Bull

AIX Commands Reference Vol.1
ac to cxref

AIX

ORDER REFERENCE
86 A2 38JX 02

Bull

AIX Commands Reference Vol.1
ac to cxref

AIX

Software

April 2000

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 38JX 02

Printed in France

Suggestions and criticisms concerning the form, content, and presentation of this book are invited. A form is provided at the end of this book for this purpose.

To order additional copies of this book or other Bull Technical Publications, you are invited to use the Ordering Form also provided at the end of this book.

Trademarks and Acknowledgements

We acknowledge the right of proprietors of trademarks mentioned in this book.

AIX® is a registered trademark of International Business Machines Corporation, and is being used under licence.

UNIX is a registered trademark in the United States of America and other countries licensed exclusively through the Open Group.

Year 2000

The product documented in this manual is Year 2000 Ready.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Commands Reference, Volume 1

First Edition (October 1997)

This edition of the *AIX Version 4.3 Commands Reference, Volume 1* applies to the AIX Version 4.3, 3270 Host Connection Program 2.1 and 1.3.3 for AIX, and Distributed SMIT 2.2 for AIX licensed programs, and to all subsequent releases of these products until otherwise indicated in new releases or technical newsletters.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to Publications Department, Internal Zip 9561, 11400 Burnet Road, Austin, Texas 78758−3493. To send comments electronically, use this commercial internet address: `aix6kpub@austin.ibm.com`. Any information that you supply may be used without incurring any obligation to you.

Trademarks and Acknowledgements

The following trademarks and acknowledgements apply to this book:

ADM is a trademark of Lear Siegler, Inc.

AIX is a registered trademark of International Business Machines Corporation.

Connect is a trademark of INTERACTIVE Systems Corporation.

DEC is a trademark of Digital Equipment Corporation.

DEC VT100, VT220, VT320, and VT330 are trademarks of Digital Equipment Corporation.

GL is a trademark of Silicon Graphics, Inc.

HP is a trademark of Hewlett−Packard Company.

IBM is a registered trademark of International Business Machines Corporation.

INed is a trademark of INTERACTIVE Systems Corporation.

InfoExplorer is a trademark of International Business Machines Corporation.

Intel is a trademark of Intel Corporation.

Interleaf is a trademark of Interleaf, Inc.

LaserJet Series II is a trademark of Hewlett−Packard Company.

Micro Channel is a registered trademark of International Business Machines Corporation.

NetView is a trademark of International Business Machines Corporation.

Network Computing System is a trademark of Apollo Computer, Inc.

OSF and OSF/Motif are trademarks of Open Software Foundation, Inc.

Personal Computer AT and AT is a registered trademark of International Business Machines Corporation.

Personal System/2 is a registered trademark of International Business Machines Corporation.

PS/2 is a registered trademark of International Business Machines Corporation.

POSIX is a trademark of the Institute of Electrical and Electronic Engineers (IEEE).

PostScript is a trademark of Adobe Systems Incorporated.

Proprinter is a registered trademark of International Business Machines Corporation.

Quickwriter is a registered trademark of International Business Machines Corporation.

Quiet is a trademark of International Business Machines Corporation.

RS/6000 is a trademark of International Business Machines Corporation.

RT is a registered trademark of International Business Machines Corporation.

Sun is a trademark of Sun Microsystems, Inc.

Tektronix is a trademark of Tektronix, Inc.

Televideo is a trademark of Televideo, Inc.

The Source is a service mark of Source Telecomputing Corp., a subsidiary of The Reader's Digest Assn., Inc.

UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

WY−50 is a trademark of the WYSE Corporation.

WYSE is a trademark of WYSE Corporation.

About This Book

This book is Volume 1 of the six−volume *AIX Version 4.3 Commands Reference*, SBOF−1877, which contains reference information on Advanced Interactive Executive (AIX) Operating System commands. It describes the tasks each command performs, how commands can be modified, how they handle input and output, who can run them and provides a master index for all six volumes.

For a quick reference list of commands arranged in functional groups, see Volume 6.

Who Should Use This Book

This book is intended for users of AIX commands.

How to Use This Book

A command is a request to perform an operation or run a program. You use commands to tell the AIX Operating System what task you want it to perform. When commands are entered, they are deciphered by a command interpreter (also known as a shell) and that task is processed.

Some commands can be entered simply by typing one word. It is also possible to combine commands so that the output from one command becomes the input for another command. This is known as pipelining.

Flags further define the actions of commands. A flag is a modifier used with the command name on the command line, usually preceded by a dash.

Commands can also be grouped together and stored in a file. These are known as shell procedures or shell scripts. Instead of executing the commands individually, you execute the file that contains the commands.

Some commands can be constructed using Web−based System Manager applications or the System Management Interface Tool (SMIT).

Highlighting

The following highlighting conventions are used in this book:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or information you should actually type.

Format

Each command may include any of the following sections:

Purpose	A description of the major function of each command.
Syntax	A syntax diagram showing command line options.
Description	A discussion of the command describing in detail its function and use.

Flags	A list of command line flags and associated variables with an explanation of how the flags modify the action of the command.
Parameters	A list of command line parameters and their descriptions.
Subcommands	A list of subcommands (for interactive commands) that explains their use.
Exit Status	A description of the exit values the command returns.
Security	Specifies any permissions needed to run the command.
Examples	Specific examples of how you can use the command.
Files	A list of files used by the command.
Related Information	A list of related commands in this book and related discussions in other books.

Implementation Specifics

To list the installable software package (fileset) of an individual command use the **lslpp** command with the **−w** flag. For example, to list the fileset that owns the **installp** command, enter:

```
lslpp −w /usr/sbin/installp
```

Output similar to the following displays:

```
File                            Fileset                 Type
-----------------------------------------------------------------------
/usr/sbin/installp                bos.rte.install           File
```

To list the fileset that owns all file names that contain `installp`, enter:

```
lslpp −w "*installp*"
```

Output similar to the following displays:

```
File                            Fileset                 Type
-----------------------------------------------------------------------
/usr/sbin/installp                bos.rte.install           File
/usr/clvm/sbin/linstallpv         prpq.clvm                 File
/usr/lpp/bos.sysmgt/nim/methods/c_installp
                                  bos.sysmgt.nim.client     File
```

Syntax Diagrams

AIX command syntax is represented by syntax diagrams and usage statements.

Syntax diagrams are designed to provide information about how to enter the command on the command line. A syntax diagram can tell you:

- Which flags can be entered on the command line
- Which flags must take a parameter
- Which flags have optional parameters
- Default values of flags and parameters, if any
- Which flags can and cannot be entered together
- Which flags and parameters are optional
- When you can repeat flag and parameter sequences.

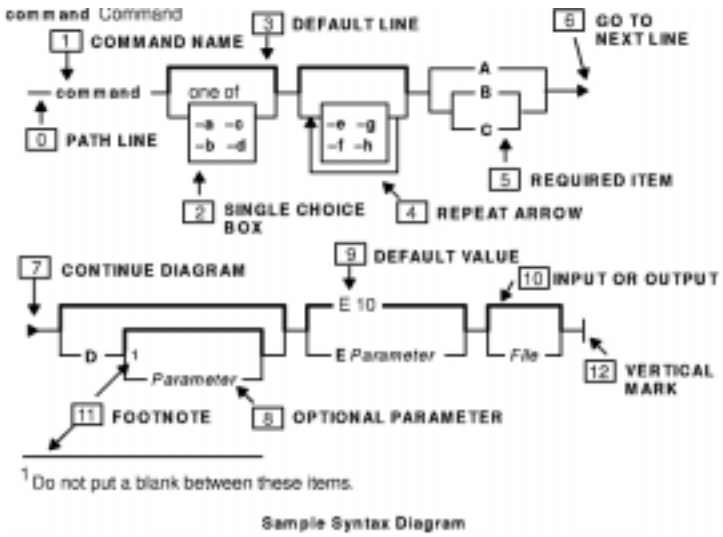
AIX commands use the following conventions in their syntax diagrams:

- Diagram items that must be entered literally on the command line are in **bold**. These items include

the command name, flags, and literal characters.

- Diagram items representing variables that must be replaced by a name are in *italics*. These items include parameters that follow flags and parameters that the command reads, such as *Files* and *Directories*.
- Default values that do not have to be entered are in the normal font on a **bold** path.

The Sample Syntax Diagram illustrates the conventions used in syntax diagrams. Each part of the diagram is labeled. An explanation of the labels follows the diagram.



Sample Syntax Diagram

You interpret the example diagram as follows.

0 PATH LINE
The path line begins the syntax diagram.

1 COMMAND NAME
This item in the diagram is the name of the command you want to invoke. It is in bold, which indicates that it must be entered exactly as it appears in the diagram.

In the example diagram, the path branches into two paths after the command name. You can follow either the lower path (discussed in item 2) or the upper path (discussed in item 3).

2 SINGLE CHOICE BOX
If you follow the lower path, you encounter a box with the words *one of* over it. You can choose only one item from this box.

3 DEFAULT LINE
If you follow the upper path, you bypass the single choice box, and enter nothing. The bold line around the box is a default line, which means that you do not have to enter anything from that part of the diagram. Exceptions are usually explained under "Description." One important exception, the blank default line around input and output files, is explained in item 10.

4 REPEAT ARROW
When you follow a path that takes you to a box with an arrow around it, you must choose at least one item from the box. Then you can either follow the arrow back around and continue to choose items from the box, or you can continue along the path. When following an arrow that goes around a box (rather than an arrow that includes several branches in the diagram), do not choose the same item more than once.

5 REQUIRED ITEM
Following the branch with the repeat arrow is a branch with three choices and no default line around them. This means that you must choose one of A, B, or C.

6 GO TO NEXT LINE
If a diagram is too long to fit on one line, this character tells you to go to the next line of the diagram to continue entering your command. Remember, the diagram does not end until you reach the vertical mark.

7 CONTINUE DIAGRAM	This character shows you where to continue with the diagram after it breaks on the previous line.
8 OPTIONAL PARAMETER	If a flag can (but does not have to) take a parameter, the path branches after the flag. If you cannot enter a space between the flag and parameter, you are told in a footnote.
9 DEFAULT VALUE	Often, a command has default values or actions that it will follow if you do not enter a specific item. These default values are indicated in normal font in the default line if they are equivalent to something you could enter on the command line (for example, a flag with a value). If the default is not something you can enter on the command line, it is not indicated in the diagram. **Note:** Default values are included in the diagram for your information. It is not necessary to enter them on the command line.
10 INPUT OR OUTPUT	A command that can read either input files or standard input has an empty 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 an empty 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.
11 FOOTNOTE	If a command has special requirements or restrictions, a footnote calls attention to these differences.
12 VERTICAL MARK	This ends the syntax diagram.

Running Commands in the Background

If you are going to run a command that takes a long time to process, you can specify that the command run in the background. Background processing is a useful way to run programs that process slowly. To run a command in the background, you use the & (ampersand) operator at the end of the command:

```
Command&
```

Once the process is running in the background, you can continue to work and enter other commands on your system.

At times, you might want to run a command at a specified time or on a specific date. Using the **cron** daemon, you can schedule commands to run automatically. Or, using the **at** and **batch** commands, you can run commands at a later time or when the system load level permits.

Entering Commands

When you work with AIX, you typically enter commands following the shell prompt on the command line. The shell prompt can vary. In the following examples, $ is the prompt.

To display a list of the contents of your current directory, you would type **ls** and press the Enter key:

```
$ ls
```

When you enter a command and it is running, the operating system does not display the shell prompt. When the command completes its action, the system displays the prompt again. This indicates that you can enter another command.

The general format for entering AIX commands is:

```
Command Flag(s) Parameter
```

The flag alters the way a command works. Many commands have several flags. For example, if you type the **−l** (long) flag following the **ls** command, the system provides additional information about the contents of the current directory. The following example shows how to use the **−l** flag with the **ls** command:

```
$ ls −l
```

A parameter consists of a string of characters that follows a command or a flag. It specifies data, such as the name of a file or directory, or values. In the following example, the directory named **/usr/bin** is a parameter:

```
$ ls −l /usr/bin
```

When entering commands in AIX, it is important to remember the following:

- Commands are usually entered in lowercase.
- Flags are usually prefixed with a − (minus sign).
- More than one command can be typed on the command line if the commands are separated by a ; (semicolon).
- Long sequences of commands can be continued on the next line by using the \ (backslash). The backslash is placed at the end of the first line. The following example shows the placement of the backslash:

  ```
  $ cat /usr/ust/mydir/mydata > \
  /usr/usts/yourdir/yourdata
  ```

When certain commands are entered, the shell prompt changes. Because some commands are actually programs (such as the **telnet** command), the prompt changes when you are operating within the command. Any command that you issue within a program is known as a subcommand. When you exit the program, the prompt returns to your shell prompt.

AIX can operate with different shells (for example, Bourne, C, or Korn) and the commands that you enter are interpreted by the shell. Therefore, you must know what shell you are using so that you can enter the commands in the correct format.

Stopping Commands

If you enter a command and then decide to stop that command from running, you can halt the command from processing any further. To stop a command from processing, press the Interrupt key sequence (usually Ctrl−C or Alt−Pause). When the process is stopped, your shell prompt returns and you can then enter another command.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

AIX 32−Bit Support for the X/Open UNIX95 Specification

Beginning with AIX Version 4.2, the operating system is designed to support the X/Open UNIX95 Specification for portability of UNIX−based operating systems. Many new interfaces, and some current ones, have been added or enhanced to meet this specification. Beginning with Version 4.2, AIX is even more open and portable for applications.

At the same time, compatibility with previous AIX releases is preserved. This is accomplished by the creation of a new environment variable, which can be used to set the system environment on a per−system, per−user, or per−process basis.

To determine the proper way to develop a UNIX95−portable application, you may need to refer to the X/Open UNIX95 Specification, which can be obtained on a CD−ROM by ordering the printed copy of *AIX Version 4.3 Commands Reference*, order number SBOF−1877, or by ordering *Go Solo: How to Implement and Go Solo with the Single Unix Specification*, order number SR28−5705, a book which includes the X/Open UNIX95 Specification on a CD−ROM.

AIX 32−Bit and 64−Bit Support for the UNIX98 Specification

Beginning with AIX Version 4.3, the operating system is designed to support the X/Open UNIX98 Specification for portability of UNIX−based operating systems. Many new interfaces, and some current ones, have been added or enhanced to meet this specification. Making AIX Version 4.3 even more open and portable for applications.

At the same time, compatibility with previous AIX releases is preserved. This is accomplished by the creation of a new environment variable, which can be used to set the system environment on a per−system, per−user, or per−process basis.

To determine the proper way to develop a UNIX98−portable application, you may need to refer to the X/Open UNIX98 Specification, which can be obtained on a CD−ROM by ordering the printed copy of *AIX Version 4.3 Commands Reference*, order number SBOF−1877, or by ordering *Go Solo: How to Implement and Go Solo with the Single Unix Specification*, order number SR28−5705, a book which includes the X/Open UNIX98 Specification on a CD−ROM.

Related Information

The following books contain information about or related to commands:

- *AIX and Related Products Documentation Overview*, Order Number SC23−2456.
- *AIX Version 4.3 Files Reference*, Order Number SC23−4168.
- *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*, Order Number SC23−4128.
- *AIX Version 4.3 Problem Solving Guide and Reference*, Order Number SC23−4123.
- *AIX Version 4.3 System Management Guide: Communications and Networks*, Order Number SC23−4127.
- *AIX Version 4.3 System Management Guide: Operating System and Devices*, Order Number SC23−4126.
- *AIX Version 4.3 System User's Guide: Operating System and Devices*, Order Number SC23−4121.
- *AIX Version 4.3 System User's Guide: Communications and Networks*, Order Number SC23−4122.
- *AIX Versions 3.2 and 4 Performance Tuning Guide*, Order Number SC23−2365.
- *AIX Version 4.3 Guide to Printers and Printing*, Order Number SC23−4130.
- *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*, Order Number SC23−4125.
- *5080 Graphics System Installation, Operation, and Problem Determination*, Order Number GA23−2063.

- *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 1* Order Number SC23−4159
- *AIX Version 4.3 Technical Reference: Base Operating System and Extensions Volume 2*, Order Number SC23−4160.
- *AIX Version 4.3 Technical Reference: Communications Volume 1*, Order Number SC23−4161.
- *AIX Version 4.3 Technical Reference: Communications Volume 2*, Order Number SC23−4162
- *AIX Version 4.3 Technical Reference: Kernel and Subsystems Volume 1*, Order Number SC23−4163.
- *AIX Version 4.3 Technical Reference: Kernel and Subsystems Volume 2*, Order Number SC23−4164.
- *AIX Version 4 Keyboard Technical Reference*, Order Number SC23−2631.
- *Distributed SMIT 2.2 for AIX: Guide and Reference*, Order Number SC23−2667.
- *3270 Host Connection Program 2.1 and 1.3.3 for AIX: Guide and Reference*, Order Number SC23−2563.

The following books also may be helpful:

- Lamb, Linda. *Learning the vi Editor*. Sebastopol, CA: O'Reilly & Associates, 1990. Order Number SR28−4966.
- Dougherty, Dale. *sed & awk*. Sebastopol, CA: O'Reilly & Associates, 1990. Order Number SR28−4968.
- Hunt, Craig. *TCP/IP Network Administration*. Sebastopol, CA: O'Reilly & Associates, 1992. Order Number SR23−7422.

Ordering Publications

You can order publications from your sales representative or from your point of sale.

To order additional copies of this book, use order number SC23−4115.

To order additional copies of all six volumes of *AIX Version 4.3 Commands Reference*, use Order Number SBOF−1877.

Use *AIX and Related Products Documentation Overview* for information on related publications and how to obtain them.

Alphabetical Listing of Commands

ac Command

Purpose

Prints connect−time records.

Syntax

/usr/sbin/acct/ac [**−d**] [**−p**] [**−w** *File*] [*User* ...]

Description

The **ac** command prints the total connect time for all users or the connect time for specified users. Records are based on who logged in during the life of the current **wtmp** data file.

Connect−time records are created by the **init** and the **login** programs and are collected in the **/var/adm/wtmp** file, if that file exists. The root user or a member of the **adm** group should create the **/var/adm/wtmp** file with an initial record length of 0 (zero). Records should be processed periodically to keep the file from becoming too full. If the file has not been created, the following error message is returned:

```
No /var/adm/wtmp
```

If the file becomes too full, additional **wtmp** files are created. These files can be printed, if specified with the **−w** flag.

Flags

−d Creates a printout for each day, from midnight to midnight.

−p Prints connect−time totals by individual login. Without this flag, a total for the time period is printed.

−w *File* Specifies a **wtmp** file other than the **/var/adm/wtmp** file.

Security

Access Control: This command should grant execute (x) access to all users.

Examples

1. To obtain a printout of the connect time for all users who logged in during the life of the current **wtmp** data file, enter:

   ```
   /usr/sbin/acct/ac
   ```

2. To obtain a printout of the total connect time for users `smith` and `jones`, as recorded in the current **wtmp** data file, enter:

```
/usr/sbin/acct/ac smith jones
```

3. To obtain a printout of the connect−time subtotals for users `smith` and `jones`, as recorded in the current **wtmp** data file, enter:

   ```
   /usr/sbin/acct/ac −p smith jones
   ```

Files

/usr/sbin/acct/ac Contains the **ac** command.

/var/adm/wtmp Contains the active data file for the collection of connect−time records.

Related Information

The **init** and **login** commands.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

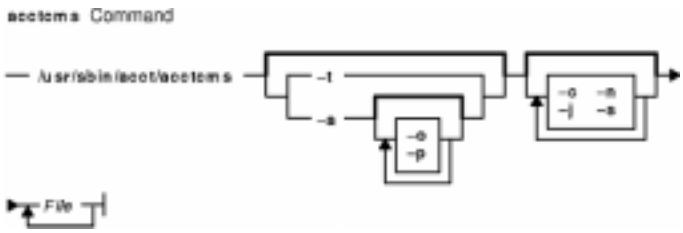
Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains the steps you must take to establish an accounting system.

acctcms Command

Purpose

Produces command−usage summaries from accounting records.

Syntax



/usr/sbin/acct/acctcms [**−t** | **−a** [**−o**] [**−p**]] [**−c**] [**−j**] [**−n**] [**−s**] [*File ...*]

Description

The **acctcms** command reads each file specified by the *File* parameter, adds and sorts all records for identically named processes, and writes the records to standard output. By default, the output file is in binary format. Input files are usually in the **acct** file format.

When you use the **−o** and **−p** flags together, the **acctcms** command produces a report that combines prime and nonprime time. Prime and nonprime times are defined by entries in the **/etc/acct/holidays** file. Prime times are assumed to be the period when the system is most active, such as weekdays. Saturdays and Sundays are always nonprime time for the accounting systems, as are any holidays that you specify in the **/etc/acct/holidays** file. All the output summaries are of total usage, except for number of times run, CPU minutes, and real minutes, which are split into prime and nonprime minutes.

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, total kcore time (memory measurement in kilobyte segments), total CPU time, total real time, mean memory size (in K−bytes), mean CPU time per invocation of the command, and the CPU usage factor. The listed times are all in minutes. The **acctcms** command normally sorts its output by total kcore minutes. The unit kcore minutes is a measure of the amount of memory used (in kilobytes) multiplied by the amount of time it was in use. This flag cannot be used with the **−t** flag.

The default items have the following headings in the output:

```
                TOTAL COMMAND SUMMARY


COMMAND  NUMBER  TOTAL     TOTAL    TOTAL
NAME     CMDS    KCOREMIN  CPU−MIN  REAL−MIN


MEAN    MEAN     HOG     CHARS   BLOCKS
```

```
                SIZE-K  CPU-MIN  FACTOR  TRNSFD  READ
```

−c	Sorts by total CPU time rather than total kcore minutes. When this flag is used with the **−n** flag, only the **−n** flag takes effect.
−j	Combines all commands called only once under the heading `other`.
−n	Sorts by the number of times the commands were called. When this flag is used with the **−c** flag, only the **−n** flag takes effect.
−o	Displays a command summary of nonprime time commands. You can use this flag only when the **−a** flag is used.
−p	Displays a command summary of prime time commands. You can use this flag only when the **−a** flag is used.
−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 nonprime time sections. This flag cannot be used with the **−a** flag.

Security

Access Control: This command should grant execute (x) access only to members of the **adm** group.

Examples

To collect daily command accounting records in a `today` file and maintain a running total in a `total` file, add the following to a shell script:

```
acctcms File . . . > today
cp total previoustotal
acctcms −s today previoustotal > total
acctcms −a −s total
```

The *File* parameters that you specify are redirected to a file called `today`, added to the previous total (in a file renamed `previoustotal`) to produce a new total (called `total`). All files are binary files. In the last line, the **−a** flag displays the `total` file in ASCII format so you can view the report.

