same.

relative addressing. A means of addressing instructions and data areas by designating their locations relative to some symbol.

relative path name. The name of a directory or file expressed as a sequence of directories followed by a file name, beginning from the current directory.

remote. Pertaining to a system or device that is connected to your system through a communications line. Contrast with **local**.

remote cluster site. A site on the cluster that the user is not logged in to. The term **remote** normally refers to a TCF cluster site.

replicated root file system. The replicated root file system is a file system with key common files and directories for basic system operation. Almost all system binaries, programs and libraries are in the replicated root file system. Other user and system file systems (like the local file system) are mounted on top of directories in the replicated root file system.

reserved character. A character or symbol that has a special (non-literal) meaning unless quoted.

reserved word. A word that is defined in a programming language for a special purpose, and that must not appear as a user-declared identifier.

reset. To return a device or circuit to a clear state.

restore. To return to an original value or image. For example, to restore a library from diskette.

right adjust. The process of aligning lines of text at the right margin or tab setting such that the rightmost character in the line or file is in the rightmost position.

right justify. See right align.

right margin. The area on a page between the last text character and the right upper edge.

right-adjust. To place or move an entry in a field so that the rightmost character of the field is in the rightmost position. Contrast with **left-adjust**.

root. Another name sometimes used for superuser.

root directory. The top level of a tree-structured directory system.

root file system. The basic AIX Operating System file system, which contains operating system files and onto which other file systems can be mounted. The root file system is the file system that contains the files that are run to start the system running.

routine. A set of statements in a program causing the system to perform an operation or a series of related operations.

run. To cause a program, utility, or other machine function to be performed.

run-time environment. A collection of subroutines and shell variables that provide commonly used functions and information for system components.

scratch file. A file, usually used as a work file, that exists until the program that uses it ends.

screen. See **display screen**.

scroll. To move information vertically or horizontally to bring into view information that is outside the display screen boundaries.

sector. (1) An area on a disk track or a diskette track reserved to record information. (2) The smallest amount of information that can be written to or read from a disk or diskette during a single read or write operation.

security. The protection of data, system operations, and devices from accidental or intentional ruin, damage, or exposure.

segment. A contiguous area of virtual storage allocated to a job or system task. A program segment can be run by itself, even if the whole program is not in main storage.

separator. A character used to separate parts of a command or file.

sequential access. An access method in which records are read from, written to, or removed from a file based on the logical order of the records in the file.

server. (1) On a network, the computer that contains programs, data, or provides the facilities to be accessed by other computers on the network. (2) A program that handles protocol, queueing, routing, and other tasks necessary for data transfer between devices in a computer system. (3) An application program that usually runs in the background (daemon) and is controlled by the System Program Controller.

session records. In the accounting system, a record of time connected and line usage for connected display stations, produced from log in and log out records.

set flags. Flags that can be put into effect with the shell set command.

shared printer. A printer that is used by more than one work station.

shell. A program that accepts and interprets commands for the operating system, such as sh, csh, and the DOS shell program. Also called a **shell**

program.

shell options. The shell provides two different types of options. **Set options** are put into effect with the set command and alter the way the shell runs. **Command line options** are entered on the command line (but not with the **set** command) and alter the way the shell starts.

shell procedure. A series of commands contained in a file that carry out a particular function when the file is run or when the file is specified as an argument to the sh command. Also called a **shell script**.

shell program. See **shell**.

shell prompt. The character string on the command line indicating that the system can accept a command (typically the **$** character).

shell script. See **shell procedure**.

shell variables. Facilities of the shell program for assigning variable values to constant names.

shutdown. The process of ending the operation of a system or a subsystem by following a defined procedure.

size field. In an inode, a field that indicates the size, in bytes, of the file associated with the inode.

software. Programs.

sort. To rearrange some or all of a group of items based upon the contents or characteristics of those items.

source diskette. The diskette containing data to be copied, compared, restored, or backed up.

source program. A set of instructions written in a programming language, that must be translated to machine language compiled before the program can be run.

special character. A character that has special meaning to a shell (for example, ? and *).

special file. Used in the AIX system to provide an interface to input/output devices. There is at least one special file for each device connected to the computer. Contrast with **directory** and **file**. See also **block special file** and **character special file**.

spool files. Files used in the transmission of data among devices.

standalone shell. A limited version of the shell program used for system maintenance.

stand-alone system. See **stand-alone work station**.

stand-alone work station. A work station that can be used to perform tasks independent of (without being connected to) other resources such as servers or host systems.

standard error (STDERR). The place where many programs place error messages.

standard input (STDIN). The primary source of data going into a command. Standard input comes from the keyboard unless redirection or piping is used, in which case standard input can be from a file or the output from another command.

standard output (STDOUT). The primary destination of data coming from a command. Standard output goes to the display unless redirection or piping is used, in which case standard output can be to a file or another command.

stanza. A group of lines in a file that together have a common function. Stanzas are usually separated by blank lines, and each stanza has a name.

statement. An instruction in a program or procedure.

status. (1) The current condition or state of a program or device. For example, the status of a printer. (2) The condition of the hardware or software, usually represented in a status code.

STDERR. See **standard error**.

STDIN. See **standard input**.