Files

/etc/acct/holidays	Specifies prime and nonprime time for accounting records.
/usr/sbin/acct/acctcms	Contains the **acctcms** command.

Related Information

The **lastcomm** command, **runacct** command.

The **acct** file format, **utmp**, **wtmp**, **failedlogin** file format.

The **acct** subroutine.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

Accounting Commands in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

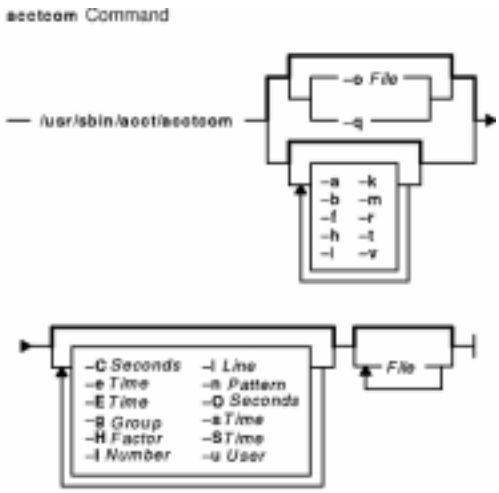
AIX Performance Monitoring and Tuning Commands in *AIX Versions 3.2 and 4 Performance Tuning Guide*.

acctcom Command

Purpose

Displays selected process accounting record summaries.

Syntax



/usr/sbin/acct/acctcom [[**−q** | **−o***File*] | [**−a**] [**−b**] [**−f**] [**−h**] [**−i**] [**−k**] [**−m**] [**−r**] [**−t**] [**−v**]] [**−C** *Seconds*] [**−g** *Group*] [**−H** *Factor*] [**−I** *Number*] [**−l** *Line*] [**−n** *Pattern*] [**−O** *Seconds*] [**−u** *User*] [**−e** *Time*] [**−E** *Time*] [**−s** *Time*] [**−S** *Time*] [*File ...*]

Description

The **acctcom** command reads process accounting records from files specified by the *File* parameter from standard input or from the **/var/adm/pacct** file. Then the **acctcom** command writes the records you request to standard output. This command is stored in the **/usr/sbin/acct** directory, for access by all users.

If you do not specify a *File* parameter and if standard input is assigned to a workstation or to the **/dev/null** file, as when a process runs in the background, the **acctcom** command reads the **/var/adm/pacct** file.

If you specify a *File* parameter, the **acctcom** command reads each file chronologically by process completion time. Usually, the **/var/adm/pacct** file is the current file that you want the **acctcom** command to examine. Because the **ckpacct** procedure keeps this file from growing too large, a busy system may have several **pacct** files. All but the current file have the path name **/var/adm/pacct***?*, where *?* (question mark) represents an integer.

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)
```

If a process was run by the root user, the process name is prefixed with a # (pound sign). If a process is not assigned to a known workstation (for example, when the **cron** daemon runs the process), a ? (question mark)

appears in the TTYNAME field.

Notes:

1. The **acctcom** command only reports 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.

Security

Access Control: This command should grant execute (x) access to all users.

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 the **acctcom** command reads standard input.
−C *Seconds*	Shows only processes whose total CPU time (system time + user time) exceeds the value specified by the *Seconds* variable.
−e *Time*	Selects processes existing at or before the specified time. You can use the current locale to specify the order of hours, minutes, and seconds. The default order is *hh:mm:ss*.
−E *Time*	Selects processes ending at or before the specified time. You can use the current locale to specify the order of hours, minutes, and seconds. The default order is *hh:mm:ss*. If you specify the same time for both the **−E** and **−S** flags, the **acctcom** command displays the processes that existed at the specified time.
−f	Displays two columns related to the ac_flag field of the **acct.h** file: the first indicates use of the **fork** command to create a process, the second indicates the system exit value. Refer to the **acct** structure described in the **acct** file format in *AIX Version 4.3 Files Reference*.
−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 (hog factor). This factor is computed as: `(total CPU time) / (elapsed time)`
−H *Factor*	Shows only the processes that exceed the value of the *Factor* parameter. This factor, called the hog factor, is computed as: `(total CPU time) / (elapsed time)`
−i	Displays columns showing the number of characters transferred in read or write operations (the I/O counts).
−k	Instead of memory size, shows total kcore minutes (memory measurement in kilobyte segments used per minute of run time).
−l *Line*	(lowercase L) Shows only processes belonging to workstation **/dev/***Line*.
−I *Number*	(uppercase i) Shows only processes transferring more than the specified number of characters.
−m	Shows mean main−memory size. This is the default. The **−h** flag or **−k** flag turn off the **−m** flag.
−n *Pattern*	Shows only commands matching the value of the *Pattern* variable,

	where *Pattern* is a regular expression. Regular expressions are described in the **ed** command. In addition to the usual characters, the **acctcom** command allows you to use a + (plus sign) as a special symbol for 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. This flag cannot be used with the **−q** flag.
−O *Seconds*	Shows only processes with CPU system time exceeding the specified number of seconds.
−q	Displays statistics but not output records. The statistics are the same as those displayed using the **−a** flag. The **−q** flag cannot be used with the **−o** flag.
−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. You can use the current locale to specify the order of hours, minutes, and seconds. The default order is *hh:mm:ss*.
−S *Time*	Shows only those processes starting at or after the specified time. You can use the current locale to specify the order of hours, minutes, and seconds. The default order is *hh:mm:ss*.
−t	Shows separate system and user CPU times.
−u *User*	Shows only processes belonging to the specified user. Enter one of the following for the *User* variable: a user ID, a login name to be converted to a user ID, a # (pound sign) to select processes run by the root user, or a ? (question mark) to select processes associated with unknown user IDs.
−v	Eliminates column headings from the output.

Examples

1. To display information about processes that exceed 2 seconds of CPU time, enter:

   ```
   /usr/sbin/acct/acctcom −O 2 < /var/adm/pacct
   ```

   The process information is read from the **/var/adm/pacct** file.

2. To display information about processes belonging to the `finance` group, enter:

   ```
   /usr/sbin/acct/acctcom −g Finance < /var/adm/pacct
   ```

   The process information is read from the **/var/adm/pacct** file.

3. To display information about processes that belong to the **/dev/console** workstation and that run after 5 p.m., enter:

   ```
   /usr/sbin/acct/acctcom −l /dev/console −s 17:00
   ```

The process information is read from the **/var/adm/pacct** file by default.

Files

/usr/sbin/acct/acctcom	Contains the **acctcom** command.
/var/adm/pacct	Contains the current process accounting file.
/etc/group	Contains the basic group attributes of groups.

/etc/passwd	Contains the basic attributes of users.

Related Information

The **ed** command, **ps** command, **runacct** command, **su** command.

The **cron** daemon.

The **acct** subroutine.

The **acct** file format, **utmp**, **wtmp, failedlogin** file format.

Accounting Commands in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

For more information about the accounting system, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

in AIX Performance Monitoring and Tuning Commands in *AIX Versions 3.2 and 4 Performance Tuning Guide*

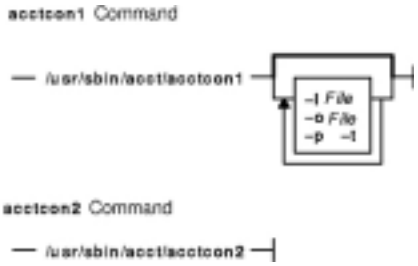
The environment File describes environment variables and their functions.

acctcon1 or acctcon2 Command

Purpose

Performs connect−time accounting.

Syntax



acctcon1 [**−l** *File*] [**−o***File*] [**−p**] [**−t**]

acctcon2

Description

acctcon1

The **acctcon1** command is called by the **runacct** command to convert a sequence of login and logoff records (read from standard input) to a sequence of login session records (written to standard output). Input is normally redirected from the **/var/adm/wtmp** file. The input file can be a file other than **/var/adm/wtmp**, as long as it is in the correct format.

The **acctcon1** command displays the following in ASCII format:

- Login device
- User ID
- Login name
- Prime connect time (seconds)
- Nonprime connect time (seconds)
- Session starting time (numeric)
- Starting date and time (in date/time format)

The **acctcon1** command also maintains a list of ports on which users are logged in. When the **acctcon1** command reaches the end of its input, the command writes a session record for each port that still appears to be active. Unless the **−t** flag is used, the **acctcon1** command assumes that input is a current file and uses the current time as the ending time for each session still in progress.

The summary file generated with the **−l** flag helps an administrator track line usage and identify bad lines. All hang−ups, terminations of the **login** command, and terminations of the login shell cause the system to write logoff records. Consequently, the number of logoffs is often much higher than the number of sessions.

acctcon2

The **acctcon2** command, also called by the **runacct** command, converts a sequence of login session records produced by the **acctcon1** command into connect−time total accounting records. These records are merged with other total accounting records by the **acctmerg** command to produce a daily report.

Flags

Note: The following flags are used with the **acctcon1** command.

−l *File* (lowercase L) Writes a line−usage summary file showing the line name, the number of minutes used, the percentage of total elapsed time, the number of sessions charged, the number of logins, and the number of logoffs. If you do not specify a file name, the system creates the information in the **/var/adm/acct/nite/lineuse** file.

−o *File* Writes to the specified file an overall record for the accounting period, giving starting time, ending time, number of restarts, and number of date changes. If you do not specify a file name, the system creates the **/var/adm/acct/nite/reboots** file.

−p Displays only input. Line name, login name, and time are shown in both numeric and date/time formats. Without the **−p** flag specified, the **acctcon1** command would display input, converting input to session records, and write reports.

−t Uses the last time found in the input as the ending time for any current processes. This, rather than current time, is necessary in order to have reasonable and repeatable values for files that are not current.

Security

Access Control: These commands should grant execute (x) access only to members of the **adm** group.

Examples

1. To convert a sequence of login records (in the **/var/adm/wtmp** file) to a sequence of login session records (stored in the **/var/adm/logsess** file), include the following in a shell script:

   ```
   acctcon1 −t −l/var/adm/acct/nite/lineuse \
   −o/var/adm/acct/nite/reboots \
   </var/adm/wtmp > /var/adm/logsess
   ```

   The login session reports show an ending time that corresponds with the last time input was provided. Two reports are generated: a line−usage summary file named **/var/adm/acct/nite/lineuse**, an overall record for the accounting period, reported in the **/var/adm/acct/nite/reboots** file.

2. To convert a series of login session records (in the **/var/adm/acct/nite/ctmp** file) to a total accounting record (stored in the **/var/adm/logacct** file), include the following in a shell script:

   ```
   acctcon2 < /var/adm/acct/nite/ctmp \
    > /var/adm/logacct
   ```

Files

/usr/sbin/acct/acctcon1 Contains the **acctcon1** command.

/usr/sbin/acct/acctcon2 Contains the **acctcon2** command.

/var/adm/wtmp Contains connect−time accounting data, including login, logout, and shutdown records.

Related Information

The **acctmerg** command, **fwtmp**, **acctwtmp**, or **wtmpfix** command, **init** command, **login** command, **runacct** command.

The **acct** file format, **utmp**, **wtmp**, **failedlogin** file format.

The **acct** subroutine.

For more information about the Accounting System, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

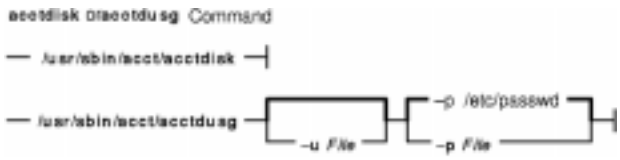
Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

acctdisk or acctdusg Command

Purpose

Performs disk−usage accounting.

Syntax



/usr/sbin/acct/acctdisk

/usr/sbin/acct/acctdusg [**−u***File*] [**−p***File*]

Description

The **acctdisk** and **acctdusg** commands are called by the **dodisk** command to perform disk−usage accounting. Usually, this procedure is initiated when the **cron** daemon runs the **dodisk** command.

Normally, the output of the **diskusg** command becomes the input of the **acctdisk** command. If a more thorough but slower version of disk accounting is needed, use the **dodisk −o** command to call the **acctdusg** command instead of the **diskusg** command.

Accounting is only done for files on the local file system for local users. System administrators who want to count remote users (such as YP clients or diskless clients) should use the **acctdusg −p** command.

acctdisk

The **acctdisk** command reads the output lines of the **diskusg** or **acctdusg** commands from standard input, converts each individual record into a total accounting record, and writes the records to standard output. These records are merged with other accounting records by the **acctmerg** command to produce the daily accounting report.

acctdusg

The **acctdusg** command is called by using the **dodisk −o** command, when a slow and thorough version of disk accounting is needed. Otherwise, the **dodisk** command calls the **diskusg** command.

The **acctdusg** command reads a list of files from standard input (usually piped from a **find / −print** command), computes the number of disk blocks (including indirect blocks) allocated to each file owner, and writes an individual record for each user to standard output. By default, the command searches for login names and numbers in the **/etc/passwd** file. You can search other files by specifying the **−p***File* flag and variable. Each output record has the following form:

```
uid login #blocks
```

The `#blocks` value is the number of 1KB blocks utilized by the user.

Flags

−p *File* Searches the specified file for login names and numbers, instead of searching the **/etc/passwd** file.
−u *File* Places, in the specified file, records of the file names that are exempt from charges.

Security

Access Control: These commands should grant execute (x) access only to members of the **adm** group.

Examples

1. To start normal disk accounting procedures, add a line similar the following to a **crontab** file so that the **cron** daemon runs disk accounting commands automatically:

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk
   ```

   In this example, the **dodisk** procedure runs at 2 a.m. (`0 2`) every Thursday (4) and the **dodisk** procedure calls the **diskusg** and **acctdisk** commands to write disk usage records to the **/usr/adm/acct/nite/dacct** file.

2. To start a thorough disk accounting procedure, add a line similar the following to a **crontab** file so that the **cron** daemon runs disk accounting commands automatically:

   ```
   0 2 * * 4 /usr/sbin/acct/dodisk −o
   ```

In this example, the **dodisk** procedure runs at 2 a.m. (`0 2`) every Thursday (4) and the **dodisk** procedure calls the **acctdusg** and **acctdisk** commands to write disk usage records to the **/var/adm/acct/nite/dacct** file.

Files

/usr/sbin/acct/acctdisk	Contains the **acctdisk** command.
/usr/sbin/acct/acctdusg	Containsthe **acctdusg** command.
/etc/passwd	Contains the basic attributes of user.
/usr/sbin/acct	Directory holding all accounting commands.

Related Information

The **acctmerg** command, **diskusg** command, **dodisk** command, **runacct** command.

The **cron** daemon.

The **acct** file format, **utmp**, **wtmp**, **failedlogin** file format.

The **acct** subroutine.

Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* provides more information about the accounting system, the preparation of daily and monthly reports, and the accounting files.

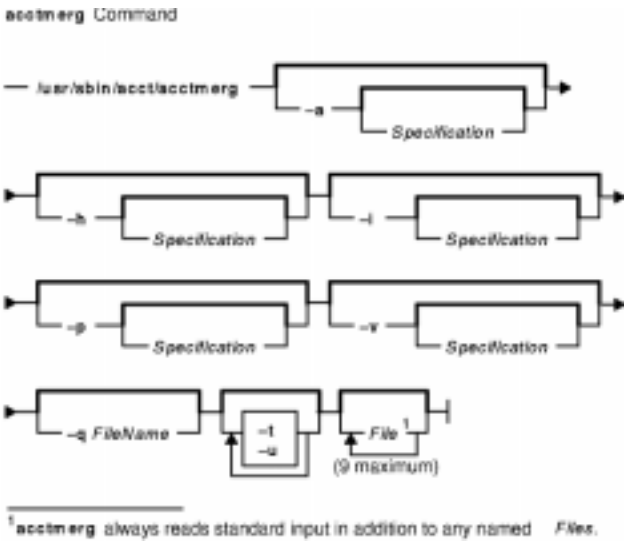
Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

acctmerg Command

Purpose

Merges total accounting files into an intermediary file or a daily report.

Syntax



/usr/sbin/acct/acctmerg [**−a** [*Specification*]] [**−h** [*Specification*]] [**−i** [*Specification*]] [**−p** [*Specification*]] [**−q***Filename*] [**−v** [*Specification*]] [**−t**] [**−u**] [*File ...*]

Description

The **acctmerg** command merges process, connect−time, fee, disk−usage, and queuing (printer) total accounting records (in **tacct** binary or **tacct** ASCII format) and then writes the results to standard output. (See the **tacct** structure in the **acct** File Format for a description of the total accounting format). The **acctmerg** command reads the total accounting records from standard input and from the additional files (up to nine) specified by the *File* parameter. The **acctmerg** command then merges the records by identical keys, usually a user ID and name. To facilitate storage, the **acctmerg** command writes the output in binary format unless you use either the **−a**, **−v**, or **−p** flag.

The **acctmerg** command is called by the **runacct** command to produce either an intermediate report when one of the input files is full, or to merge the intermediate reports into a cumulative total. The intermediate report is stored in the **/var/adm/acct/nite/daytacct** file. The cumulative report is stored in the **/var/adm/acct/sum/tacct** file. The cumulative total is the source from which the **monacct** command produces the ASCII−format monthly summary report. The monthly summary report is stored in the **/var/adm/acct/fiscal** file.

The *Specification* variable allows you to select input or output fields, as illustrated in Example 1. A field specification is a comma−separated list of field numbers, in the order specified in the **tacct** structure in the **acct** File Format. Field ranges may be used, with array sizes taken into account, except for the *ta_name* characters. In the following example:

```
−h2−3,11,15−13,2
```

The **−h** flag causes column headings to display for the following types of data, in this order:

- login name (2)
- prime CPU (3)
- connect time (11)
- fee (15)
- queuing system (14, as implied in the range)
- disk usage data (13)
- the login name again (2)

The default displays all fields, otherwise specified as 1−18 or 1−, and produces wide output lines containing all the available accounting data.

Queueing system, disk usage, or fee data can be converted into **tacct** records by using the **acctmerg −i***Specification* command.

The **tacct** fields are:

No. Header	Description
1 UID	User ID number.
2 LOGIN NAME	Login name of user.
3 CPU PRIME	Cumulative CPU minutes during prime hours.
4 CPU NPRIME	Cumulative during non−prime hours.
5 KCORE PRIME	Cumulative minutes spent in the kernel during prime hours.
6 KCORE NPRIME	Cumulative during non−prime hours.
7 BLKIO PRIME	Cumulative blocks transferred during prime hours.
8 BLKIO NPRIME	Cumulative during non−prime hours.
9 RW/WR PRIME	Cumulative blocks read/written during prime hours.
10 RW/WR NPRIME	Cumulative during non−prime hours.
11 CONNECT PRIME	Cumulative connect time (minutes) during prime hours.
12 CONNECT NPRIME	Cumulative during non−prime hours.
13 DISK BLOCKS	Cumulative disk usage.
14 PRINT	Queuing system charges. (pages)
15 FEES	Fee for special services.
16 # OF PROCS	Count of processes.
17 # OF SESS	Count of login sessions.
18 # OF SAMPLES	Count of count of disk samples.

Flags

−a[*Specification*]	Produces output in the form of ASCII records.
−h[*Specification*]	Displays column headings. This flag implies the **−a** flag, but is effective with **−p** or **−v**.
−i[*Specification*]	Expects input files composed of ASCII records, which are converted to binary records.
−p[*Specification*]	Displays input without processing. The output is in ASCII format.
−q*Filename*	Reads the specified **qacct** file (**accrec.h** file format) and produces output records sorted by user ID and user name. These records contain the user ID, user name, and number of pages printed.
−t	Produces a single record that contains the totals of all input.
−u	Summarizes by user ID rather than by user name.
−v[*Specification*]	Produces output in ASCII format, with more precise notation for floating−point numbers.

Security

Access Control: This command should grant execute (x) access only to members of the **adm** group.

Examples

1. To merge disk accounting file `dacct` with field specification `−i1−2,13,18` into an existing total accounting file, `tacct`, enter:

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

   The **acctmerg** command reads the field specifications for the user ID, login name, number of blocks, and number of disk samples (`i1−2,13,18`) from the **dacct** file, merges this information with a **tacct** record, and writes the result to standard output.

2. To make repairs to the **tacct** format file `jan2.rpt`, first enter:

   ```
   acctmerg −v <Jan.2.rpt >jan2.tmp
   ```

   Now edit the file `jan2.tmp` as desired. This command redirects the content of `Jan2.rpt` to `Jan2.tmp`, with the output in ASCII format.

3. To redirect `Jan2.tmp` to `Jan2.rpt`, with the output in binary record format, enter the following command:

   ```
   acctmerg −i <jan2.tmp >jan2.rpt
   ```

Files

/usr/sbin/acct/acctmerg	Contains the **acctmerg** command.
/usr/include/sys/acct.h	Contains the **acct** and **tacct** file formats.
/var/adm/acct/nite/daytacct	Contains an intermediate daily total accounting report in binary format.
/var/adm/acct/sum/tacct	Contains the cumulative total accounting report for the month in binary format.
/var/adm/acct/fiscal	Contains the monthly accounting summary report, produced from the records in the **/var/adm/acct/sum/tacct** file.

Related Information

The **acctcms** command, **acctcom** command, **acctcon1** or **acctcon2** command, **acctdisk** command, **acctprc1**, **acctprc2**, or **accton** command, **fwtmp** command, **runacct** command.

The **acct** file format, **utmp**, **wtmp**, **failedlogin** file format.

The **acct** subroutine.

Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Queuing System Overview for System Management in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

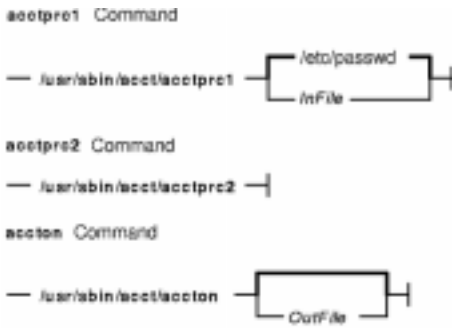
Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

acctprc1, acctprc2, or accton Command

Purpose

Performs process−accounting procedures.

Syntax



/usr/sbin/acct/acctprc1 [*InFile*]

/usr/sbin/acct/acctprc2

/usr/sbin/acct/accton [*OutFile*]

Description

The three **acctprc** commands, **acctprc1**, **acctprc2**, and **accton,** are called by the **runacct** command to perform process−accounting shell procedures.

The **acctprc1** command reads records from standard input that are in the **acct** format, 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 1024−byte units), the total number of blocks read and written, and mean memory size (in 64−byte units) for each process.

If specified, the *InFile* parameter contains a list of login sessions in **utmp** format, sorted by user ID and login name. If the *File* parameter is not specified, **acctprc1** gets login names from the**/etc/passwd** password file. The information in the *InFile* parameter helps distinguish among different login names that share the same user ID.

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

When the **accton** command is used without parameters, process accounting is turned off. If you specify the *OutFile* parameter (an existing file), process accounting is turned on, and the kernel adds records to that file. You must specify the *OutFile* parameter for process accounting to start. The *OutFile* parameter is not created by the **accton** command. The file specified by the *OutFile* parameter must already exist with the proper group, owner, and permissions. Many shell scripts expect the **/var/adm/pacct** file.

Security

Access Control: These commands should grant execute (x) access only to members of the **adm** group.

Examples

1. To add a user name to each process−accounting record in a binary file and convert the records to an ASCII file named `out.file`, enter the following commands or use the lines in a shell script:

   ```
   /usr/sbin/acct/acctprc1 < /var/adm/pacct >out.file
   ```

2. To produce a total accounting record of the ASCII output file in example 1, enter the following commands or use the lines in a shell script:

   ```
   /usr/sbin/acct/acctprc2 < out.file > \
   /var/adm/acct/nite/daytacct
   ```

   The resulting file is a binary total accounting file in **tacct** format, containing individual records sorted by user ID. The file `/var/adm/acct/nite/daytacct` is merged with other total accounting records by the **acctmerg** command to produce the daily summary record in the **/var/adm/acct/sum/tacct** file.

3. To turn off process accounting, enter:

   ```
   /usr/sbin/acct/accton
   ```

Files

/usr/sbin/acct/acctprc1	Contains the **acctprc1** command.
/usr/sbin/acct/acctprc2	Contains the **acctprc2** command.
/usr/sbin/acct/accton	Contains the **accton** command.
/etc/accton	Symbolic link to the actual **accton** command directory.
/etc/passwd	Contains the basic user attributes, including the user IDs used by the **acctprc1** command.

Related Information

The **acctmerg** command,**runacct** command.

The **acct** file format, **utmp** file format.

For more information about the accounting system, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

Accounting Commands in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

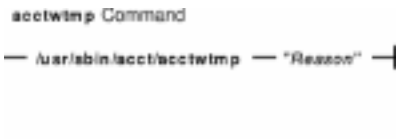
AIX Performance Monitoring and Tuning Commands in *AIX Versions 3.2 and 4 Performance Tuning Guide*

acctwtmp Command

Purpose

Manipulates connect−time accounting records by writing a **utmp** record to standard output.

Syntax



/usr/sbin/acct/acctwtmp "*Reason*"

Description

The **acctwtmp** command is called by the **runacct** command to write a **utmp** record to standard output. The standard output includes the current date and time, plus a *Reason* string of 11 characters or less that you must enter.

Flags

None.

Parameters

Reason String of 11 characters or less.

Security

Access Control: These commands should grant execute (x) access only to members of the **adm** group.

Files

/usr/sbin/acct/accwtmp	Contains the **acctwtmp** command.
/var/adm/wtmp	Contains records of date changes that include an old date and a new date.
/usr/include/utmp.h	Contains history records that include a reason, date, and time.

Related Information

The **acctcon1** or **acctcon2** command, **acctmerg** command, **fwtmp** command, **runacct** command, **wtmpfix** command.

Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the steps you must take to establish an accounting system.

See the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* for a list of accounting commands that can be run automatically or entered from the keyboard and about the preparation of daily and monthly reports, and the accounting files.

acfgd Daemon

Purpose

Provides auto−configuration facility for PCMCIA devices.

Syntax

/usr/sbin/acfgd

Description

The **acfgd** daemon (Auto−config daemon) enables auto−configuration of PCMCIA cards. It can run scripts before or after the configuration or unconfiguration of PCMCIA devices. In addition, the **acfgd** daemon can also run configure or unconfigure methods when a PCMCIA device is inserted or removed.

Releated Device ODM Attributes

The following ODM attributes (PdAt or CuAt) in PCMCIA devices are read by the auto−config daemon. If the corresponding device is configured or unconfigured, the commands defined in the following ODM attributes are also run with bin (UID=2) user authority. TCP/IP configuration scripts, for example, are defined in these attributes of PCMCIA LAN cards.

ODM Attribute Name	Description
pre_mkdev	Commands performed before a configure method. If the command exits with non 0 value, the configure method is not performed.
post_mkdev	Commands performed after a configure method. If a configure method exits with non 0 value, the command is not performed.
pre_rmdev	Commands performed before an unconfigure method. If the command exits with non 0 value, the unconfigure method is not performed.
post_rmdev	Commands performed after an unconfigure method. If an unconfigure method exits with non 0 value, the command is not performed.

The following strings are interpreted by the auto−config daemon when it reads the above attributes.

Original String	Interpreted String
%d	ODM name of the target device.
%n	Number following the device name prefix in ODM target device name.
%%	%

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Security

Access Control: Any User

Files Accessed: None

Example

To enable auto−configuration of PCMCIA cards, enter:

```
/usr/sbin/acfgd
```

Files

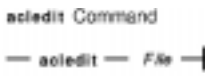
/usr/sbin/acfgd Contains the **acfgd** daemon.

acledit Command

Purpose

Edits the access control information of a file.

Syntax



acledit*File*

Description

The **acledit** command lets you change the access control information of the file specified by the *File* parameter. The command displays the current access control information and lets the file owner change it with the editor specified by the **EDITOR** environment variable. Before making any changes permanent, the command asks if you want to proceed.

> **Note:** The **EDITOR** environment variable must be specified with a complete path name; otherwise, the **acledit** command will fail. The entire ACL for a file cannot exceed one memory page (4096 bytes).

The access control information that displays includes a list of attributes, base permissions, and extended permissions.

The following is an example of the access control information of a file:

```
attributes: SUID
base permissions:
    owner  (frank): r w -
    group (system): r - x
    others        : - - -
extended permissions:
    enabled
        permit    r w -    u:dhs
        deny      r - -    u:chas,    g:system
        specify   r - -    u:john,    g:gateway, g:mail
        permit    r w -    g:account, g:finance
```

Base permissions are assigned to the file owner, group and other users and are the traditional read (r), write (w), and execute (x). Extended permissions give the owner of a file the ability to define access to that file more precisely. Three attributes can be added: **setuid** (**SUID**), **setgid** (**SGID**) and **savetext** (**SVTX**). For a complete discussion refer to the Access Control Lists.

> **Note:** If the **acledit** command is operating in a trusted path, the editor must have the **trusted process** attribute set.

Security

Access Control: This command should be a standard user command and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
x	**/usr/bin/aclget**
x	**/usr/bin/aclput**

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the **acledit** command will generate the following audit record (event) every time the command is executed:

Event	Information
FILE_Acl	Lists access controls.

See "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more details about how to properly select and group audit events, and how to configure audit event data collection.

Examples

To edit the access control information of the `plans` file, enter:

```
acledit plans
```

Files

/usr/bin/acledit Contains the **acledit** command.

Related Information

The **aclget** command, **aclput** command, **auditpr** command, **chmod** command.

Access Control Lists in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

The Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains more about audits and audit events.

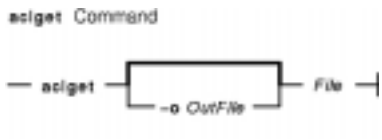
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Introduction in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

aclget Command

Purpose

Displays the access control information of a file.

Syntax



aclget [**−o***OutFile*] *File*

Description

The **aclget** command writes the access control information of the file specified by the *File* parameter to standard output or to the file specified by the *OutFile* parameter.

The information that you view includes attributes, base permissions, and extended permissions. To see an example of access control information, refer to Access Control List.

Flags

−o*OutFile* Specifies that the access control information be written to the file specified by the *OutFile* parameter.

Security

Access Control: This command should be a standard user program and have the **trusted computing base** attribute.

Access Control Lists

In an access control list, attributes, base and extended permissions are in the following format:

```
Attributes: ( SUID | SGID | SVTX )

Base Permissions:

Owner (name): Mode

Group (group): Mode

Others: Mode

Extended Permissions: ( Enabled | Disabled )
```

```
Permit  Mode   u:Username,g:groupname

Deny    Mode   u:Username,g:groupname

Specify Mode   u:Username,g:groupname
```

The access modes are: read (r), write (w), and execute/search (x), with the Mode parameter expressed as rwx (with a dash replacing each unspecified permission)

For example, the following ACL indicates that the file belongs to user *user1* and the group *staff*. In addition, the user *user2* has read access for the file:

```
Attributes:

Base Permissions:

Owner (user1): rw-

Group (group): r--

Others: ---

Extended Permissions:  Enabled

Permit  r--   u:user2
```

The following ACL indicates that the file belongs to same user the group, but in this example, every other user has read access except for *user2*:

```
Attributes:

Base Permissions:

Owner (user1): rw-

Group (group): r--

Others: r--

Extended Permissions:  Enabled

Deny  r--   u:user2
```

Examples

1. To display the access control information for the `status` file, enter:

   ```
   aclget  status
   ```

   An access control list appears, similar to the example in Access Control Lists.

2. To copy the access control information of the `plans` file to the `status` file, enter:

   ```
   aclget  plans  |  aclput  status
   ```

   This copies the access control information.

3. To save the access control information of the `plans` file in the `acl1` file to edit and use later, enter:

   ```
   aclget  -o  acl1  plans
   ```

Files

/usr/bin/aclget Contains the **aclget** command.

Related Information

The **acledit** command, **aclput** command, **chmod** command.

Access Control Lists in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

The Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains more about audits and audit events.

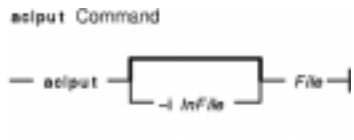
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Introduction in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

aclput Command

Purpose

Sets the access control information of a file.

Syntax



aclput [**−i***InFile*] *File*

Description

The **aclput** command sets the access control information of the file specified by the *File* parameter. The command reads standard input for the access control information, unless you specify the **−i** flag.

> **Note:** If you are reading from standard input your entries must match the format of the access control information or you will get an error message. Use the Ctrl−D key sequence to complete the session.

Access Control List

In an access control list, attributes, base and extended permissions are in the following format:

```
Attributes: ( SUID | SGID | SVTX )

Base Permissions:

   Owner (name): Mode

   Group (group): Mode

   Others: Mode

Extended Permissions: ( Enabled | Disabled )

   Permit  Mode   u:Username,g:groupname

   Deny    Mode   u:Username,g:groupname

   Specify Mode   u:Username,g:groupname
```

The access modes are: read (r), write (w), and execute/search (x), with the Mode parameter expressed as rwx (with a dash replacing each unspecified permission)

For example, the following ACL indicates that the file belongs to user *user1* and the group *staff*. In addition, the user *user2* has read access for the file:

```
Attributes:
```

```
Base Permissions:

    Owner (user1): rw-

    Group (group): r--

    Others: ---

Extended Permissions:  Enabled

    Permit  r--   u:user2
```

The following ACL indicates that the file belongs to same user the group, but in this example, every other user has read access except for *user2*:

```
Attributes:

Base Permissions:

    Owner (user1): rw-

    Group (group): r--

    Others: r--

Extended Permissions:  Enabled

    Deny  r--   u:user2
```

Flags

-i*InFile* Specifies the input file for access control information. If the access control information in the file specified by the *InFile* parameter is not correct, when you try to apply it to a file, an error message preceded by an asterisk is added to the input file.

Note: The entire Access Control List for a file cannot exceed one memory page (4096 bytes).

Security

Access Control: This command should be a standard user program and have the **trusted computing base** attribute.

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the **aclput** command will generate the following audit record (event) every time the command is executed:

Event	Information
FILE_Acl	Lists file access controls.

See "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more details about how to properly select and group audit events, and how to configure audit event data collection.

Examples

1. To set the access control information for the status file with information from standard input, enter:

   ```
   aclput status
   attributes: SUID
   ```

and then press the Ctrl−D sequence to exit the session.

2. To set the access control information for the `status` file with information stored in the `acldefs` file, enter:

   ```
   aclput   −i   acldefs   status
   ```

3. To set the access control information for the `status` file with the same information used for the `plans` file, enter:

   ```
   aclget   plans   |   aclput   status
   ```

4. To set the access control information for the `status` file with an edited version of the access control information for the `plans` file, you must enter two commands. First, enter:

   ```
   aclget   −o acl   plans
   ```

   This stores the access control information for the `plans` file in the `acl` file. Edit the information in the `acl` file, using your favorite editor. Then, enter:

   ```
   aclput   −i   acl   status
   ```

This second command takes the access control information in the `acl` file and puts it on the `status` file.

Files

/usr/bin/aclput Contains the **aclput** command.

Related Information

The **acledit** command, **aclget** command, **auditpr** command, **chmod** command.

Access Control Lists in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

The Auditing Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains more about audits and audit events.

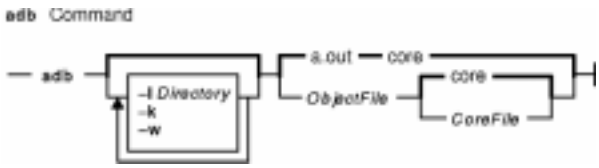
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Introduction in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

adb Command

Purpose

Provides a general purpose debug program.

Syntax



adb [**−k**] [**−l** *Directory*] [**−w**] [*ObjectFile* [*CoreFile*]]

Description

The **adb** command provides a debug program for programs. With this debug program, you can examine object and core files and provide a controlled environment for running a program.

Normally, the *ObjectFile* parameter is an executable program file that contains a symbol table. If the *ObjectFile* parameter does not contain a symbol table, the symbolic features of the **adb** command cannot be used, although the file can still be examined. The default for the *ObjectFile* parameter is **a.out**.

The *CoreFile* parameter is a core image file produced by running the *ObjectFile* parameter. The default for the *CoreFile* parameter is **core**.

While the **adb** command is running, it takes standard input and writes to standard output. The **adb** command does not recognize the Quit or Interrupt keys. If these keys are used, the **adb** command waits for a new command.

In general, requests to the **adb** command are in the following form:

[*Address*] [,*Count*] [*Command*] [;]

where *Address* and *Count* are expressions. The default for the *Count* expression is a value of 1. If the *Address* expression is specified, the **.** (period) variable is set to *Address*.

The interpretation of an address depends on the context in which it is used. If a subprocess is being debugged, addresses are interpreted in the usual way in the address space of the subprocess.

Enter more than one command at a time by separating the commands with a **;** (semicolon).

The **adb** debug program allows the use of various:

- expressions
- operators
- subcommands
- variables
- addresses

See the adb Debug Program Overview in *AIX General Programming Concepts: Writing and Debugging Programs* for detailed information.

Note: If the object file does not contain the symbol table, the **adb** command will not be able to show the value of static, automatic, and external variables of a program.

Flags

−k	Causes kernel mapping.
−l *Directory*	Specifies a directory where files to be read with $< or $<< are sought. The default is the **/usr/ccs/bin/adb** file.
−w	Opens the *ObjectFile* and the *Corefile* parameters for reading and writing. If either file does not exist, this flag creates the file.

Return Values

The **adb** debug program is printed when there is no current command or format. The **adb** command indicates such things as inaccessible files, syntax errors, and abnormal termination of commands. Exit status is a value of 0, unless the last command was unsuccessful or returned non−zero status.

Files

/dev/mem	Provides privileged virtual memory read and write access.
a.out	Provides common assembler and link editor output.
core	Contains an image of a process at the time of an error.

Related Information

The **cc** command, **dbx** command.

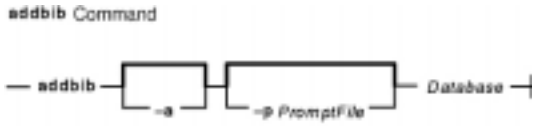
adb Debug Program Overview in *AIX General Programming Concepts: Writing and Debugging Programs*.

addbib Command

Purpose

Creates or extends a bibliographic database.

Syntax



addbib [**−a**] [**−p** *PromptFile*] *Database*

Description

The **addbib** command uses a series of prompts to guide the user through creating or extending a bibliographic database. The user can define responses to these prompts. All default prompts and instructions are contained in the **refer** message catalog.

The first prompt is `Instructions?`. If the answer is affirmative, you can receive directions.

If the answer is negative or if you press the Enter key, you cannot receive directions. The **addbib** command then prompts for various bibliographic fields, reads responses from the terminal, and sends output records to the database specified by the *Database* parameter.

Pressing the Enter key (a null response) means to omit a particular field. Typing a − (minus sign) means to return to the previous field. A trailing backslash allows a field to be continued on the next line. The repeating `Continue?` prompt allows you to resume, to quit the current session, or to edit the database. To resume, type the defined affirmative answer or press the Enter key. To quit the current session, type the defined negative answer.

To edit the database, enter any system text editor (vi, ex, edit, ed).

Flags

−a Suppresses prompting for an abstract. Prompting for an abstract is the default. Abstracts are ended by pressing a Ctrl−D key sequence.

−p*PromptFile* Causes the **addbib** command to use a new prompting skeleton, which is defined in the file specified by the *PromptFile* parameter. This file contains prompt strings, a tab, and the key letters written to the specified database.

The following are the most common key letters and their meanings. The **addbib** command insulates you from these key letters, since it gives you prompts in English. If you edit the bibliography file later, you 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 the **refer** command)

%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** flag of the **refer** command

%M
Bell Labs memorandum (undefined)

%N
Number within volume

%O
Other commentary, printed at end of reference

%P
Page numbers

%Q
Corporate or foreign author (unreversed)

%R
Report, paper, or thesis (unpublished)

%S
Series title

%T
Title of article or book

%V
Volume number

%X
Abstract used by the **roffbib** command, not by the **refer** command

%Y,Z
Ignored by the **refer** command.

Note: Except for the **%A** key letter, each field should be given just once. Only relevant fields should be supplied.

Examples

The following is an example of a bibliography file:

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

Related Information

The **indxbib** command, **lookbib** command, **refer** command, **roffbib** command, **sortbib** command.

addX11input Command

Purpose

Adds an X11 input extension record into the ODM (Object Data Manager) database.

Syntax

addX11input

Description

The **addX11input** command is used to add an X11 input extension record into the ODM database. When you enter **addX11input** on the command line, the **addX11input** command requests *DeviceName, GenericName,* and *ModuleName* values in turn. The entire record is then added to the ODM database.

The command is a root/system user command. Its action fails with a permissions error if an unauthorized user attempts to add a record.

Error Codes

ODM could not open class	Returned if the X11 Input extension records in the ODM database are not found in the **/usr/lib/objrepos** directory.

Related Information

The **deleteX11input** command, **listX11input** command.

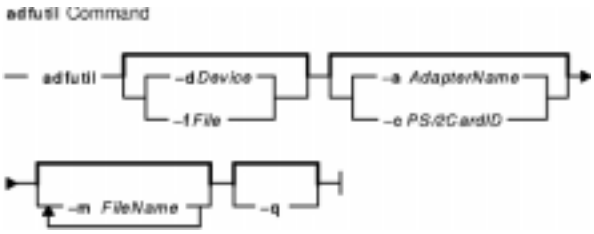
Calls and Functions Reference in *AIX Version 4.3 Technical Reference*

adfutil Command

Purpose

Provides the capability to merge Micro Channel information for PS/2 adapters with the Version 3 Configuration Database.

Syntax



adfutil [**−d** *Device* | **−f** *File*] [**−a** *AdapterName* | **−c** *PS/2CardID*] [**−m** *FileName* ...] [**−q**]

Description

The **adfutil** command provides the capability to field merge Micro Channel resource information for existing PS/2 adapters with predefined information in the Version 3 Configuration database. This is accomplished by processing information found on DOS formatted diskettes provided with the PS/2 adapter hardware. Included on these diskettes are adapter description files that are ASCII representations of adapter hardware attributes.

The naming convention for an adapter description file found on the DOS formatted diskette is @XXXX.ADF where XXXX is the *PS/2CardID*. If the command is invoked without arguments, the search centers around the home directory of the default device. If no files are found in the form of @XXXX.ADF, an error message is sent to standard output and the **adfutil** command ends. If a single adapter description file is found, execution is continued on that file. If multiple adapter description files are found, an error message is written to standard output and processing ends. If the **−c** flag is specified, a string is built that represents the corresponding DOS file name representation of the desired adapter description file. If this file does not exist on the specified device and path name or default, an error message is sent to standard output and processing ends.

When the adapter description file is found, the contents are written in the **/tmp/adf***nnn* file where *nnn* is the current process ID. This ID is removed after successful completion of the command.

Microcode files can be loaded independently of any adapter description file processing, and without disturbing the adapters database representation. Use the **−m** flag to load microcode files into the **/usr/lib/microcode** directory.

> **Attention:** Micro Channel adapters require bus attribute processing beyond what is supported by the bus configuration program, and should not be added to the system due to the possibility of adversely effecting the configuration of other devices on the system.

Flags

−a*AdapterName* Searches the ODM database for candid information to form correct DOS filename for the adapter description file. The *AdapterName* parameter is a valid device name.

−c *PS/2CardID* Identifies the *PS/2CardID* for the adapter. The card identifier is a four character

	alpha−numeric string that is found in the root of the DOS filename of the adapter description file. There is no default.
−d *Device*	Identifies the *Device* where the adapter description file resides. The default is **/dev/fd0**.
−f *File*	Identifies the file system path name for source adapter description file. If the **−f** flag is specified, any microcode keyword found in the adapter description file must specify a file system path name of the microcode source file.
−m*FileName*	Loads only microcode files found on diskette. If the *FileName* parameter is specified, files are loaded into the **/usr/lib/microcode** file. This is a microcode only flag.
−q	Toggles off the message to insert the adapter description file diskette.

Examples

1. To search the diskette drive **/dev/fd0** in the home directory for an adapter description file, enter:

   ```
   adfutil
   ```

2. To read **/home/owner/adf.file** as an adapter description file, enter:

   ```
   adfutil −f /home/owner/adf.file
   ```

3. To search the default device **/dev/fd0** for the adapter description file labeled @0FFE.ADF without interrupting execution for the `insert diskette` prompt, enter:

   ```
   adfutil −c 0FFE −q
   ```

4. To load adapter microcode without processing adapter description file information, enter:

   ```
   adfutil −m FileName
   ```

Related Information

The **dosdir** command.