STDOUT. See **standard output**.

storage. (1) The location of saved information. (2) In contrast to memory, the saving of information on physical devices such as disk or tape. See **memory**.

storage device. A device for storing and/or retrieving data.

string. A series of characters to be taken literally by the system. A string may be specified for a context search, for instance, or for global substitutions.

su. See **superuser**.

subdirectory. A directory contained within another directory in the file system hierarchy.

subprogram. A program invoked by another program. Contrast with **main program**.

subroutine. (1) A sequenced set of statements that may be used in one or more computer programs and at one or more points in a computer program. (2) A routine that can be part of another routine. See also **routine**.

subscript. An integer or variable whose value refers to a particular element in a table or an array.

subshell. An instance of the shell program started from an existing shell program.

substitution. A procedure used by a text editor like **ed** or **vi** to replace one specified string of characters with another. If a global substitution is made, all occurrences of the specified text pattern are replaced with the new one.

substring. A part of a character string.

subsystem. A secondary or subordinate system, usually capable of operating independently of, or synchronously with, a controlling system.

superblock. The most critical part of the file system containing information about every allocation or deallocation of a block in the file system.

superuser (su). The user who can operate without the restrictions designed to prevent data loss or damage to the system (user ID 0). Also known as **superuser** or **su**.

superuser authority. The unrestricted ability to access and modify any part of the operating system that is associated with the user who manages the system. The authority obtained when one logs in as **root**.

symbolic link. A mechanism that lets you assign a second name to a file or directory. There are no restrictions pertaining to source or destination location. A file may be deleted if the only pointers to the file are symbolic links.

synchronous. Occurring in a regular or predictable sequence.

synchronous transmission. In data communication, a method of transmission in which the sending and receiving of characters is controlled by timing signals. Contrast with **asynchronous transmission**.

system. The computer and its associated devices and programs.

system call. A request by an active process for a service by the system kernel.

system customization. A process of specifying the devices, programs, and users for a particular data processing system.

system date. The date assigned by the system user during setup and maintained by the system.

system dump. A copy of memory made whenever an error stops the system. Contrast with **task dump**.

system management. The tasks involved in maintaining the system in good working order and modifying the system to meet changing requirements.

system parameters. See **kernel parameters**.

system primary site. The machine (cluster site) designated to hold the primary copy of the replicated root file system. When files are changed in the replicated root file system, the primary site for the cluster must be available.

system profile. A file containing the default values used in system operations.

system unit. The part of the system that contains the processing unit, the disk drives, and the diskette drives.

target diskette. The diskette to be used to receive data from a source diskette.

task. A basic unit of work to be performed. Examples are a user task, a server task, and a processor task.

task dump. A copy of memory associated program that failed (and its data). Contrast with **system dump**.

terminal. An input/output device containing a keyboard and either a display device or a printer. Terminals usually are connected to a computer and allow a person to interact with the computer.

text. A type of data consisting of a set of linguistic characters (for example, alphabet, numbers, and symbols) and formatting controls.

text application. A program defined for the purpose of processing text data (for example, memos, reports, and letters).

text editing program. See **editor** and **text application**.

trace. To record data that provides a history of events occurring in the system.

trace table. A storage area into which a record of the performance of computer program instructions is stored.

track. A circular path on the surface of a fixed disk or diskette on which information is magnetically recorded and from which recorded information is read.

trap. An unprogrammed, hardware-initiated jump to a specific address. Occurs as a result of an error or certain other conditions.

tree-structured directories. A method for connecting directories such that each directory is listed in another directory except for the root directory, which is at the top of the tree.

truncate. To shorten a field or statement to a specified length.

TTY. Designates a terminal. On a system with more than one terminal, the TTY field of the process status displayed by the **ps** command indicates which terminal started the process.

typematic key. A key that repeats its function multiple times when held down.

typestyle. Characters of a given size, style and design.

Uid. See **user number**.

UNIX link. A mechanism that lets you use the **ln** command to assign more than one name to a file. Both the new name and the file being linked to must be in the same file system. A file is deleted when all the UNIX links (including the first link--the original name) have been removed. Synonym for **hard link**.

update. An improvement for some part of the system.

user. The name associated with an account.

user account. See **account**.

user ID. See **user number**.

user name. A name that uniquely identifies a user to the system.

user number (Uid). A unique number identifying an operator to the system. This string of characters limits the functions and information the operator is allowed to use. The Uid can often be substituted in commands that take a user's name as an argument.

user profile. A file containing a description of user characteristics and defaults (for example, printer assignment, formats, group ID) to be conveyed to the system while the user is signed on.

utility. A service; in programming, a program that performs a common service function.

valid. (1) Allowed. (2) True, in conforming to an appropriate standard or authority.

value. (1) In Usability Services, information selected or typed into a pop-up. (2) A set of characters or a quantity associated with a parameter or name. (3) In programming, the contents of a storage location.

variable. A name used to represent a data item whose value can change while the program is running. Contrast with **constant**.

verify. To confirm the correctness of something.

version. Information in addition to an object's name that identifies different modification levels of the same logical object.

virtual device. A device that appears to the user as a separate entity but is actually a shared portion of a real device. For example, several virtual terminals may exist simultaneously, but only one is active at any given time.

virtual storage. Addressable space that appears to be real storage. From virtual storage, instructions and data are mapped into real storage locations.

virtual terminal. Any of several logical equivalents of a display station available at a single physical display station.

Volume ID (Vol ID). A series of characters recorded on the diskette used to identify the diskette to the user and to the system.

wildcard. See **pattern-matching characters**.

word. A contiguous series of 32 bits (4 bytes) in storage, addressable as a unit. The address of the first byte of a word is evenly divisible by four.

work file. A file used for temporary storage of data being processed.

work station. A device at which an individual may transmit information to, or receive information from, a computer for the purpose of performing a task, for example, a display station or printer. See **programmable work station** and **dependent work station**.

working directory. See **current directory**.

wrap around. Movement of the point of reference in a file from the end of one line to the beginning of the next, or from one end of a file to the other.