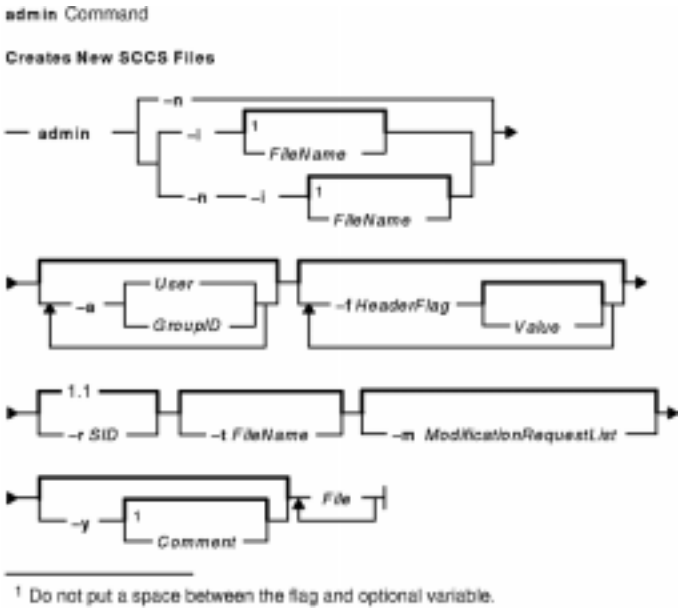
admin Command (SCCS)

Purpose

Creates and controls Source Code Control System (SCCS) files.

Syntax

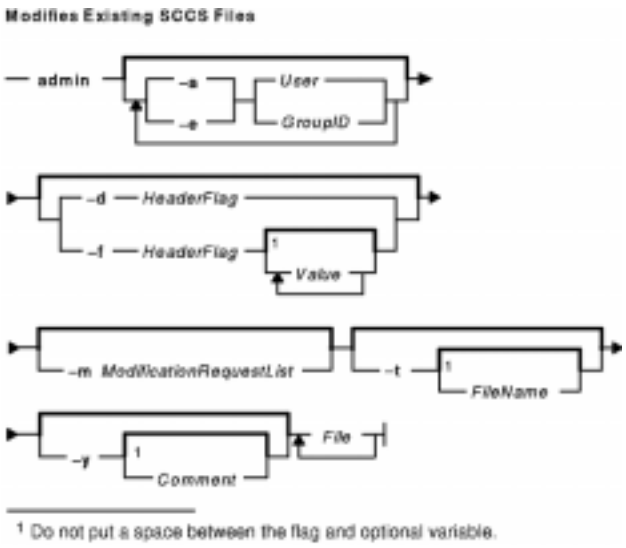
To Create New SCCS Files



admin { **−n** **−i**[*FileName*] } [**−a** {*User* | *GroupID* }] ... [**−f***HeaderFlag*[*Value*] ...] [**−r***SID*] [**−t** *FileName*] [**−m***ModificationRequestList*] [**−y**[*Comment*]] *File* ...

Note: Do not put a space between a flag and an optional (bracketed) variable.

To Modify Existing SCCS Files

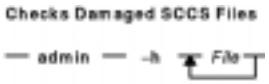


admin [**−a** {*User* | *GroupID* }] ... [**−e** { *User* | *GroupID* }] ... [{ **−d** *HeaderFlag* | **−f** *HeaderFlag*[*Value*]

... }] [**−m***ModificationRequestList*] [**−t**[*FileName*]] [**−y**[*Comment*]] *File ...*

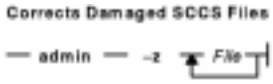
Note: Do not put a space between a flag and an optional (bracketed) variable.

To Check Damaged SCCS Files



admin−h *File ...*

To Correct Damaged SCCS Files



admin−z *File ...*

Description

The **admin** command creates new Source Code Control System (SCCS) files or changes specified parameters in existing SCCS files.

The **admin** command can change the parameters controlling how the **get** command builds the files that you can edit. The parameters can also set conditions about who can access the file and which releases of the files may be edited.

If the file specified by the *File* parameter exists, the **admin** command modifies the file as specified by the flags. If the file does not exist and you supply the **−i** or **−n** flag, the **admin** command creates a new file and provides default values for unspecified flags.

If you specify a directory name for the *File* parameter, the **admin** command performs the requested actions on all SCCS files in that directory. All SCCS files contain the **s.** prefix before the file name. If you use a **−** (minus sign) for the *File* parameter, the **admin** command reads standard input and interprets each line as the name of an SCCS file. An end−of−file character ends input.

You must have write permission in the directory to create a file. All SCCS file names must have the form **s.***Name*. New SCCS files are created with read−only permission. The **admin** command writes to a temporary x−file, which it calls **x.***Name*. If it already exists, the x−file has the same permissions as the original SCCS file. The x−file is read−only if the **admin** command must create a new file. After successful completion of the **admin** command, the x−file is moved to the name of the SCCS file. This ensures that changes are made to the SCCS file only if the **admin** command does not detect any errors while 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). The 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.

You can enter flags and input file names in any order. All flags apply to all the files. Do not put a space between a flag and an optional variable (variable enclosed in bracket). Header flags can be set with the **−f** flag and unset with the **−d** flag. Header flags control the format of the g−file created with the **get** command.

Flags

−a*User* or **−a** *GroupID*	Adds the specified user to the list of users that can make sets of changes (deltas) to the SCCS file. The *User* value can be either a user name or a group ID. Specifying a group ID 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. If the *User* or *GroupID* parameter is preceded by an ! (exclamation point), specified users are denied permission to make deltas. For example, enter `−a !User`.
−d*HeaderFlag*	Deactivates the effects of the specified header flag within 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. Refer to the list of header flags that follows to learn more about the supported values.
−e*User* or **−e***GroupID*	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.
−f*HeaderFlag*[*Value*]	Activates the specified header flag and value in the SCCS file. You can specify more than one header flag in a single **admin** command. There are 12 header flags. Refer to the list of header flags that follows to learn more about the supported values. Do not put a space between the *HeaderFlag* and *Value* variables.
−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 damaged. This flag helps you detect damage caused by the improper use of non−SCCS commands to modify SCCS files, as well as accidental damage. 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 re−compute the checksum. Then test to see if the file is corrected by using the **−h** flag again.
−i[*FileName*]	Gets the text for a new SCCS file from the *FileName* variable. This text is the first delta of the file. If you specify the **−i** flag but omit the file name, the **admin** command reads the text from standard input until it reaches an end−of−file character. If you do not specify the **−i** flag, but you do specify the **−n** flag, the command creates an empty SCCS file. The **admin** command can only create one file containing text at a time. If you are creating two or more SCCS files with one call to the **admin** command, you must use the **−n** flag, and the SCCS files created will be empty. Each line of the file specified by the *FileName* variable cannot contain more than 512 characters. The file name can include MBCS (multibyte character set) characters. Do not put a space between the flag and the *FileName* variable.
−m*ModificationRequestList*	Specifies a list of Modification Request (MR) numbers to be inserted into the SCCS file as the reason for creating the initial delta. A null or empty list can be considered valid, depending on the validation program used. The **v** header flag must be set. The MR numbers are validated if the **v** header flag has a value (the name of an MR number validation program). The **admin** command reports an

error if the **v** header flag is not set or if MR validation fails.

−n Creates a new, empty SCCS file. When the **−n** flag is used without the **−i** flag, the SCCS file is created with control information but without any file data.

−r*SID* Specifies the SCCS identification string (SID) file version to be created. The *SID* variable accepts a delta with four levels: release, level, branch, and sequence, for example 3.2.5.1. If only release is specified, the **admin** command automatically assumes level 1. If you do not specify the **−r** flag, the initial delta becomes release 1, level 1 (that is, 1.1). For more details on specifying the SID, refer to the SID Determination table described in the **get** command.

You can specify the **−r** flag only if you also specify the **−i** or **−n** flag. Use this flag only when creating an SCCS file.

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

- Without a file name, the **−t** flag removes any descriptive text currently in the SCCS file.
- With a file name, the **−t** flag replaces any descriptive text currently in the SCCS file with text in the named file.
- The file name can include MBCS (multibyte character set) characters.

Do not put a space between the flag and the *FileName* variable.

−y [*Comment*] Inserts the specified comment 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, the **admin** command inserts a line of the following form:

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

The comments can include MBCS (multibyte character set) characters. Do not put a space between the flag and the *FileName* variable.

−z Re−computes the SCCS file checksum and stores it in the first line of the SCCS file (see the **−h** flag).

> **Attention:** Using the **admin** command with the **−z** 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.

File Specifies the name of the file created or altered by the **admin** command. If a **−** (minus sign) is specified, the **admin** command reads from standard input. An end−of−file character ends standard input.

Header Flags

The following list contains the header flags that can be set with the **−f** flag and unset with the **−d** flag. Header flags control the format of the g−file created with the **get** command.

b Lets you use the **−b** flag of a **get** command to create branch deltas.

c*Number* Makes the *Number* variable the highest release number that a **get −e** command can use. The value of the *Number* variable must be greater than 0 and less than or equal to 9999. (The default value is 9999.)

d*SID* Makes the *SID* variable the default delta supplied to a **get** command.

f*Number* Makes the *Number* variable the lowest release number that a **get −e** command can retrieve. The *Number* variable must be greater than 0 and less than 9999. (The default value is 1.)

i [*String*] Treats the following informational message, issued by the **get** or **delta** command, as an error:

```
There are no SCCS identification keywords in the file. (cm7)
```

In the absence of this flag, the message is only a warning. The message is issued if no SCCS identification keywords are found in the text retrieved or stored in the SCCS file (refer to the **get** command). If a string is supplied, the keywords must match exactly the given string. The string must contain a keyword and have no embedded newlines.

j Permits concurrent **get** commands for editing the same SID of an SCCS file. Use of the **j** header flag allows multiple concurrent updates to the same version of the SCCS file.

l*List* (lowercase L) Locks the releases specified by the *List* variable against editing, so that a **get −e** command against one of these releases fails. The list has the following syntax:

```
<List> : : = <Range> | <List> , <Range>
<Range> : : = SID | a
```

where character a in the list is equivalent to specifying all releases for the named SCCS file.

m*Module* Substitutes the *Module* variable for all occurrences of the **%M%** keyword in an SCCS text file retrieved by a **get** command. The default *Module* variable is the name of the SCCS file without the **s.** prefix. The module name can include MBCS (multibyte character set) characters.

n Causes the **delta** command 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. Releases 3 and 4 will be created as null delta entries in the delta table of the **s.** file. 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 the specified text for all occurrences of the **%Q%** keyword in an SCCS text file retrieved by a **get** command.

t*Type* Substitutes specified type for all **%Y%** keywords in a g−file retrieved by a **get** command.

v [*Program*] Makes the **delta** command prompt for Modification Request (MR) numbers as the reason for creating a delta. The *Program* variable specifies the name of an MR−number validity−checking program. If the **v** flag is set in the SCCS file, the **−m** flag must also be used, even if its value is null. The program name can include MBCS (multibyte character set) characters.

Locating Damaged SCCS Files

Although SCCS provides some error protection, you may need to recover a file that was accidentally damaged. This damage may result from a system malfunction, operator error, or changing an SCCS file without using SCCS commands.

SCCS commands use the checksum to determine whether a file was changed since it was last used. The only SCCS command that processes a damaged file is the **admin** command when used with the **−h** or **−z** flags. The **−h** flag tells the **admin** command to compare the checksum stored in the SCCS file header against the computed checksum. The **−z** flag tells the command to re−compute the checksum and store it in the file header.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

These examples use an imaginary text file called `test.c` and an editor such as **ed** to edit files.

1. First, create an ordinary SCCS file. To create an empty SCCS file named `s.test.c`, enter:

   ```
   $ admin −n s.test.c
   ```

   Using the **admin** command with the **−n** flag creates an empty SCCS file.

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

   ```
   $ admin −itest.c s.test.c
   There are no SCCS identification keywords in the file (cm7)
   $ li
   s.test.c test.c
   ```

   If you use the **−i** flag, the **admin** command creates delta 1.1 from the specified file. Once delta 1.1 is created, rename the original text file so it does not interfere with SCCS commands:

   ```
   $ mv test.c back.c
   ```

   The message `There are no SCCS identification keywords in the file (cm7)` does not indicate an error. SCCS writes this message when there are no identification keywords in the file. Identification keywords are variables that can be placed in an SCCS file. The values of these variables provide information such as date, time, SID, or file name. See the **get** command for an explanation of identification keywords. If no identification keywords exist, SCCS writes the message. However, if the **i** header flag is set in the **s.** file, this message causes an error condition. This flag is set by the user.

   Give the SCCS file any name, beginning with **s.**. In the preceding example, the original file and the SCCS file have the same name, but that is not necessary.

   Because you did not specify a release number, the **admin** command gave the SCCS file an SID of 1.1. SCCS does not use the number 0 to identify deltas. Therefore, a file cannot have an SID of 1.0 or 2.1.1.0, for example. All new releases start with level 1.

3. To start the `test.c` file with a release number of 3.1, use the **−r** flag with the **admin** command, as shown below, and enter:

   ```
   $ admin −itest.c −r3 s.test.c
   ```

   To restrict permission to change SCCS files to a specific set of user IDs, list user IDs or group ID numbers in the user list of the SCCS file by using the **−a** flag of the **admin** command. This flag may appear multiple times on the command line. These IDs then appear in the SCCS file header. Without the **−a** flag to restrict access, all user IDs can change the SCCS files.

4. To restrict edit permission to the user ID `dan`, enter:

   ```
   $ admin −adan s.test.c
   ```

5. Check SCCS files on a regular basis for possible damage. The easiest way to do this is to run the **admin** command with the **−h** flag on all SCCS files or SCCS directories, as follows:

   ```
   $ admin −h s.file1 s.file2 ...
   ```

```
$ admin −h directory1 directory2 ...
```

If the **admin** command finds a file where the computed checksum is not equal to the checksum listed in the SCCS file header, it displays this message:

```
ERROR [s. filename]:
1255−057 The file is damaged. (co6)
```

If a file was damaged, try to edit the file again or read a backup copy. After fixing the file, run the **admin** command with the **−z** flag and the repaired file name:

```
$ admin −z s.file1
```

This operation replaces the old checksum in the SCCS file header with a new checksum based on the current file contents. Other SCCS commands can now process the file.

Files

/usr/bin/admin Contains the SCCS **admin** command.

Related Information

The **delta** command, **ed** command, **get** command, **prs** command, **sccshelp** command, **what** command.

The **sccsfile** file format.

List of SCCS Commands in *AIX General Programming Concepts: Writing and Debugging Programs*.

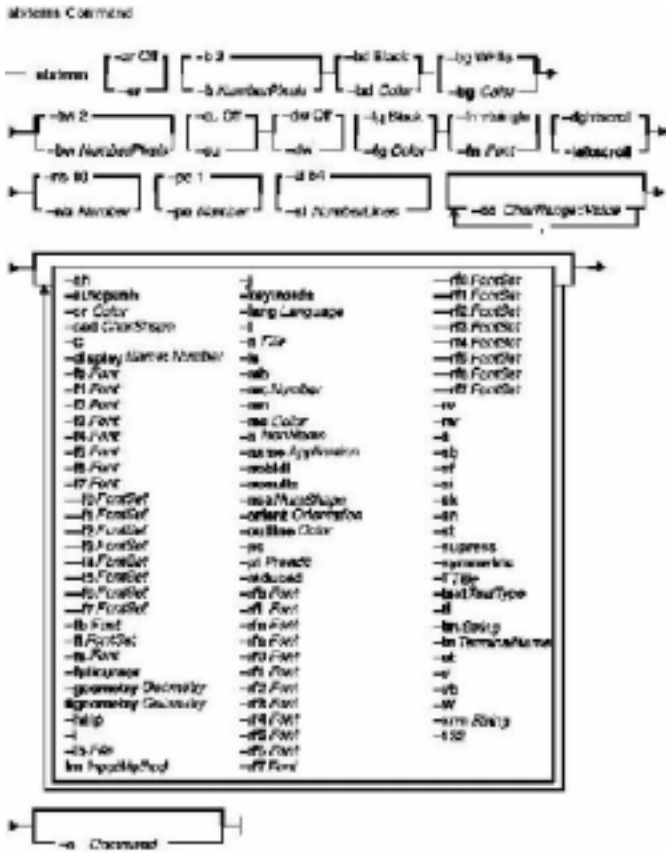
Source Code Control System (SCCS) Overview in *AIX General Programming Concepts: Writing and Debugging Programs*.

aixterm Command

Purpose

Initializes an Enhanced X−Windows terminal emulator.

Syntax



aixterm [**−ah**] [**−ar**] [**−autopush**] [**−b** *NumberPixels*] [**−bd** *Color*] [**−bg** *Color*]
[**−bw** *NumberPixels*] [**−cc** *CharRange***:***Value* [**,**...]] [**−cr** *Color*] [**−csd** *CharShape*]
[**−cu**] [**−C**] [**−display** *Name***:***Number*] [**−dw**] [**−f0** *Font*] [**−f1** *Font*] [**−f2** *Font*]
[**−f3** *Font*] [**−f4** *Font*] [**−f5** *Font*] [**−f6** *Font*] [**−f7** *Font*] [**−−f0** *FontSet*]
[**−−f1** *FontSet*] [**−−f2** *FontSet*] [**−−f3** *FontSet*] [**−−f4***FontSet*] [**−−f5** *FontSet*]
[**−−f6** *FontSet*] [**−−f7** *FontSet*] [**−fb** *Font*] [**−fg** *Color*] [**−fi** *FontSet*] [**−fn** *Font*]
[**−fs** *Font*] [**−fullcursor**] [**−geometry** *Geometry*] [**#geometry** *Geometry*] [**−help**] [**−i**]
[**−ib** *File*] [**−im** *InputMethod*] [**−j**] [**−keywords**] [**−lang** *Language*] [**−l**] [**−leftscroll**]
[**−lf** *File*] [**−ls**] [**−mb**] [**−mc** *Number*] [**−ms** *Color*] [**−mn**] [**−n** *IconName*]
[**−name** *Application*] [**−nb** *Number*] [**−nobidi**] [**−nonulls**] [**−nss** *NumShape*]
[**−orient** *Orientation*] [**−outline** *Color*] [**−po** *Number*] [**−ps**] [**−pt** *Preedit*] [**−reduced**]
[**−rfb** *Font*] [**−rfi** *Font*] [**−rfn** *Font*] [**−rfs** *Font*] [**−rf0** *Font*] [**−rf1** Font]
[**−rf2** Font] [**−rf3** Font] [**−rf4** Font] [**−rf5** Font] [**−rf6**Font] [**−rf7** Font]
[**−−rf0** *FontSet*] [**−−rf1** *FontSet*] [**−−rf2** *FontSet*] [**−−rf3** *FontSet*] [**−−rf4** *FontSet*]
[**−−rf5** *FontSet*] [**−−rf6** *FontSet*] [**−−rf7** *FontSet*] [**−rv**] [**−rw**] [**−s**] [**−sb**] [**−sf**]
[**−si**] [**−sk**] [**−sl** *NumberLines*] [**−sn**] [**−st**] [**−suppress**] [**−symmetric**] [**−T** *Title*]
[**−text** *TextType*] [**−ti**] [**−tm***String*] [**−tn** *TerminalName*] [**−ut**] [**−v**] [**−vb**] [**−W**]
[**−xrm** *String*] [**−132**] [**−e** *Command*]

Description

The **aixterm** command provides a standard terminal type for programs that do not interact directly with Enhanced X−Windows. This command provides an emulation for a VT102 terminal or a high function terminal (HFT). The VT102 mode is activated by the **−v** flag.

The **aixterm** command supports the display for up to 16 colors at a time.

The **aixterm** terminal supports escape sequences that perform terminal functions such as cursor control, moving and deleting lines, and **aixterm** private functions.

Many of the special **aixterm** terminal features (like the scroll bar) can be modified under program control through a set of private **aixterm** command escape sequences. You can also use escape sequences to change the title in the title bar.

There are three different areas in the **aixterm** window:

- Scroll bar
- Status line
- Terminal window.

By default, only the terminal window is initially displayed.

The terminal window is the area provided for terminal emulation. When you create a window, a pseudo terminal is allocated and a command (usually a shell) is started.

The **aixterm** command automatically highlights the window border and the text cursor when the mouse cursor enters the window (selected) and unhighlights them when the mouse cursor leaves the window (unselected). If the window is the focus window, the window is highlighted regardless of the location of the mouse cursor. Any window manager, as in the case of the AIXwindows Window Manager (MWM), can cover the **aixterm** border, and the highlight and border color do not show.

The **WINDOWID** environment variable is set to the resource ID number of the **aixterm** window.

When running in an **aixterm** window, the **TERM** environment variable should be **TERM=aixterm**.

The **TERM** environment variable on your home machine determines what the **TERM** environment variable should be on the remote machine (unless it is overridden by your **.profile**).

When you use the **rlogin**, **tn**, or **rsh** commands to login to a different machine, the **TERM** environment variable should be set to **aixterm**. If this operation does not occur, you can perform the following two command line operations:

1. **TERM=aixterm**
2. **export TERM**

If commands (for example, the **vi** command) do not recognize the term type **aixterm** when you login to another system, perform the following one−time operation on the remote system:

1. **su**
2. **cd/tmp**
3. **mkdir***Xxxxx*
4. **cd***Xxxxx*
5. **ftp***LocalSystemName*
6. **cd /usr/share/lib/terminfo**
7. **get ibm.ti**

8. **quit**
9. **TERMINFO=/tmp/***Xxxxx*
10. **export TERMINFO**
11. **tic ibm.ti**
12. **ls**
13. **ls a**
14. **mkdir /usr/share/lib/terminfo/a**
15. **cp a/aixterm* /usr/share/lib/terminfo/a**
16. **cd /tmp**
17. **rm −r /tmp/***Xxxxx*
18. **exit**
19. On the remote machine, enter the following:
    a. **TERM=aixterm**
    b. **export TERM**

Arabic/Hebrew Support

The **aixterm** command supports bidirectional languages such as Arabic and Hebrew. This command can open a window to be used with Arabic/Hebrew applications. You can create an Arabic/Hebrew window by specifying an Arabic or Hebrew locale (**ar_AA**, **Ar_AA**, **iw_IL**, or **Iw_IL**) with the **−lang** flag or by predefining an Arabic or Hebrew locale from SMIT for the system. You can also use the Web−based System Manager **wsm system** fast path and selecting the **Cultural Environment** icon.

The Arabic/Hebrew window supports bidirectional text display. Thus, English and Arabic or Hebrew text can be displayed on the same line. There are different aspects in the Arabic/Hebrew window:

- Screen Orientation
- Text mode
- Character shaping
- Numeric representation
- Status line

Screen Orientation

The screen orientation in an Arabic/Hebrew window can be either left−to−right or right−to−left. The default orientation is left−to−right unless otherwise specified with a flag or in the **.Xdefaults** file. While the window is active, you can reverse the screen orientation using special key combinations. You can reverse the screen orientation according to your needs.

Text Mode

An Arabic/Hebrew window supports two text modes and their corresponding manipulation:

- Implicit
- Visual

In the implicit text mode, characters are stored in same order that they are entered. The text is transformed into its visual form only when it is displayed. In the visual text mode, characters are stored in the same way that they are displayed on the window.

Character Shaping

The Arabic/Hebrew window represents Arabic and Hebrew texts differently, according to its context. Text is represented in one of the following forms:

- Automatic

- Isolated
- Initial
- Middle
- Final

Arabic/Hebrew can also be shaped according to the passthru mode. For more information on character shaping, see "Arabic Character Shaping" in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

Numeric Representation

Numerics can be represented in Arabic numerals, Hindi numerals, or in passthru mode. In implicit text mode, numerals can also be represented according to their contextual form. Thus, Arabic numbers can be displayed in English text or Hindi numbers can be displayed in Arabic text.

Status Line

The Arabic/Hebrew window can display an optional status line that shows the current status of the window. The status line contains the following values:

Value	Current Setting
E	English language
N	National language
SCR−>	Left−to−right screen orientation
<−SCR	Right−to−left screen orientation
alef	Auto shape mode
blank	Passthru shaping mode
ghain	Displayed in the currently used shaping mode
I	Implicit text mode
V	Visual text mode
U	Context numbers
A	Arabic numbers
H	Hindi numbers
P	Passthru for numbers

Note: Use the implicit text mode (the default text mode) for more efficient data sorting.

Use the following key combinations in an Arabic/Hebrew window to change certain settings.

Key Combination	Purpose
Alt + Enter	Reverses screen direction.
Alt + Right Shift	Enables Arabic/Hebrew keyboard layer.
Alt + Left Shift	Enables English keyboard layer.

For Implicit Mode only:

Alt + Kpd* Adjusts the column heading.

For Visual Mode only:

Alt + Kpd 1 Shapes characters in their initial form.

Alt + Kpd 2 Shapes characters in their isolated form.
Alt + Kpd 3 Shapes characters in their passthru form.
Alt + Kpd 4 Shapes characters automatically (Valid also for Implicit).
Alt + Kpd 7 Shapes characters in their middle form.
Alt + Kpd 8 Shapes characters in their final form.
Shift + Kpd / Toggles the Push Mode (Push/End Push).
Alt + Kpd / Toggles the Autopush function.

For more information on the Autopush function, the Push/End Push function, or other Arabic/Hebrew functions, see the **telnet,tn or tn3270** command.

Using the aixterm Command Data−Stream Support

The following is a list of the escape sequences supported by the **aixterm** command.

Some escape sequences activate and deactivate an alternate screen buffer that is the same size as the display area of the window. This capability allows the contents of the screen to be saved and restored. When the alternate screen is activated, the current screen is saved and replaced with the alternate screen. Saving lines scrolled off of the window is disabled until the original screen is restored.

The following table uses these abbreviations in the right hand column:

Xv
Supported by the **aixterm** command running in VT100 mode.
Xh
Supported by the **aixterm** command running in HFT mode.
H
Found in the HFT data stream in AIX version 3.
V
Found in the VT100 data stream.

Name	Function	Data Stream	Support
	SINGLE−BYTE CONTROLS		
BEL	Bell	0x07	Xv, Xh, H, V
BS	Backspace	0x08	Xv, Xh, H, V
HT	Horizontal tab	0x09	Xv, Xh, H, V
LF	Linefeed	0x0A	Xv, Xh, H, V
VT	Vertical tab	0x0B	Xv, Xh, H, V
FF	Form feed	0x0C	Xv, Xh, H, V
CR	Carriage return	0x0D	Xv, Xh, H, V
SO	Shift out	0x0E	Xv, Xh, H, V

SI	Shift in	0x0F	Xv, Xh, H, V
DC1	Device control 1	0x11	H, V
DC3	Device control 3	0x13	H, V
CAN	Cancel	0x18	H, V
SUB	Substitute (also cancels)	0x1A	H, V
ESC	Escape	0x1B	Xv, Xh, H, V
SS4	Single Shift 4	0x1C	H
SS3	Single Shift 3	0x1D	H
SS2	Single Shift 2	0x1E	H
SS1	Single Shift 1	0x1F	H
cbt	cursor back tab	ESC [Pn Z	Xv, Xh, H
cha	cursor horizontal absolute	ESC [Pn G	Xv, Xh, H
cht	cursor horizontal tab	ESC [Pn I	H
ctc	cursor tab stop control	ESC [Pn W	H
cnl	cursor next line	ESC [Pn E	H
cpl	cursor preceding line	ESC [Pn F	Xv, Xh, H
cpr	cursor position report	ESC [Pl; Pc R	Xv, Xh, H, V
cub	cursor backward	ESC [Pn D	Xv, Xh, H, V
cud	cursor down	ESC [Pn B	Xv, Xh, H, V
cuf	cursor forward	ESC [Pn C	Xv, Xh, H, V
cup	cursor position	ESC [Pl; Pc H	Xv, Xh, H, V
cuu	cursor up	ESC [Pn A	Xv, Xh, H, V
cvt	cursor vertical tab	ESC [Pn Y	H
da1	DEVICE ATTRIBUTES		
	request (host to vt100)	ESC [c	Xv, Xh, V
	request (host to vt100)	ESC [0 c	Xv, Xh,

			V
	response (vt100 to host)	ESC [? 1 ; 2 c	Xv, Xh, V
dch	delete character	ESC [Pn P	Xv, Xh, H
decaln	screen alignment display	ESC # 8	Xv, Xh, V
deckpam	keypad application mode	ESC =	Xv, V
deckpnm	keypad numeric mode	ESC >	Xv, V
decrc	restore cursor & attributes	ESC 8	Xv, Xh, V
decsc	save cursor & attributes	ESC 7	Xv, Xh, V
decstbm	set top & bottom margins	ESC [Pt; Pb r	Xv, Xh, V
dl	delete line	ESC [Pn M	Xv, Xh, H
dsr	device status report	ESC [Ps n	
	0 response from vt100: ready		Xv, Xh, V
	5 command from host: please report status		Xv, Xh, V
	6 command from host: report active position		Xv, Xh, H, V
	13 error report sent from virtual terminal to host		H
dmi	disable manual input	ESC ` (back quote)	H
emi	enable manual input	ESC b	H
ea	erase area	ESC [Ps O	
	0 erase to end of area		Xv, Xh, H
	1 erase from area start		Xv, Xh, H
	2 erase all of area		Xv, Xh, H
ed	erase display	ESC [Ps J	
	0 erase to end of display		Xv, Xh, H, V
	1 erase from display star		Xv, Xh, H, V

	2 erase all of display		Xv, Xh, H, V
ef	erase field−e,s,all	ESC [Ps N	
	0 erase to end of field		Xv, Xh, H
	1 erase from field start		Xv, Xh, H
	2 erase all of field		Xv, Xh, H
el	erase line	ESC [Ps K	
	0 erase to end of line		Xv, Xh, H, V
	1 erase from start of line		Xv, Xh, H, V
	2 erase all of line		Xv, Xh, H, V
ech	erase character	ESC [Pn X	Xv, Xh, H
hts	horizontal tab stop	ESC H	Xv, Xh, H, V
hvp	horizontal and vertical position	ESC [Pl; Pc f	Xv, Xh, H, V
ich	insert character	ESC [Pn @	Xv, Xh, H
il	insert line	ESC [Pn L	Xv, Xh, H
ind	index	ESC D	Xv, Xh, H, V
ls2	lock shift G2	ESC n	Xv
ls3	lock shift G3	ESC o	Xv
nel	next line	ESC E	Xv, Xh, H, V
ksi	keyboard status information	ESC [Ps p	H
pfk	PF key report	ESC [Pn q	Xh, H
rcp	restore cursor position	ESC [u	Xv, Xh, H
ri	reverse index	ESC M	Xv, Xh, H, V
ris	reset to initial state	ESC c	Xv, Xh, H, V

rm	reset mode, ANSI specified modes: See "set mode" following in this column.	ESC [Ps;...;Ps	
	reset mode, other private modes and XTERM private modes: See "set mode" following in this column.	ESC [? Ps;...;Ps l	
	restore mode, other private modes and XTERM private modes: See "set mode" following in this column.	ESC [? P;...;Ps r	
	save mode, other private modes and XTERM private modes: See "set mode" following in this column.	ESC [? Ps;...;Ps s	
sapv	select alternate presentation variant	ESC [Ps1;...Psn]	Xh
	0 set default values for BIDI		
	1 set Arabic numeric shapes		
	2 set Hindi numberic shapes		
	3 set symmetric swapping mode for directional characters		
	5 the following graphic character ispresented in its isolated form (Arabic only)		
	6 the following graphic character is presented in its initial form (Arabic only)		
	7 the following graphic character is presented in its middle form (Arabic only)		
	8 the following graphic character is presented in its final form (Arabic only)		
	13 set Special shaping mode		
	14 set standard shaping mode		
	15 reset symmetric mode		
	18 Passthru (everything)		
	19 Passthru (everything except numbers)		
	20 Contextual numbers (device dependant)		
	21 lock 5, 6, 7, 8		
	22 unlock		
	23 set the nonull mode		
	24 reset the nonull mode		
	Values 5−8 effect only the following character unless used with values 21 or 22.		
scp	save cursor postion	ESC [s	Xv, Xh, H
scs	select character set		
	United Kingdom Set	ESC (A (GO)	Xv, V

		ESC) A (G1)	Xv, V
		ESC * A (G2)	Xv, V
		ESC + A (G3)	Xv, V
	ASCII Set (USASCII)	ESC (B (G0)	Xv, V
		ESC) B (G1)	Xv, V
		ESC * B (G2)	Xv, V
		ESC + B (G3)	Xv, V
	special graphics	ESC (0 (G0)	Xv, V
		ESC) 0 (G1)	Xv, V
		ESC * 0 (G2)	Xv, V
		ESC + 0 (G3)	Xv, V
sd	scroll down	ESC [Pn T	H
sl	scroll left	ESC [Pn Sp @	H
spd	select screen direction	ESC [Ps1;1 S	Xh
	0 turn screen to left−to−right, set to Latin keyboard		
	1 turn screen direction to right−to−left set to National keyboard		
sr	scroll right	ESC [Pn Sp A	H
srs	select reversed string	ESC [Ps[	Xh
	0 end push		
	1 start push		
ss2	single shift G2	ESC N	Xv
ss3	single shift G3	ESC O	Xv
su	scroll up	ESC [Pn S	Xv, Xh, H
sgr	set graphic rendition	ESC [Ps m	
	0 normal		Xv, Xh, H, V
	1 bold		Xv, Xh, H, V
	4 underscore		Xv, Xh, H, V
	5 blink (appears as bold)		Xv, Xh, H, V
	7 reverse		Xv, Xh, H, V

	8 invisible		Xh, H
	10..17 fonts		Xh, H
	30..37 foreground colors		Xh, H
	40..47 background colors		Xh, H
	90..97 foreground colors		Xh, H
	100..107 background colors		Xh, H
sg0a	set GO character set	ESC (<	Xh, H
sg1a	set G1 character set	ESC) <	Xh, H
sm	set mode		
	ANSI specified modes	ESC [Ps;...;Ps h	
	4 IRM insert mode		Xv, Xh, H
	12 SRM send/rec mode		H
	18 TSM tab stop mode		H
	20 LNM linefeed/newline		Xv, Xh, H, V
	Other private modes	ESC [? Ps;...;Ps h	
	1 normal/application cursor		Xv, V
	3 80/132 columns		Xv, Xh, V
	4 smooth/jump scroll		Xv, Xh, V
	5 reverse/normal video		Xv, Xh, V
	6 origin/normal		Xv, Xh, V
	7 on/off autowrap		Xv, Xh, H, V
	8 on/off autorept		Xv, Xh, V
	21 CNM CR−NL		H
	XTERM private modes		
	40 132/80 column mode		Xv, Xh
	41 curses(5) fix		Xv, Xh
	42 hide/show scrollbar		Xv, Xh
	43 on/off save scroll text		Xv, Xh

	44 on/off margin bell		Xv, Xh
	45 on/off reverse wraparound		Xv, Xh
	47 alternate/normal screen buffer		Xv, Xh
	48 reverse/normal status line		Xv, Xh
	49 page/normal scroll mode		Xv, Xh
tbc	tabulation clear	ESC [Ps g (default Ps =0)	
	0 clear horizontal tab stop at active position		Xv, Xh, H, V
	1 vertical tab at line indicated by cursor		H
	2 horizontal tabs on line		H
	3 all horizontal tabs		Xv, Xh, H, V
	4 all vertical tabs		H
VTD	virtual terminal data	ESC [x	Xv, Xh, H
VTL	virtual terminal locator report	ESC [y	Xh, H
VTR	vt raw keyboard input	ESC [w	Xh, H
vts	vertical tab stop	ESC I	H
xes	erase status line	ESC [? E	Xv, Xh
xrs	return from status line	ESC [? F	Xv, Xh
xhs	hide status line	ESC [? H	Xv, Xh
xss	show status line	ESC [? S	Xv, Xh
xgs	go to column of status line	ESC [? Ps T	Xv, Xh
xst	set text parameters	ESC] Ps ; Pt \007	Xv, Xh
	0 change window name and title to Pt		Xv, Xh
	1 sets only the icon name		Xv, Xh
	2 sets only the title name		Xv, Xh
	Everything between ESC−P and ESC\ is ignored. aixterm will work as usual after the ESC\.	ESC−P...ESC\	Xv, Xh

Copy, Paste, and Re−execute Functions

When you create a terminal window, the **aixterm** command allows you to select text and copy it within the same window or other windows by using copy, paste, and re−execute button functions. These text functions are available in HFT and VT102 emulations. The selected text is highlighted while the button is pressed.

The copy, paste, and re−execute button functions perform as follows:

Copy The left button is used to save text into the cut buffer. The **aixterm** command does a text cut, not a box cut. Move the cursor to beginning of the text, hold the button down while moving the cursor to the end of the region, and release the button. The selected text is highlighted and saved in the global cut buffer and made the **PRIMARY** selection when the button is released.

- Double clicking selects by words.
- Triple clicking selects by lines.
- Quadruple clicking goes back to characters, and so on.

Multiple clicking is determined from the time the button is released to the time the button is pressed again, so you can change the selection unit in the middle of a selection.

The right button extends the current selection. If you press this button while moving closer to the right edge of the selection than the left, it extends or contracts the right edge of the selection. If you contract the selection past the left edge of the selection, the **aixterm** command assumes you really meant the left edge, restores the original selection, and extends or contracts the left edge of the selection. Extension starts in the selection unit mode that the last selection or extension was performed in; you can multiple click to cycle through them.

Paste Pressing both buttons at once (or the middle button on a three−button mouse) displays (pastes) the text from the **PRIMARY** selection or from the cut buffer into the terminal window that contains the mouse cursor, inserting it as keyboard input.

Re−execute Pressing the Shift key and the left mouse button takes the text from the cursor (at button release) through the end of the line (including the new line), saves it in the global cut buffer and immediately retypes the line, inserting it as keyboard input. The selected text is highlighted. Moving the mouse cursor off of the initial line cancels the selection. If there is no text beyond the initial cursor point, the **aixterm** command sounds the bell, indicating an error.

By cutting and pasting pieces of text without trailing new lines, you can take text from several places in different windows and form a command to the shell. For example, you can take output from a program and insert it into your favorite editor. Since the cut buffer is globally shared among different applications, you should regard it as a file whose contents you know. The terminal emulator and other text programs should treat it as if it were a text file, that is, the text is delimited by new lines.

Menu Usage

The **aixterm** command has two different menus:

- Options
- Modes

Each menu pops up under the correct combinations of key and button presses. Most menus are divided into two sections that are separated by a horizontal line. The top portion contains various modes that can be altered. A check mark is displayed next to a mode that is currently active. Selecting one of these modes toggles its state. The bottom portion of the menu provides the command entries; selecting one of these performs the indicated function.

The Options menu pops up when the Ctrl key and the left mouse button are pressed simultaneously while the mouse cursor is in a window. The menu contains items that apply to all emulation modes.

The Modes menu sets various modes for each emulation mode. The menu is activated by pressing the Ctrl key and the middle mouse button at the same time, while the mouse cursor is in the window. In the command section of this menu, the soft reset entry resets the scroll regions. This is convenient when a program leaves the scroll regions set incorrectly. The full reset entry clears the screen, resets tabs to every eight columns, and resets the terminal modes (such as wrap and smooth scroll) to their initial states after the **aixterm** command finishes processing the command−line options. When the Auto Linefeed option is turned on, a carriage return is added when a carriage return, vertical tab, or form feed is received. The shells

generally do this for the linefeed, but not for the vertical tab or form feed.

Scroll Bar

The **aixterm** command supports an optional scroll bar composed of a scroll button that displays at the top of the scroll bar and a scroll region that displays at the bottom. The scroll bar is hidden until you request it to display.

The scroll region displays the position and amount of text currently showing in the window (highlighted) relative to the amount of text actually saved in the scrolling buffer. As more text is saved in the scrolling buffer (up to the maximum), the size of the highlighted area decreases.

The scroll button causes the window to scroll up and down within the saved text. Clicking the right button moves the window position up (the text scrolls downward); clicking the left button moves the window position down (the text scrolls upward). The amount of scrolling is modified by the Shift and Ctrl keys. If neither key is pressed, the window scrolls a single line at a time. Pressing the Shift key causes the text to scroll a full window at a time, minus one line. Pressing the Ctrl key causes the text to be positioned at the extreme top or bottom of the file.

Character Classes

Clicking the left mouse button (the copy function) twice in rapid succession causes all characters of the same class (that is, letters, white space, punctuation, and so on) to be selected. Because people have different preferences for what should be selected (for example, if file names be selected as a whole or only the separate subnames), you can override the default mapping by using the **charClass** (class **CharClass**) resource.

The **charClass** resource is a list of *CharRange***:***Value* pairs where the range is either a single number or a low−to−high number in the range of 0 to 127, corresponding to the ASCII code for the character or characters to be set. The value is arbitrary, although the default table uses the character number of the first character occurring in the set.

The default table is as follows:

```
static int charClass[128] = {

/* NUL  SOH  STX  ETX  EOT  ENQ  ACK  BEL */

   32,   1,   1,   1,   1,   1,   1,   1,

/*  BS   HT   NL   VT   NP   CR   SO   SI */

    1,  32,   1,   1,   1,   1,   1,   1,

/* DLE  DC1  DC2  DC3  DC4  NAK  SYN  ETB */

    1,   1,   1,   1,   1,   1,   1,   1,

/* CAN   EM  SUB  ESC   FS   GS   RS   US */

    1,   1,   1,   1,   1,   1,   1,   1,

/*  SP   !    "    #    $    %    &    ' */
```

```
   32,  33,  34,  35,  36,  37,  38,  39,
/*  (    )    *    +    ,    -    .    / */
   40,  41,  42,  43,  44,  45,  46,  47,
/*  0    1    2    3    4    5    6    7 */
   48,  48,  48,  48,  48,  48,  48,  48,
/*  8    9    :    ;    <    =    >    ? */
   48,  48,  58,  59,  60,  61,  62,  63,
/*  @    A    B    C    D    E    F    G */
   64,  48,  48,  48,  48,  48,  48,  48,
/*  H    I    J    K    L    M    N    O */
   48,  48,  48,  48,  48,  48,  48,  48,
/*  P    Q    R    S    T    U    V    W */
   48,  48,  48,  48,  48,  48,  48,  48,
/*  X    Y    Z    [    \    ]    ^    _ */
   48,  48,  48,  91,  92,  93,  94,  48,
/*  `    a    b    c    d    e    f    g */
   96,  48,  48,  48,  48,  48,  48,  48,
/*  h    i    j    k    l    m    n    o */
   48,  48,  48,  48,  48,  48,  48,  48,
/*  p    q    r    s    t    u    v    w */
   48,  48,  48,  48,  48,  48,  48,  48,
/*  x    y    z    {    |    }    ~   DEL */
   48,  48,  48, 123, 124, 125, 126,   1};
```

For example, the string `"33:48,37:48,45-47:48,64:48"` indicates that the ! (exclamation mark), %

(percent sign), − (dash), . (period), / (slash), and & (ampersand) characters should be treated the same way as characters and numbers. This is very useful for cutting and pasting electronic mailing addresses and UNIX file names.

Key Translations

It is possible to rebind keys (or sequences of keys) to arbitrary strings for input. Changing the translations for events other than key and button events is not expected, and causes unpredictable behavior.

The actions available for key translations are as follows:

insert()	Processes the key in the normal way (that is, inserts the ASCII character code corresponding to the keysym found in the keyboard mapping table into the input stream).
string(*String***)**	Rebinds the key or key sequence to the string value; that is, inserts the string argument into the input stream. Quotation marks are necessary if the string contains white space or nonalphanumeric characters. If the string argument begins with the characters ``0x," it is interpreted as a hex character constant and the corresponding character is sent in the normal way.
keymap(*Name***)**	Takes a single string argument naming a resource to be used to dynamically define a new translation table; the name of the resource is obtained by appending the string Keymap to *Name*. The keymap name **None** restores the original translation table (the very first one; a stack is not maintained). Uppercase and lowercase is significant.
insert−selection(*Name*[,*Name*]...**)**	Retrieves the value of the first (leftmost) named selection that exists and inserts the value into the input stream. The *Name* parameter is the name of any selection, for example, **PRIMARY** or **SECONDARY**. Uppercase and lowercase is significant.

For example, a debugging session might benefit from the following bindings:

```
*aixterm.Translations: #override <Key>F13: keymap(dbx)
*aixterm.dbxKeymap.translations:\
<Key>F14: keymap(None) \n\
<Key>F17: string("next") string(0x0d) \n\
<Key>F18: string("step") string(0x0d) \n\
<Key>F19: string("continue") string(0x0d) \n\
<Key>F20: string("print") insert−selection(PRIMARY)
```

Key and Button Bindings

The key and button bindings for selecting text, pasting text, and activating the menus are controlled by the translation bindings. In addition to the actions listed in the Key Translations section, the following actions are available:

mode−menu()	Posts one of the two mode menus, depending on which button is pressed.
select−start()	Deselects any previously selected text and begins selecting new text.
select−extend()	Continues selecting text from the previous starting position.
start−extend()	Begins extending the selection from the farthest (left or right) edge.
select−end(*Name*[**,***Name*]...**)**	Ends the text selection. The *Name* parameter is the name of a selection

	into which the text is to be copied. The **aixterm** command asserts ownership of all the selections named. Uppercase and lowercase is significant.
ignore()	Quietly discards the key or button event.
bell([*Volume*]**)**	Rings the bell at the specified volume increment above or below the base volume.

The default bindings are:

```
static char defaultTranslations =
"                    <KeyPress>:  insert() \n\
~Shift Ctrl   ~Meta  <Btn1Down>:   mode-menu(options) \n\
~Shift Ctrl   ~Meta  <Btn2Down>:   mode-menu() \n\
~Shift Ctrl   ~Meta  <Btn3Down>:   mode-menu(modes) \n\
~Shift ~Ctrl  ~Meta  <Btn1Down>:   select-start() \n\
~Shift ~Ctrl  ~Meta  <Btn1Motion>: select-extend() \n\
~Shift ~Ctrl  ~Meta  <Btn1Up>:     select-end(PRIMARY)\n\
~Shift ~Ctrl  ~Meta  <Btn2Down>:   ignore() \n\
~Shift ~Ctrl  ~Meta  <Btn2Up>:     insert-selection(PRIMARY)\n\
~Shift ~Ctrl  ~Meta  <Btn3Down>:   start-extend() \n\
~Shift ~Ctrl  ~Meta  <Btn3Motion>: select-extend() \n\
~Shift ~Ctrl  ~Meta  <Btn3Up>:     select-end(PRIMARY)\n\
Shift ~Ctrl   ~Meta  <Btn1Down>:   reexecute() \n\
Shift ~Ctrl   ~Meta  <Btn1Motion>: select-extend() \n\
Shift ~Ctrl   ~Meta  <Btn1Up>:     select-end(PRIMARY)\n\
Shift ~Ctrl   ~Meta  <Btn2Down>:   select-start() \n\
Shift ~Ctrl   ~Meta  <Btn2Motion>: select-extend() \n\
Shift ~Ctrl   ~Meta  <Btn2Up>:     select-end(PRIMARY)\n\
Shift ~Ctrl   ~Meta  <Btn3Down>:   ignore() \n\
Shift ~Ctrl   ~Meta  <Btn3Up>:     insert-selection(PRIMARY)\n\
Shift Ctrl    ~Meta  <BtnDown>:    size(toggle) \n\
Shift Ctrl    ~Meta  <BtnUp>:      ignore() \n\
                     <BtnDown>:    bell(0) \n\
                     <BtnUp>:      bell(0) \n\
";
```

aixterm Command Internationalization (I18N)

To run an aixterm with a different keyboard layout than the X server's (such as a French keyboard layout on a Swiss German X server), run the following commands:

1. Change the X server to a French keyboard:

   ```
   xmodmap /usr/lpp/X11/defaults/xmodmap/Fr_FR/keyboard
   ```

2. Set the locale environment variable to Fr_FR using one of the following:
   - For Korn shells: `export LANG=Fr_FR`
   - For C shells: `setenv LANG Fr_FR`
   - For Bourne shells: `LANG=Fr_FR; export LANG`
3. Start an aixterm terminal emulator:

   ```
   aixterm &
   ```

4. Reset the X server's keyboard file to its original language:

   ```
   xmodmap /usr/lpp/X11/defaults/xmodmap/Gr_SW/keyboard
   ```

The **aixterm** command continues to use the keyboard layout that the X server was using when the aixterm started. It ignores **KeymapNotify** by default.

The **aixterm** command uses the Input Method to convert the X server's keysyms into either printable

characters or nonprintable escape strings such as function keys. The Input Method uses its own keymap files, in **/usr/lib/nls/loc**, to convert X keysyms into code points for the printable characters, and escape strings for nonprintable characters. There is a keymap file for each language and one keymap file for escape sequences. The escape sequences are in **C@outbound.imkeymap**; the source is **C@outbound.imkeymap.src**. The other keymap files begin with the locale name and look like: **locale.imkeymap** and **locale.codeset.imkeymap**. For example:

US English in codeset IBM−850	En_US.IBM−850.imkeymap
US English in codeset ISO8859−1	en_US.ISO8859−1.imkeymap
Turkish in codeset ISO8859−9	tr_TR.ISO8859−9.imkeymap
Japanese in codeset IBM−932	Ja_JP.IBM−932.imkeymap
Japanese in codeset IBM−943	Ja_JP.IBM−943.imkeymap
Japanese in codeset EUC(JP)	ja_JP.IBM−eucJP.imkeymap

The following dependencies apply:

- You can change the locale by entering the following SMIT fast path: `smit mle_sel_menu`, or by using the Web−based System Manager **wsm system** fast path and selecting the **Cultural Environment** icon. You can also change the locale temporarily by modifying the LANG environment variable.
- You can change the system keyboard definition by selecting the following SMIT menu items: System Environments, Manage Language Environment, and Change the Keyboard Map for the Next System Restart, or by using the Web−based System Manager **wsm system** fast path and selecting the **Cultural Environment** icon.
- Codeset depends on the locale (LC_ALL, LANG environment variables).
- Default fonts and font sets depend on the codeset and locale. Using a font that does not match the codeset may produce incorrect output.
- Input Method depends on the locale. The Input Method for the locale should be installed. The Input Method maps Keysyms to a codeset.
- Compose keys (dead keys) depend on the Input Method and X keyboard mapping. An incorrect input method or X keyboard mapping may produce incorrect input.
- Error messages and menu contents depend on the locale and a correct font or fontset. The message catalogs for the locale should be installed. The default messages are English. An incorrect font or fontset can result in garbled menu text and messages.
- Text display depends on the locale and a correct font or fontset. An incorrect font or fontset can result in garbled text. Changing the locale (LC_ALL, LANG environment variables) in an aixterm does not change the codeset that the aixterm displays. If the codeset of the new locale differs from the codeset of **aixterm**, incorrect output (garbled text) may be displayed.
- The X keyboard mapping depends on the system keyboard definition. Xinit sets the X keyboard mapping to match the system keyboard definition. The mapping is changed with **xmodmap**. The X keyboard mapping maps key presses to Keysyms.

Availability of Characters in aixterm

ASCII characters 32 (0x20) to 126 (0x7e) are available in most of the codesets and fonts. Characters (bytes) 0 (0x00) to 31 (0x1f) are treated as control sequences and unprintable characters. Other characters 127 (0x7f) to 255 (0xff) vary with codeset and fonts. Using a font that does not match the codeset the aixterm is started in leads to unpredictable results. For example, box characters (line drawing) are available in **aixterm** vt100 mode with the default vtsingle font. If you use a different font, other characters may be displayed instead. Another example is using a ISO8859−1 font while running in the IBM−850 codeset. Trying to display box characters (line drawing) generates accented characters. Trying to display accented characters generates different accented characters or blanks.

Box, Line Drawing and Special Graphics Characters in aixterm

Older versions of **aixterm** operate in codeset IBM−850 (pc850). This codeset uses box characters for drawing boxes around things for SMIT. The new codesets do not have these box characters. New escape sequences were added to the **terminfo** entry to support drawing these characters no matter what codeset is being used. The older versions of **aixterm** do not support these new escape sequences. When using an older **aixterm** to access a newer **aixterm** (such as a 3.1 **aixterm** and **telnet** to access 3.2 on another system), set the TERM environment variable to `aixterm−old` (or `aixterm−m−old`). These **terminfo** entries assume the IBM−850 codeset.

Key Assignments for Bidirectional Languages

In addition to the above key and button bindings, the following key assignments for bidirectional languages are supported by the **aixterm** command:

scr−rev() Reverses the screen orientation and sets the keyboard layer to the default language of the new orientation.

ltr−lang() Enables the English keyboard layer.

rtl−lang() Enables the Arabic/Hebrew keyboard layer.

col−mod() Enables the column heading adjustment which handles each word as a separate column.

auto−push() Toggles the Autopush function. This function handles mixed left−to−right and right−to−left text. When you enable the Autopush function, reversed segments are automatically initiated and terminated according to the entered character or the selected language layer. Thus, you are relieved of manually invoking the Push function.

chg−push() Toggles the Push mode. This mode causes the cursor to remain in its position and pushes the typed characters in the direction opposed to the field direction.

shp−in() Shapes Arabic characters in their initial forms.

shp−is() Shapes Arabic characters in their isolated forms.

shp−p() Shapes Arabic characters in their passthru forms.

shp−asd() Shapes Arabic characters in their automatic forms.

shp−m() Shapes Arabic characters in their middle forms.

shp−f() Shapes Arabic characters in their final forms.

The BIDI bindings (for Arabic/Hebrew) are:

```
~Shift ~Ctrl Mod1   <Key>Return:         scr−rev() \n\
~Shift ~Ctrl Mod2   <Key>Return:         scr−rev() \n\
~Shift ~Ctrl Mod1   <Key>Shift_L: ltr−lang() \n\
~Shift ~Ctrl Mod2   <Key>Shift_L: ltr−lang() \n\
~Shift ~Ctrl Mod1   <Key>Shift_R: rtl−lang() \n\
~Shift ~Ctrl Mod2   <Key>Shift_R: rtl−lang() \n\
~Shift ~Ctrl Mod1   <Key>KP_Multiply:    col−mod() \n\
~Shift ~Ctrl Mod2   <Key>KP_Multiply:    col−mod() \n\
~Shift ~Ctrl Mod1   <Key>KP_Divide:      auto−push() \n\
~Shift ~Ctrl Mod2   <Key>KP_Divide:      auto−push() \n\
~Shift ~Ctrl ~Meta  <Key>KP_Divide:      chg−push() \n\
~Shift ~Ctrl Mod1   <Key>KP_1:           shp−in() \n\
~Shift ~Ctrl Mod2   <Key>KP_2:           shp−in() \n\
~Shift ~Ctrl Mod1   <Key>KP_1:           shp−is() \n\
~Shift ~Ctrl Mod1   <Key>KP_2:           shp−is() \n\
~Shift ~Ctrl Mod1   <Key>KP_3:           shp−p() \n\
~Shift ~Ctrl Mod2   <Key>KP_3:           shp−p() \n\
~Shift ~Ctrl Mod1   <Key>KP_4:           shp−asd() \n\
~Shift ~Ctrl Mod2   <Key>KP_4:           shp−asd() \n\
~Shift ~Ctrl Mod1   <Key>KP_7:           shp−m() \n\
~Shift ~Ctrl Mod2   <Key>KP_7:           shp−m() \n\
```

```
~Shift ~Ctrl Mod1   <Key>KP_8:           shp-f() \n\
~Shift ~Ctrl Mod2   <Key>KP_8:           shp-f() \n\
```

You can change these values in the **.Xdefaults** file. For example, if you want to use Ctrl+Shift to change language layer, you can add the following line in the **.Xdefaults** file:

```
Translations:   Ctrl<Key>Shift_R: rtl-lang() \n\
                Ctrl<Key>Shift_L: ltr-lang()
```

Flags

A flag takes on the opposite value if the − (minus sign) is changed to a + (plus sign). The following options override those set in the **.Xdefaults** file:

−ah Highlights the cursor at all times.

−ar Turns on the autoraise mode of **aixterm**, which automatically raises the window (after a delay determined by the **.Xdefaults** keyword **autoRaiseDelay**) when the mouse cursor enters the window. The default is off.

This flag can be turned on and off from the Options menu.

− autopush Enables the Autopush function for the visual text type.

−b *NumberPixels* Specifies the width in pixels of an inner border. The inner border is the distance between the outer edge of the characters and the window border. The default is 2.

−bd *Color* Specifies the color of the highlighted border on color displays. The default is black.

−bg *Color* Specifies the color of the window background on color displays. The default is white.

−bw *NumberPixels* Specifies the width of the window border in pixels. The default is 2 pixels. Some window managers can override this option.

−C Intercepts console messages.

−cc*CharRange***:***Value***,...** Changes the types of characters that are part of a word. For example, the string `−cc 48−52:3` would make the characters `01234` one word and `56789` a different word. The `:3` defines a word group number 3. By default, numbers are in class 48. The character classes are used by cut and paste.

−cr *Color* Determines the color of the text cursor on color displays. The default is the foreground color.

−csd *CharShape* Specifies the default shape of Arabic text. The *CharShape* variable can be one of the following options:

automatic
Shapes the characters automatically.

passthru
Does not shape the characters. The characters are displayed in the same way that they are entered.

isolated
Displays the characters in their isolated form (valid in visual mode only).

initial
Displays the characters in their initial form (valid in visual mode only).

middle
Displays the characters in their middle form (valid in visual mode only).

final
Displays the characters in their final form (valid in visual mode only).

−cu
Causes certain curses applications to display leading tabs correctly. The default is off.

This flag can be turned on and off from the Modes menu.

−display *Name*:*Number*
Identifies the host name and X Server display number where the **aixterm** command is to run. By default, **aixterm** gets the host name and display number from the **DISPLAY** environment variable.

−dw
Causes the mouse cursor to move (warp) automatically to the center of the **aixterm** window when the **aixterm** icon window is deiconified. The default is off.

−e *Command*
Specifies a command to be executed in the window. This flag runs the command; it does not start a shell. If this flag is used, the command and its arguments (if any) must be displayed last on the **aixterm** command line.

When the command exits, the **aixterm** command exits.

−f0*Font*
Specifies the name of the default font on the command line. Also specifies the name of the font placed in position 0 in the font table. This flag is similar to the **−fn** flag. For example, to specify a default font on the command line, enter the following:

```
aixterm −f0 rom11 −tn aixterm−old
```

−f1*Font*
Specifies the name of the font placed in position 1 in the font table. This flag is similar to the **−fb** flag.

−f2*Font*
Specifies the name of the font placed in position 2 of the font table. This flag is similar to the **−fi** flag.

−f3*Font*
Specifies the name of the font placed in position 3 of the font table.

−f4*Font*
Specifies the name of the font placed in position 4 of the font table.

−f5*Font*
Specifies the name of the font placed in position 5 of the font table.

−f6*Font*
Specifies the name of the font placed in position 6 of the font table.

−f7*Font*
Specifies the name of the font for position 7 in the font table.

−−f0*FontSet*
Specifies the name of the font set for position 0 in the font table. This flag is similar to the **−fn** flag.

−−f1*FontSet*
Specifies the name of the font set for position 1 in the font table. This flag is similar to the **−fb** flag.

−−f2*FontSet*
Specifies the name of the font set for position 2 in the font table. This flag is similar to the **−fi** flag.

−−f3*FontSet*
Specifies the name of the font set for position 3 in the font table.

−−f4*FontSet*
Specifies the name of the font set for position 4 in the font

table.

−−f5*FontSet* — Specifies the name of the font set for position 5 in the font table.

−−f6*FontSet* — Specifies the name of the font set for position 6 in the font table.

−−f7*FontSet* — Specifies the name of the font set for position 7 in the font table.

−fb *Font* — Specifies the name of the bold font. This font must be the same height and width as the normal font.

−fi *FontSet* — Specifies the name of the italic font set.

−fg *Color* — Determines the foreground color of the text on color displays. The default is black.

−fn *Font* — Specifies the name of a normal full−text font set. Any fixed−width font set can be used. In HFT emulation, the default is **Rom14.500** for a large display or **Rom10.500** for a small display. In VT102 emulation, the default is **vtsingle**. To specify a font set in the resource file, use **aixterm.Fontset***FontSet*.

−fs *Font* — Specifies the name of the special graphics font.

−fullcursor — Uses a full block cursor instead of the default underscore cursor.

−geometry *Geometry* — Specifies the location and dimensions of a window. The default is **80x25+0+0**. Some window managers (such as the **mwm** command) can override these defaults.

#geometry*Geometry* — Specifies the location of an icon window. If specified, width and height are ignored. Width and height are taken from the size of the bitmap and the length of the title. The window manager can override the location of the icon.

> **Note:** When you use one of these values as part of an **sh** (shell) command, enclose the value in "" (double quotation marks). Normally, #(the pound sign) indicates a comment in a shell script.

−help — Lists the available option flags.

−i — Displays the icon window rather than the normal window when the window is opened. The default is false.

> **Note:** This flag does not work unless the window manager has started.

−ib *File* — Specifies name of the bitmap file to read for use as the icon bitmap file instead of the default bitmap file. You can access a **/usr/include/X11/bitmaps** file from an AIX shell to see a sample bitmap file.

−im*InputMethod* — Specifies a modifier string that identifies the input method to be used by the **aixterm** command.

−j — Causes the **aixterm** command to move multiple lines up at once (jump scroll) if many lines are queued for display. The default is false.

This flag can be turned on and off from the Modes menu.

−keywords — Lists the **.Xdefaults** keywords.

−lang *Language* — Specifies the language to be used under the **aixterm** command. The language should follow the format

for the locale, as used by the **setlocale** function.

−l Causes the **aixterm** command to append output from the window to the end of the **logfile** file. The default is false.

This flag can be turned on and off from the Options menu.

This does not override **LogInhibit** in the **.Xdefaults** file.

−leftscroll Places the scroll bar on the left when it is displayed. The default is on the right side of the text window.

−lf *File* Specifies the file where the output is saved, instead of the default **AixtermLog.***XXXXXX* file, where *XXXXXX* is the process ID of the **aixterm** command. The file is created in the directory where the **aixterm** command is started, or in the home directory for a login **aixterm** command. If the file name begins with a | (pipe symbol), the rest of the string is interpreted as a command to be executed by the shell, and a pipe is opened to the process.

This flag must be used in conjunction with the **−l** flag to work effectively.

−ls Causes the shell run under the **aixterm** command to be a login shell. The user's **.login** or **.profile** file is read, and the initial directory is usually the home directory. The default is false.

−mb Turns on the right margin bell. The default is false.

This flag can be turned on and off from the Modes menu.

−mc *Number* Determines the multiple−click time. This is used by the cut and paste button functions.

−mn Ignores the **XMappingNotify** event. The **−mn** flag is the default.

−ms *Color* Determines the color of the mouse cursor on color displays. The default is the foreground color.

−n *IconName* Specifies the icon name for use by the **aixterm** command.

−name*Application* Specifies the application name to use for the **.Xdefaults** file.

−nb *Number* Specifies the right margin distance at which the margin bell rings. The default is 10 spaces from the right edge of the window.

−nobidi Disables the Arabic/Hebrew functions such as screen reverse, while maintaining an Arabic/Hebrew locale.

− nonulls Enables a Nonulls mode in which nulls within a line are replaced by spaces.

−nss*NumShape* Specifies the default shape of numerals. The *NumShape* variable can be one of the following options:

bilingual
Displays numerals according to the surrounding text. For example, Arabic numerals are displayed within Arabic text and English numerals within English text.

hindi
Displays numerals in Hindi.

arabic
Displays numerals in Arabic.

passthru

Displays numerals the same way they are entered.

− orient*Orientation*
Specifies the default screen orientation. The orientation can be one of the following options:
LTR
Left−to−right screen orientation
RTL
Right−to−left screen orientation

−outline*Color*
Determines the color of the outline attribute (Keisen) on color displays. The default is the foreground color.

The outline attribute for a character is similar to other character attributes such as bold or reverse video. The outline attribute is displayed as a box drawn to enclose a character or group of characters.

−po *Number*
Specifies the number of lines from the previous screen that display on the screen when the window scrolls one page. The default is 1 line.

−ps
Turns on the page scroll mode.

After a page of lines is displayed, the **aixterm** command stops displaying new lines and the text cursor is no longer displayed. Pressing the Enter key displays one new line. Pressing the Spacebar key or a character key displays a new page. The default is false.

−pt*Preedit*
Specifies the pre−edit type for text composing. The possible pre−edit types are:
over
Places the pre−edit window over the spot of character composition.
off
Places the pre−edit window off the spot of character composition in the status area.
root
Composes character outside of the current window tree.
none
Specifies that the input method has no pre−edit area.

−reduced
Causes the **aixterm** command to begin in reduced mode.

−rfb *Font*
Specifies the name of the reduced bold font. This font must be the same width and height as the reduced normal font.

−rfi *Font*
Specifies the name of the reduced italic font. This font must be the same width and height as the reduced normal font.

−rfn *Font*
Specifies the name of the reduced normal font.

−rfs *Font*
Specifies the name of the reduced special graphics font.

−rf0 *Font*
Specifies the name of the reduced font placed in position 0 in the font table. This flag is similar to the **−rfn** flag.

−rf1 *Font*
Specifies the name of the reduced font placed in position 1 in the font table. This flag is similar to the **−rfb** flag.

−rf2 *Font*
Specifies the name of the reduced font placed in position 2 in the font table. This flag is similar to the **−rfi** flag.

−rf3 *Font*
Specifies the name of the reduced font placed in position 3 in the font table.

−rf4 *Font* — Specifies the name of the reduced font placed in position 4 in the font table.

−rf5 *Font* — Specifies the name of the reduced font placed in position 5 in the font table.

−rf6 *Font* — Specifies the name of the reduced font placed in position 6 in the font table.

−rf7 *Font* — Specifies the name of the reduced font placed in position 7 in the font table.

−−rf0 *FontSet* — Specifies the name of the reduced fontset placed in position 0 in the font table. This flag is similar to the **−rfn** flag.

−−rf1 *FontSet* — Specifies the name of the reduced fontset placed in position 1 in the font table. This flag is similar to the **−rfb** flag.

−−rf2 *FontSet* — Specifies the name of the reduced fontset placed in position 2 in the font table. This flag is similar to the **−rfi** flag.

−−rf3 *FontSet* — Specifies the name of the reduced fontset placed in position 3 in the font table.

−−rf4 *FontSet* — Specifies the name of the reduced fontset placed in position 4 in the font table.

−−rf5 *FontSet* — Specifies the name of the reduced fontset placed in position 5 in the font table.

−−rf6 *FontSet* — Specifies the name of the reduced fontset placed in position 6 in the font table.

−−rf7 *FontSet* — Specifies the name of the reduced fontset placed in position 7 in the font table.

−rv — Reverses the foreground and background colors. This becomes the normal video mode.

This flag can be turned on and off from the Modes menu.

−rw — Turns on the reverse−wraparound mode. The default is false.

This mode allows the cursor to wraparound from the leftmost column to the rightmost column of the previous line. This can be useful in the shell to allow erasing characters backwards across the previous line.

This flag can be turned on and off from the Modes menu.

−s — Turns off synchronous scrolling on the display. The default is true.

When this flag is specified, the **aixterm** command no longer attempts to keep the screen current while scrolling and can run faster when network latencies are very high.

−sb — Causes the scroll bar to display. This flag can be turned on and off from the Modes menu. The default is off.

−sf — Generates the Sun function keycodes for programmed−function (PF) keys in VT102 mode.

−si — Specifies that while using the scroll bar to review previous lines of text, the window is normally repositioned automatically at the bottom of the scroll region before output to the screen is processed. The default is true.

This flag disables window repositioning on output.

−sk
Causes the window to be repositioned automatically in the normal position at the bottom of the scroll region when a key is pressed. The default is false.

This flag is intended for use with the scroll bar to review previous lines of text.

Pressing a key also creates output, which is affected by the **−si** flag.

This flag can be turned on and off from the Scrollbar menu.

−sl *NumberLines*
Specifies the maximum number of lines to save that scroll off of the top of the window. The default is 64.

−sn
Displays the status line to be displayed in normal video (the status line is still enclosed in a box). By default, the status line is displayed in reverse−video relative to the rest of the window. This flag can be turned on and off from the Modes menu.

−st
Displays the status line on startup. The default is false.

−suppress
Specifies that the preediting function in the input method **IMIoctl** call is suppressed.

− symmetric
Enables the Symmetric Swapping mode for handling bidirectional character pairs such as <> and ().

−T *Title*
Sets the title bar name, but not the icon name. If the **−n** option is not specified, or the icon name is not a specified keyword in the **.Xdefaults** file, the title is used as the icon name.

−text*TextType*
Specifies the type of data stream. The *TextType* variable can be one of the following options:

− implicit
Characters are stored in key stroke order.

− visual
Characters are stored the same way that they are displayed. You can use the Autopush mode or Push mode with different shape types.

−ti
Displays the title to the right of the bitmap in the icon window. By default, the title is displayed under the bitmap (if the window manager allows it).

−tm *String*
Specifies a series of terminal setting keywords followed by the characters that should be bound to those functions. Allowable keywords include: **intr**, **quit**, **erase**, **kill**, **eof**, **eol**, **start**, **stop**, **susp**, **dsusp**, **rprnt**, **flush**, **weras**, and **lnext**.

−tn *TerminalName*
Specifies the terminal environment variable. Use the **−tn** flag to change the terminal environment variable only. The terminal environment variable should not be changed to match the terminal in which the X Server is running. The **aixterm** command has no direct access to the terminal where the X Server is running.

−ut
Disables the addition of the login ID to **/etc/utmp**.

−v
Enables VT102 emulation. By default, HFT is emulated.

Note: The keyboard map is needed for this mode.

−vb
Enables the visual bell mode. The visual bell flashes the

	window on receipt of the Ctrl−G key combination instead of ringing the bell. The default is false.
−W	Causes the mouse cursor to move (warp) to the middle of the **aixterm** window when the window is created. The default is false.
−xrm*String*	Sets the resource string. For example, `aixterm.foreground: blue`
−132	Causes the sm/rm escape sequences to be recognized and the **aixterm** window to be resized as specified. Normally, the sm/rm escape sequences that switch between the 80−column and 132−column modes are ignored. The default is false. This flag can be turned on and off from the Modes menu.

.Xdefaults Keywords

Use the following keywords to set the defaults for the **aixterm** command.

alwaysHighlight	If true, always highlights the cursor, even when the mouse pointer is outside the window.
autoRaise	If true, raises the **aixterm** window automatically (after a delay of **autoRaiseDelay**) when the mouse cursor enters the window. The default is false. Window managers can override this option.
autoRaiseDelay	If **autoRaise** is true, specifies the number of seconds to delay before automatically raising a window. The default is 2 seconds. Window managers can override this option.
background	Specifies the color of the window background on color displays. The default is a white background.
boldFontSet	Specifies the name of a bold font. This font must have the same height and width as the normal sized font.
borderColor	Specifies the color of the window border. Window managers can override this option.
borderWidth	Specifies the width of the window border in pixels. The default is 2 pixels.
c132	If true, specifies that the sm/rm escape sequences to resize the **aixterm** window between 80 and 132 columns be recognized. The default is false.
charClass	Specifies the character class.
charShape	If set to automatic, the characters are shaped automatically. If set to passthru, the characters do not exert any shaping. If set to isolated, the characters are displayed in isolated shape. If set to initial, the characters are displayed in initial shape. If set to final, the characters are displayed in final shape.
console	If set to true, the **aixterm** command intercepts console messages. The default is false.
curses	If true, causes certain curses applications to display leading tabs correctly. The default is false.
cursorColor	Specifies the color of the text cursor on color displays. The default is the foreground color.
deiconifyWarp	If true, moves or warps the mouse to the center of the window when replacing the **aixterm** icon window with the **aixterm** window. The default is false.
expandTail	The "seen", "sheen", "sad", "dad" Arabic characters and their tails are displayed as two characters.
fASD	Enables the automatic shaping function.
fAutoPush	Enables the Autopush function.
fEndPush	Enables the End Push function.

fLTR	Enables the LTR screen orientation.
font0	Specifies the name of the font placed in position 0 in the font table. This flag is similar to the **−fn** flag.
font1	Specifies the name of the font placed in position 1 in the font table. This flag is similar to the **−fb** flag.
font2	Specifies the name of the font placed in position 2 of the font table. This flag is similar to the **−fi** flag.
font3	Specifies the name of the font placed in position 3 of the font table.
font4	Specifies the name of the font placed in position 4 of the font table.
font5	Specifies the name of the font placed in position 5 of the font table.
font6	Specifies the name of the font placed in position 6 of the font table.
font7	Specifies the name of the font for position 7 in the font table.
fontSet	Specifies the name of the normal sized text font used in the body of the **aixterm** window.
fontSet0	Specifies the name of the font set for position 0 in the font table. This flag is similar to the **−fn** flag.
fontSet1	Specifies the name of the font set for position 1 in the font table. This flag is similar to the **−fb** flag.
fontSet2	Specifies the name of the font set for position 2 in the font table. This flag is similar to the **−fi** flag.
fontSet3	Specifies the name of the font set for position 3 in the font table.
fontSet4	Specifies the name of the font set for position 4 in the font table.
fontSet5	Specifies the name of the font set for position 5 in the font table.
fontSet6	Specifies the name of the font set for position 6 in the font table.
fontSet7	Specifies the name of the font set for position 7 in the font table.
foreground	Specifies the color for the text displayed inside the body of the window on color displays. The default is black.
fPush	Enables the Push function.
fRTL	Enables the RTL screen orientation.
fScrev	Enables the Screen Reverse function.
fShapeF	Enables the Final Shape function.
fShapeIN	Enables the Initial Shape function.
fShapeIS	Enables the Isolated Shape function.
fShapeM	Enables the Middle Shape function.
fShapeP	Enables the Passthru shape function.
fullCursor	Displays the full cursor. The default is an underscore cursor.
geometry	Specifies the location or dimensions of the window.
iconBitmap	Reads the bitmap file name and uses the resulting bitmap as the icon.
iconGeometry	Specifies the location of the icon window.
iconName	Specifies the icon name.
iconStartup	If true, causes the **aixterm** command to start by displaying an icon window rather than the normal window.
inputMethod	Specifies the input method to be used by the **aixterm** command.
internalBorder	Specifies the number of pixels between the text characters and the window border. The default is 2 pixels.
italicFontSet	Specifies the name of the italic font set.
jumpScroll	If true, enables jump scroll. The default is false.

language	Specifies the language to be used under the **aixterm** command. The language should follow the format for the locale, as used by the **setlocale** function.
logFile	If **logging** is true, specifies the file in which the log is written. The default is **AixtermLog.***XXXXXX*, where *XXXXXX* is a unique ID of the **aixterm** command.
logging	If true, appends all input from the pseudo tty to the logfile. The default is false.
logInhibit	If true, prevents a user or an application program from enabling logging. This overrides any values set for **logging**.
loginShell	If true, indicates that the **aixterm** command should start as a login shell. The default is false.
mappingNotify	If set to true, ignores the **XMappingNotify** event. The default is true.
marginBell	If true, enables the right margin bell. The default is false.
multiClickTime	Specifies the number of milliseconds between button clicks when cutting and pasting. The default is 250 milliseconds.
multiScroll	If true, allows asynchronous scrolling.
nMarginBell	Specifies the distance from the right edge of the window where the margin bell rings. The default is 10 spaces from the right edge of the window.
noNulls	Replaces nulls with spaces within a line.
numShape	If set to bilingual, the numbers are shaped according to context. If set to hindi, the numbers are represented in Arabic. If set to arabic, the numbers are represented in English. If set to passthru, the numbers are represented as they are.
orientation	If set to LTR, left−to−right is set as the default screen orientation. If set to RTL, right−to−left is set as the default screen orientation.
outline	Determines the color of the outline attribute (Keisen) on color displays. The default is the foreground color. The outline attribute for a character is similar to other character attributes such as bold or reverse video. The outline attribute is displayed as a box drawn to enclose a character or group of characters.
pageOverlap	Specifies the number of lines from the previous screen that remain on the screen when the terminal scrolls one page. In page scroll mode, a page is the number of lines in the scrolling region minus the page overlap. The default is 1 line.
pageScroll	If true, enables the page scroll mode. The default is false. After a page of lines displays, **aixterm** stops displaying new lines and the text cursor disappears. Pressing the Enter key displays one new line. Pressing the Spacebar key or a character key displays a new page.
preeditType	Specifies the pre−edit type for text composing. The possible pre−edit types are:
over	Places the pre−edit window over the spot of character composition.
off	Places the pre−edit window off the spot of character composition in the status area.
root	Composes character outside of the current window tree.
none	Specifies that the input method has no pre−edit area.
pointerColor	Specifies the color of the mouse cursor on color displays. The default is the foreground color.
pointerShape	Specifies the shape of the mouse cursor to be used in an **aixterm** window. The default is **XC_xterm**. The cursors are listed in the **/usr/include/X11/cursorfont.h** file.
reducedBoldFontSet	Specifies the name of the reduced fontset placed in position 1 in the font table.
reducedFont0	Specifies the name of the reduced font placed in position 0 in the font table.
reducedFont1	Specifies the name of the reduced font placed in position 1 in the font table.
reducedFont2	Specifies the name of the reduced font placed in position 2 in the font table.

reducedFont3	Specifies the name of the reduced font placed in position 3 in the font table.
reducedFont4	Specifies the name of the reduced font placed in position 4 in the font table.
reducedFont5	Specifies the name of the reduced font placed in position 5 in the font table.
reducedFont6	Specifies the name of the reduced font placed in position 6 in the font table.
reducedFont7	Specifies the name of the reduced font placed in position 7 in the font table.
reducedFontSet	Specifies the name of the reduced fontset placed in position 0 in the font table.
reducedFontSet0	Specifies the name of the reduced fontset placed in position 0 in the font table.
reducedFontSet1	Specifies the name of the reduced fontset placed in position 1 in the font table.
reducedFontSet2	Specifies the name of the reduced fontset placed in position 2 in the font table.
reducedFontSet3	Specifies the name of the reduced fontset placed in position 3 in the font table.
reducedFontSet4	Specifies the name of the reduced fontset placed in position 4 in the font table.
reducedFontSet5	Specifies the name of the reduced fontset placed in position 5 in the font table.
reducedFontSet6	Specifies the name of the reduced fontset placed in position 6 in the font table.
reducedFontSet7	Specifies the name of the reduced fontset placed in position 7 in the font table.
reducedItalicFontSet	Specifies the name of the reduced fontset placed in position 2 in the font table.
reducedSpecialFont	Specifies the name of the reduced special graphics font.
reducedStartup	Causes the **aixterm** command to begin in reduced mode.
reverseVideo	If true, reverses the foreground and background color. The default is false.
reverseWrap	If true, sets reverse−wraparound mode, which allows the cursor to wrap from the leftmost column to the rightmost column of the previous line. The default is false.
rtArrow	The Right Arrow key is handled as a movement key.
saveLines	Specifies the maximum number of lines to save when lines scroll off the top of a window. The default is 64 lines.
scrollBar	If true, displays the scroll bar during startup.
scrollInput	Specifies whether output to the terminal automatically causes the scroll bar to go to the bottom of the scrolling region. The default is true.
scrollKey	If true, repositions the window at the bottom of the scroll region (normal position) when a key is pressed while using the scroll bar to review previous lines of text. The default is false. Pressing a key also creates input, which is affected by the **scrollInput** keyword.
scrollPosition	If left, positions the scroll bar to the left side of the screen. The default is right.
signalInhibit	If true, specifies that the signals should not be listed. The default is false.
specialFont	Specifies the name of the special graphics font.
statusLine	If true, displays the status line on startup. The default is false.
statusNormal	If true, displays the status line in normal video (the status line is still enclosed in a box). By default, the status line is in reverse−video relative to the rest of the window.
sunFunctionKeys	If true, the PF keys generate Sun function keycodes when in the VT102 mode. The default is false.
suppress	If true, specifies that the pre−editing function in the input method **IMIoctl** call is suppressed.
symmetric	Enables symmetric character swapping.
termName	Specifies the terminal environment variable, **$TERM**. Use the **termName** keyword to change the terminal environment variable only. The terminal environment variable should not be changed to match the terminal in which the X Server is running. The **aixterm** command has no direct access to the terminal where the X Server is running.
textType	If set to implicit, the data stream type is set to implicit. If set to visual, the data stream type is set to visual.

textUnderIcon	If False, displays the title of the icon window at the right of the bitmap in the icon window. By default, the title is displayed under the bitmap.
title	Specifies the title to show in the title bar. The default is **aixterm**.
ttyModes	Specifies the tty settings.
translations	Specifies the key and button translations to be supplied.
utmpInhibit	If False, adds the login ID to the **/etc/utmp** file. The default is false**.**
visualBell	If true, enables the visual bell mode which flashes the window on receipt of a Ctrl−G key sequence. The default is false.
vt102	If true, enables VT102 mode. The default is emulation.
warp	If true, automatically warps (moves) the mouse cursor to the center of a newly created **aixterm** window. The default is false.

Example

The following example can be used to create an **aixterm**, specifying the size and location of the window, using a font other than the default, and also specifying the foreground color that is used in text. The **aixterm** command then runs a command in that window.

```
aixterm −geometry 20x10+0+175 −fn Bld14.500 −fg DarkTurquoise −e
/tmp/banner_cmd &
```

The **aixterm** command is NOT an X Toolkit based application. Because of this, the **aixterm** command gets resource files as follows:

- **System defaults** from the first of these it finds:

```
$XFILESEARCHPATH %T=app−defaults %N=Xdefaults %L=$LANG
$XFILESEARCHPATH %T=app−defaults %N=Xdefaults %L=
/usr/lpp/X11/defaults/$LANG/Xdefaults
/usr/lpp/X11/defaults/Xdefaults
/usr/lib/X11/$LANG/app−defaults/Xdefaults
/usr/lib/X11/app−defaults/Xdefaults
/usr/lpp/X11/defaults/app−defaults/Xdefaults
```

- **Application system defaults** from the first of these it finds:

```
$XFILESEARCHPATH %T=app−defaults %N=Aixterm %L=$LANG
$XFILESEARCHPATH %T=app−defaults %N=Aixterm %L=
$XFILESEARCHPATH %T=app−defaults %N=aixterm %L=$LANG
$XFILESEARCHPATH %T=app−defaults %N=aixterm %L=
/usr/lpp/X11/defaults/$LANG/Aixterm
/usr/lpp/X11/defaults/Aixterm
/usr/lib/X11/$LANG/app−defaults/Aixterm
/usr/lib/X11/app−defaults/Aixterm
/usr/lib/X11/defaults/app−defaults/Aixterm
/usr/lpp/X11/defaults/$LANG/aixterm
/usr/lpp/X11/defaults/aixterm
/usr/lib/X11/$LANG/app−defaults/aixterm
/usr/lib/X11/app−defaults/aixterm
/usr/lib/X11/defaults/app−defaults/aixterm
```

- **User application defaults** from the first of these it finds:

```
$XUSERFILESEARCHPATH %T=app−defaults %N=Aixterm %L=$LANG
$XUSERFILESEARCHPATH %T=app−defaults %N=Aixterm %L=
$XUSERFILESEARCHPATH %T=app−defaults %N=aixterm %L=$LANG
$XUSERFILESEARCHPATH %T=app−defaults %N=aixterm %L=
$XAPPLRESDIR/$LANG/Aixterm
$XAPPLRESDIR/Aixterm
```

```
$XAPPLRESDIR/$LANG/aixterm
$XAPPLRESDIR/aixterm
$HOME/$LANG/Aixterm
$HOME/Aixterm
$HOME/$LANG/aixterm
```

- **User defaults** from the first of these it finds:

```
dpy->xdefaults          (A.K.A. "RESOURCE_MANAGER" property)
$HOME/$LANG/.Xdefaults
$HOME/.Xdefaults
```

- **Host defaults** from the first of these it finds:

```
$XENVIRONMENT
$HOME/$LANG/.Xdefaults-hostname
$HOME/.Xdefaults-hostname
```

> **Note:** XFILESEARCHPATH and XUSERFILESEARCHPATH support is limited to the %T, %N and %L substitution strings. Also, $LANG is actually whatever the result of the setlocale(LC_CTYPE,NULL) call is.

Related Information

telnet, tn, or tn3270 command.

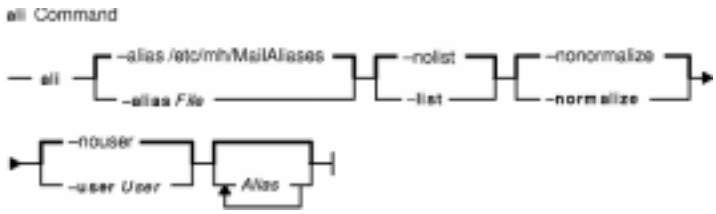
Bidirectionality and Arabic Character Shaping Overview in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

ali Command

Purpose

Lists mail aliases and their addresses.

Syntax



ali [−**alias***File*] [−**list** | −**nolist**] [−**normalize** | −**nonormalize**] [−**user***User* | −**nouser**] [*Alias ...*]

Description

The **ali** command lists mail aliases and their addresses. By default, this command searches the **/etc/mh/MailAliases** file and writes to standard output each alias and its address defined in the file. To specify an alternate mail aliases file, use the −**alias***File* flag.

If you specify the −**user** flag, the **ali** command searches the alias files for the user name and writes to standard output the aliases that contain this user name.

Flags

−**alias** *File*	Specifies the mail alias file to be searched. The default is the **/etc/mh/MailAliases** file.
−**help**	Lists the command syntax, available switches (toggles), and version information. **Note:** For MH, the name of this flag must be fully spelled out.
−**list**	Displays each address on a separate line.
−**nolist**	Displays addresses on as few lines as possible. This flag is the default.
−**nonormalize**	Prevents conversion of local host nicknames to official host names. This is the default.
−**normalize**	Converts local host nicknames to their official host names.
−**nouser**	Lists the address for an alias. This flag is the default.
−**user** *User*	Lists the aliases that contain the specified user. When the −**user** and −**nonormalize** flags are used together, the result may be a partial list of aliases that contain the specified user.

Examples

1. To display a list of all aliases and their addresses in the **/etc/mh/MailAliases** file, enter:

   ```
   ali
   ```

2. To list the names and addresses of the `mygroup` alias, enter:

   ```
   ali   mygroup
   ```

   A list similar to the following is displayed on your local system:

   ```
   mike@mercury   george@helium   vicky@venus
   ```

Files

$HOME/.mh_profile	Contains the MH user profile.
/etc/group	Contains a list of groups.
/etc/passwd	Contains a list of users.
/etc/mh/MailAliases	Contains the default mail alias file.
/usr/bin/ali	Contains the **ali** command.

Related Information

The **comp** command, **dist** command, **forw** command, **repl** command, **send** command, **whom** command.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

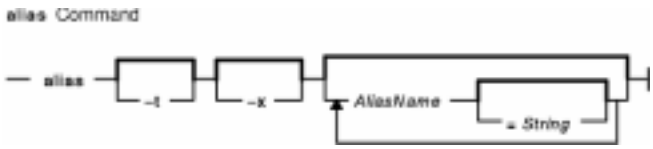
Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

alias Command

Purpose

Defines or displays aliases.

Syntax



alias [**−t**] [**−x**] [*AliasName* [=*String*]] ...

Description

The **alias** command creates or redefines alias definitions or writes existing alias definitions to standard output.

If no flags or parameters are supplied, all existing alias definitions are written to standard output. You can display a specific alias definition by using the *AliasName* parameter.

Create a new alias by using the *AliasName*=*String* parameter pair. When the shell encounters an alias on the command line or in a shell script, it substitutes the definition supplied by the string. The *String* variable can contain any valid shell text. Enclose the value of the *String* variable in single quotes if the string contains spaces. If the *AliasName* parameter is not a valid name, the **alias** command displays an error message.

If you specify the **−t** flag, the shell displays aliases that are *tracked*. A tracked command uses the full path name of the command. A tracked command can become undefined when the value of the **PATH** environment variable is reset, but aliases created with the **−t** flag remain tracked.

If you specify the **−x** flag, the shell displays aliases that are *exported*. An exported alias is active in all shells.

An alias definition affects the current shell environment and the execution environments of any subshells. The alias definition affects neither the parent process of the current shell nor any utility environment invoked by the shell.

Flags

−t Sets or displays all existing tracked aliases. If this flag is used with the *AliasName* parameter, the new alias is tracked and the alias definition includes the full path name obtained by doing a path search. When the value of the **PATH** environment variable is reset, the alias definition becomes undefined but remains tracked.

−x Displays all existing exported alias definitions. If this flag is used with the *AliasName* parameter, the new alias is exported. Exported alias are not defined across separate invocations of the shell. You must put alias definitions in your environment file to have aliases defined for separate shell invocations.

Exit Status

The following exit values are returned:

0 Successful completion.
>0 One of the specified alias name did not have an alias definition, or an error occurred.

Examples

1. To change the **ls** command so that it displays information in columns and annotates the output, enter:

   ```
   alias ls='ls −CF'
   ```

2. To create a command for repeating previous entries in the command history file, enter:

   ```
   alias r='fc −s'
   ```

3. To use 1KB units for the **du** command, enter:

   ```
   alias du=du\ −k
   ```

4. To create a command to display all active processes for user `Dee`, enter:

   ```
   alias psc='ps −ef | grep Dee'
   ```

5. To see the full path name of the **ls** command, enter:

   ```
   alias −t ls
   ```

The screen displays `ls=/usr/bin/ls`.

Files

/usr/bin/ksh Contains the Korn shell **alias** built−in command.
/usr/bin/alias Contains the **alias** command.

Related Information

The **ksh** command.

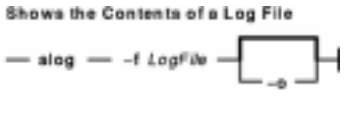
alog Command

Purpose

Creates and maintains fixed−size log files created from standard input.

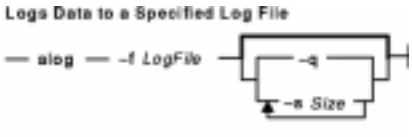
Syntax

To Show the Contents of a Log File



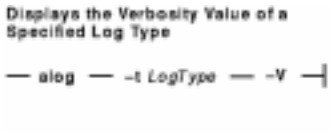
alog −f*LogFile* [**−o**]

To Log Data to a Specified Log File



alog−f*LogFile* | [[**−q**] [**−s***Size*]]

To Display the Verbosity Value of a Specified Log Type

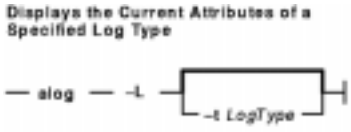


alog −t*LogType* **−V**

To Change the Attributes of a Specified Log Type

alog −C−t*LogType* [**−f** *LogFile*] [**−s***Size*] [**−w***Verbosity*]

To Display the Current Attributes of a Specified Log Type



alog −L [**−t***LogType*]

Description

The **alog** command reads standard input, writes to standard output, and copies the output into a fixed−size file. This file is treated as a circular log. If the file is full, new entries are written over the oldest existing entries.

The **alog** command works with log files that are specified on the command line or with logs that are defined in the alog configuration database. Logs that are defined in the alog configuration database are identified by *LogType*. The **File**, **Size**, and **Verbosity** attributes for each defined *LogType* are stored in the alog configuration database with the *LogType*. You can add a new *LogType* to the alog configuration database using the **odmadd** command. You can change the attributes of *LogType* defined in the alog configuration database using the **alog** command.

Flags

−C
Changes the attributes for a specified *LogType*. Use the **−C** flag with the **−f**, **−s**, and **−w** flags to change the **File**, **Size**, and **Verbosity** attributes for the specified *LogType*. The **−t***LogType* flag is required.

If the **−C** flag is used, the **alog** command does not copy standard input to standard output or to a log file.

When the **−C** flag is used to modify the attributes for the console log type, the console log file is also modified and the console device driver is updated to use the new values. This is a deviation from the normal operation of **alog −C** and is done to accommodate special formatting in the console log file.

Note: You must have root user authority to change **alog** attributes.

−f *LogFile*
Specifies the name of a log file. If the specified *LogFile* does not exist, one is created. If the **alog** command is unable to write to *LogFile*, it writes to **/dev/null**. Use the **−f***LogFile* flag with the **−C** and **−t** flags to change the **File** attribute for a *LogType* defined in the alog configuration database.

−L
Lists the log types currently defined in the alog configuration database. If you use the **−L** flag with the **−t***LogType* flag, the attributes for a specified *LogType* are listed. The current values of the **File**, **Size**, and **Verbosity** attributes are listed as colon separated values:

```
<File>:<Size>:<Verbosity>
```

If the **−L** flag is used, the **alog** command does not copy standard input to standard output or to **File**.

−o
Lists the contents of *LogFile*. Writes the contents of *LogFile* to standard output in sequential order.

−q
Copies standard input to *LogFile* but does not write to standard output.

−s *Size*
Specifies the *LogFile* size limit in bytes. The space for *LogFile* is reserved when it is created. If you create a new *LogFile* and do not specify the **Size** attribute, the minimum *LogFile* size, 4096 bytes, is used. If *LogFile* already exists, its size will be changed. The size you specify is rounded upward to the next integral multiple of 4096 bytes. If you decrease the size of *LogFile*, the oldest entries in the log are deleted if they do not fit within the new size limit. You must have write permission for *LogFile* to change its size.

Use the **−s***Size* flag with the **−C** and the **−t** flags to change the **Size** attribute for *LogType* defined in the alog configuration database. The new **Size** attribute value is used the next time *LogFile* is created.

−t *LogType*
Identifies a log defined in the alog configuration database. The **alog** command gets the log's file name and size from the alog configuration database. If *LogFile* does not exist, one is

created.

If the **alog** command cannot get the information for the specified *LogType* from the alog configuration database or if the **alog** command is unable to write to *LogFile*, it writes to **/dev/null**.

If you specify *LogType* and *LogFile* using the **−f** flag, *LogFile* is used and *LogType* is ignored.

−V Writes the current value of the **Verbosity** attribute for *LogType* that is defined in the alog configuration database to standard output. If you do not specify *LogType*, or the *LogType* you specify is not defined, nothing is written to standard output.

The value output using the **alog** command with the **−t***LogType* and the **−V** flags can be used by a command that is piping its output to the **alog** command to control the verbosity of the data it writes to the pipe.

−w *Verbosity* Changes the **Verbosity** attribute for *LogType* defined in the alog configuration database when used with the **−C** and the **−t** flags.

The **Verbosity** attribute can have a value from 0 to 9. If the value is 0, no information is copied to *LogFile* by the **alog** command. All of the information is still written to standard output. If the value is not 0, all of the information piped to the **alog** command's standard input is copied to *LogFile* and to standard output.

Examples

1. To record the current date and time in a log file named `sample.log`, enter:

   ```
   date | alog −f /tmp/sample.log
   ```

2. To list the contents of `/tmp/sample.log` log file, enter:

   ```
   alog −f /tmp/sample.log −o
   ```

3. To change the size of the log file named `/tmp/sample.log` to 8192 bytes, enter:

   ```
   echo "resizing log file" | alog −f /tmp/sample.log −s 8192
   ```

4. To add a new log type `sample` to the alog configuration database, create the `alog.add` file in the following format:

   ```
   SWservAt:
       attribute="alog_type"
       deflt="sample"
       value="sample"


   SWservAt:
       attribute="sample_logname"
       deflt="/tmp/sample.log"
       value="/tmp/sample.log"


   SWservAt:
       attribute="sample_logsize"
       deflt="4096"
       value="4096"


   SWservAt:
   ```

```
attribute="sample_logverb"
deflt="1"
value="1"
```

After creating the `alog.add` file, enter:

```
odmadd alog.add
```

This adds the `alog.add` file to the `SWservAt` database.

5. To change the name of the log file for the log type `sample` to `/var/sample.log` in the alog configuration database, enter:

```
alog -C -t sample -f /var/sample.log
```

Files

/etc/objrepos/SWservAt Software Service Aids Attributes Object Class

Related Information

The **odmadd** command.

How to Add Objects to an Object Class in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

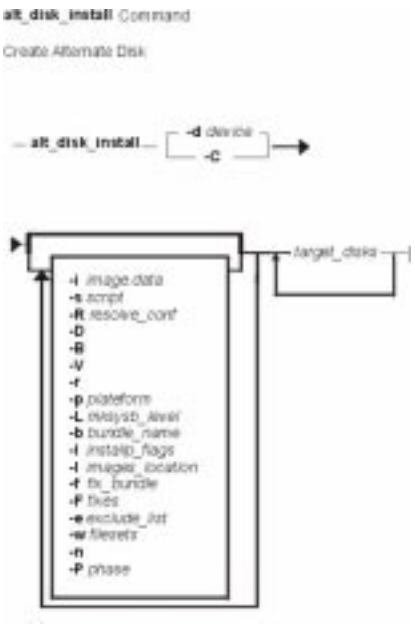
alt_disk_install Command

Purpose

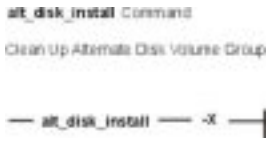
Installs an alternate disk with a mksysb install image or clones the currently running system to an alternate disk.

Syntax

Create Alternate Disk:



alt_disk_install {**−d***device* | **−C**} [**−i***image.data*] [**−s***script*] [**−R***resolv_conf*] [**−D**] [**−B**] [**−V**] [**−r**]
[**−p***platform*] [**−L***mksysb_level*]
[**−b***bundle_name*] [**−I***installp_flags*]
[**−l***images_location*] [**−f***fix_bundle*]
[**−F***fixes*] [**−e***exclude_list*] [**−w***filesets*]
[**−n**] [**−P***phase_option*] *target_disks...*
Clean Up Alternate Disk Volume Group:



alt_disk_install−X

For alt_disk_install 4.3.2 or greater:

Determine Volume Group Boot Disk:

alt_disk_install −q*disk*
Put−to−sleep Volume Group:

alt_disk_install −S
Rename Alternate Disk Volume Group:

alt_disk_install −v *new_volume_group_namedisk*
Wake−up Volume Group:

alt_disk_install −W *disk*
Clean Up Alternate Disk Volume Group:

alt_disk_install−X [*volume_group*]

Description

The **alt_disk_install** command allows users a way to update AIX to the next release or maintenance level, without taking the machine down for an extended period of time. This can be done in two ways, by installing a mksysb image on a separate disk, or by cloning the current system and then applying updates to get to the next maintenance level.

The first function, installing a mksysb, requires a 4.3 mksysb image, a 4.3 mksysb tape, or a 4.3.3 mksysb CD. The **alt_disk_install** command is called with a disk or disks that are not currently in use, and the mksysb is restored to those disks such that, if the user chooses, the next reboot boots the system on a 4.3 system.

Note: If needed, the **bootlist** command can be run after the new disk has been booted, and the bootlist can be changed to boot back to the older version of AIX.

The second function, cloning the running rootvg, allows the user to create a backup copy of the root volume group. This copy could be used as a back up in case the rootvg failed, or it could be modified by installing additional updates. One scenario might be to clone a 4.2.0 system, then install updates to bring the cloned rootvg to 4.2.1.0. This would update the system while it was still running, then rebooting from the new rootvg would bring the level of the running system to 4.2.1. If there was a problem with this level, changing the bootlist back to the 4.2.0 disk and rebooting would bring the system back to 4.2.0. Other scenarios would

include cloning the rootvg and applying individual fixes, rebooting the system and testing those fixes, and rebooting back to the original rootvg if there was a problem.

Currently, you can run the **alt_disk_install** command on 4.1.4.0 and higher systems for both of these functions. The **bos.alt_disk_install.rte** fileset must be installed on the system to execute the **alt_disk_install** command, and the **bos.alt_disk_install.boot_images** fileset must also be installed to perform a mksysb install to an alternate disk.

The mksysb image that is used must be created ahead of time and have all the necessary device and kernel support required for the system that it's going to be installed on. No new device or kernel support can be installed before the system is rebooted from the newly installed disk.

Note: The version release maintainence level of mksysb that you are installing must match the level of the **bos.alt_disk_install.boot_images** fileset.

When cloning the rootvg volume group, a new boot image is created with the **bosboot** command. When installing a mksysb image, a boot image for the level of mksysb and platform type is copied to the boot logical volume for the new alternate rootvg. When the system is rebooted, the **bosboot** command is run in the early stage of boot, and the system is rebooted once again. This is to synchronize the boot image with the mksysb that was just restored. The system then boots in normal mode.

At the end of the install, a volume group, altinst_rootvg, is left on the target disks in the varied off state as a place holder. If varied on, it shows as owning no logical volumes, but it does indeed contain logical volumes, but they have been removed from the ODM because their names now conflict with the names of the logical volumes on the running system. It is recommended that you not vary on the altinst_rootvg volume group, but just leave the definition there as a place holder.

After the system reboots from the new alternate disk, the former rootvg volume group does not show up in a **lspv** listing, unless the **alt_disk_install** version is 4.3.2 or higher.

For **alt_disk_install** 4.3.2 or greater:

> After rebooting from the new alternate disk, the former rootvg volume group shows up in a **lspv** listing as "old_rootvg", and includes all disk(s) in the original rootvg. This former rootvg volume group is set to NOT varyon at reboot, and should ONLY be removed with the −X flag (i.e. **alt_disk_install** −X old_rootvg).

If a return to the original rootvg is necessary, the bootlist command is used to change the bootlist to reboot from the original rootvg.

For **alt_disk_install** 4.3.2 or greater:

> If it is unclear which disk is the boot disk for a specific volume group, the −q flag can be used to determine the boot disk. This can be useful when a volume group is comprised of multiple disks and a change in the bootlist is necessary.

The alternate root file system is mounted as **/alt_inst**, so other file systems would have that prefix (**/alt_inst/usr**, **/alt_inst/var**). This is how they should be accessed if using a customization script.

> **Attention:** If you have created an alternate rootvg with **alt_disk_install**, but no longer wish to use it, or want to run **alt_disk_install** commands, do not run exportvg on altinst_rootvg.
>
> Simply run the **alt_disk_install −X** command to remove the altinst_rootvg definition from the ODM database. The reason you cannot run the **exportvg** command (or the **reducevg** command) is that the logical volume names and file systems now have the real names, and **exportvg** removes the stanza's for the real file system from **/etc/filesystems** for

the real rootvg.

If **exportvg** is run by accident, be sure to recreate the **/etc/filesystems** file before rebooting the system. The system will not reboot without a correct **/etc/filesystems** file.

This function is also available with the Network Installation Management (NIM). See the NIM Guide for more information.

The AIX Version 4.3.1 and greater version of **alt_disk_install** can be executed in phases. The install is divided into three phases, and the default is to perform all three phases.

Phase 1 Creates the **altinst_rootvg** volume group, the **alt_** "logical volumes", the **/alt_inst** file systems, and restores the mksysb or rootvg data.

Phase 2 Runs any specified customization script, installs updates, new filesets, fixes or bundles (cloning only), copies a **resolv.conf** file if specified, and copies files over to remain a NIM client if specified.

Phase 3 Unmounts the **/alt_inst** file systems, renames the file systems and logical volumes, removes the **alt_ logical** volumes, names ODM and varies off the altinst_rootvg. It sets the bootlist and reboots if specified.

You can run each phase separately, run Phases 1 and 2 together, or run Phases 2 and 3 together. Phase 2 can be run multiple times before Phase 3 is run.

You must run Phase 3 to get a volume group that is a usable rootvg. Running Phase 1 and 2 leaved the /alt_inst file systems mounted.

If you have run Phase 1 and or Phase 2, and want to start over (remove the altinst_rootvg), run the **alt_disk_install−x** command to clean up.

For **alt_disk_install** 4.3.2 or greater:

If data access is necessary between the original rootvg and the new alternate disk, a volume group "wake−up" can be accomplished, using the −W flag, on the non−booted volume group. The "wake−up" puts the volume group in a post **alt_disk_install** phase 1 state (i.e. the /alt_inst file systems will be mounted).

Note: The volume group that experiences the "wake−up" will be renamed "altinst_rootvg".

Limitation

The running system's version of AIX must be greater than or equal to the AIX version of the volume group that undergoes the "wake−up". This may mean that it's necessary to boot from the "altinst_rootvg" and "wake−up" the "old_rootvg".

For example: An alternate disk is created from an **alt_disk_install** 4.3.3 mksysb, on a 4.1.5 running system. To access data between the two volume groups, it is necessary to boot from the 4.3.3 alternate disk and "wake−up" the 4.1.5 "old_rootvg" volume group.

This limitation is caused by a jfs log entry incompatibility. It is possible to "wake−up" a volume group that contains a greater AIX version, but the volume group could not have ever been the system rootvg. If so, the volume group would have made jfs log entries that could not be interpreted by an older AIX version rootvg, when the volume group was experiencing a "wake−up". JFS log entries are usually present for file systems

that were not unmounted before a reboot, for example, /,/usr.

The **alt_disk_install** command will not allow a "wake−up" to occur on a volume group with a greater AIX version, unless the FORCE environment variable is set to "yes".

Attention: If a FORCE "wake−up" is attempted on a volume group that contains a greater AIX version then the running OS, **AND** the "waking" volume group has been a system rootvg, errors will occur.

When data access is no longer needed, the volume group can be put to sleep, using the −S flag.

Note: The volume group that has experienced a "wake−up" MUST be "put−to−sleep" before it can be booted and used as the rootvg.

Flags

−B Would specify not running bootlist after the mksysb or clone. If set, then **−r** flag cannot be used.

−C Clone rootvg.

Note: **−d** and **−C** are mutually exclusive.

−d*device* The value for *device* can be:
`tape device − for example, /dev/rmt0`

OR

`path name of mksysb image in a file system.`

Note: **−d** and **−C** are mutually exclusive.

−D Turn on debug (set **−x** output).

−i*image.data* Optional image.data file to use instead of default image.data from mksysb image or image.data created from rootvg. The image.date file name must be a full pathname, for example, `/tmp/my_image.data`.

For **alt_disk_install** 4.3.2 or greater:

> If certain logical volumes need to be placed on a specific target disk, this should be annotated in the logical volume LV_SOURCE_DISK_LIST field of the user specified image.data file.

−p*platform* This is a platform to use to create the name of the disk boot image, which may be supplied by a vendor that wanted to support this function. The default value would be the platform of the running system obtained with the **bootinfo −T** command on AIX Version 4.1 or the **bootinfo −p** command in AIX Version 4.2. This flag is only valid for mksysb installs (**−d** flag).

−P*phase* The *phase* to execute during this invocation of **alt_disk_install**. Valid values are: 1, 2, 3, 12, 23, or all.

- 12 − performs phases 1 and 2.
- 23 − performs phases 2 and 3.
- all − performs all three phases

−r Would specify to reboot from the new disk when the **alt_disk_install** command is complete.

−R*resolv_conf* The **resolv.conf** file to replace the existing one after the mksysb has been restored or the rootvg has been cloned. You must use a full pathname for *resolv_conf*.

−s*script* Optional customization script to run at the end of the mksysb install or the rootvg clone. This file must be executable. This script is called on the running system before the /alt_inst file systems are unmounted, so files can be copied from the running system to the

/alt_inst file systems before the reboot. This is the only opportunity to copy or modify files in the alternate file system because the logical volume names will be changed to match rootvg's, and they will not be accessible until the system is rebooted with the new alternate rootvg, or a "wake−up" is performed on the altinst_rootvg. You must use a full pathname for *script*.

−V Turn on verbose output. This shows the files that are being backed up for rootvg clones. This flag shows files that are restored for mksysb alt_disk_installs.

−L*mksysb_level* This level will be combined with the platform type to create the boot image name to use (for example, rspc_4.3.0_boot). This must be in the form V.R.M. The default will be AIX Version 4.3. The mksysb image will be checked against this level to verify that they are the same.

−n Remain NIM client. The **/.rhosts** and **/etc/niminfo** files are copied to the alternate rootvg's file system.

−X To remove the altinst_rootvg volume group definition from the ODM database. This returns the **lspv** listing for the volume group to "None". This will not remove actual data from the volume group. Therefore, you can still reboot from that volume group, if you reset your bootlist.

For **alt_disk_install** 4.3.2 or greater, the flag allows for specified volume group name ODM database definition removal, for example, −X old_rootvg.

The following flags are only valid for use when cloning the rootvg (**−C**).

−b*bundle_name* Pathname of optional file with a list of packages or filesets that will be installed after a rootvg clone. The **−l** flag must be used with this option.

−e*exclude_list* Optional exclude.list to use when cloning rootvg. The rules for exclusion follow the pattern matching rules of the **grep** command. The *exclude_list* must be a full pathname.

Note: If you want to exclude certain files from the backup, create the **/etc/exclude.rootvg** file, with an ASCII editor, and enter the patterns of file names that you do not want included in your system backup image. The patterns in this file are input to the pattern matching conventions of the **grep** command to determine which files will be excluded from the backup. If you want to exclude files listed in the **/etc/exclude.rootvg** file, select the Exclude Files field and press the Tab key once to change the default value to yes.

For example, to exclude all the contents of the directory called scratch, edit the exclude file to read as follows:

```
/scratch/
```

For example, to exclude the contents of the directory called **/tmp**, and avoid excluding any other directories that have

/tmp in the pathname, edit the exclude file to read as follows:

```
^./tmp/
```

All files are backed up relative to . (current working directory). To exclude any file or directory for which the it is important to have the search match the string at the beginning of the line, use ^ (caret character) as the first character in the search string, followed by **.** (dot character), followed by the filename or directory to be excluded.

If the filename or directory being excluded is a substring of another filename or directory, use **^.** (caret character followed by dot character) to indicate that the search should begin at the beginning of the line and/or use **$** (dollar sign character) to indicate that the search should end at the end of the line.

−f*fix_bundle* Optional file with a list of APARs to install after a clone of rootvg. The **−l** flag must be used with this option.

−F*fixes* Optional list of APARs (for example, "IX123456") to install after a clone of rootvg. The **−l** flag must be used with this option.

−I*installp_flags* The flags to use when updating or installing new filesets into the cloned alt_inst_rootvg. Default flags: "−acgX" The **−l** flag must be used with this option.

−l*images_location* Location of installp images or updates to apply after a clone of rootvg. This can be a directory full pathname or device name (like **/dev/rmt0**).

−w*filesets* List of filesets to install after cloning a rootvg. The **−l** flag must be used with this option.

The following flags are available for **alt_disk_install** version 4.3.2 or greater:

−q*disk* Used to return the volume group boot disk name. This is especially useful when trying to determine the boot disk from several disks in the "old_rootvg" volume group, after rebooting from the alternate disk.

−S Will "put−to−sleep" the volume group. This is used after a volume group "wake−up". (**−W**).

−v*new_volume_group_namedisk* Used to rename the alternate disk volume group. This is especially useful when creating multiple alternate disks, on multiple volume groups, and name identification is necessary.

−W*disk* Used to "wake−up" a volume group for data access between the rootvg and the alternate disk rootvg.

Note: The volume group that experiences the "wake−up" will be renamed "altinst_rootvg".

Limitation

The running system's version of AIX must be greater than or equal to the AIX version of the volume group that undergoes the "wake−up". This may mean that it's necessary to boot from the "altinst_rootvg" and "wake−up" the "old_rootvg".

Parameters

target_disks Specifies the name or names of the target disks where the alternate rootvg will be created. This disk or these disks must not currently contain any volume group definition. The **lspv** command should show these disks as belonging to volume group **None**.

Examples

1. To clone the running 4.2.0 rootvg to hdisk3, then apply updates from /updates to bring the cloned rootvg to a 4.2.1 level:
   ```
   alt_disk_install −C −F 4.2.1.0_AIX_ML −l /updates hdisk3
   ```

   The bootlist would then be set to boot from hdisk3 at the next reboot.

2. To install a 4.3 mksysb image on hdisk3, then run a customized script (/home/myscript) to copy some user files over to the alternate rootvg file systems before reboot:
   ```
   alt_disk_install −d /mksysb_images/4.3_mksysb −s /home/myscript hdisk3
   ```

3. To remove the original rootvg ODM database entry, after booting from the new alternate disk:
   ```
   alt_disk_install −X old_rootvg
   ```

   The **lspv** listing for the original rootvg will be changed to "None". Therefore, a new volume group could be created on those disks.

4. To determine the boot disk for a volume group with multiple physical volume:
   ```
   alt_disk_install −q hdisk0
   ```

   Illustrated Example

   ```
   # lspv
   hdisk0          00006091aef8b687     old_rootvg
   hdisk1          00076443210a72ea     rootvg
   hdisk2          0000875f48998649     old_rootvg
   # alt_disk_install −q hdisk0
   hdisk2
   ```

   In this case, the boot disk for "old_rootvg" is actually hdisk2. Therefore, you could reset your bootlist to hdisk2 and reboot to the original rootvg volume group.

5. To modify an **alt_disk_install** volume group name:
   ```
   alt_disk_install −v alt_disk_432 hdisk2
   ```

   Illustrated Example

   ```
   # lspv
   hdisk0          00006091aef8b687     rootvg
   hdisk1          00000103000d1a78     rootvg
   hdisk2          000040445043d9f3     altinst_rootvg
   hdisk3          00076443210a72ea     altinst_rootvg
   hdisk4          0000875f48998649     None
   hdisk5          000005317c58000e     None
   # alt_disk_install −v alt_disk_432   hdisk2
   #lspv
   hdisk0          00006091aef8b687     rootvg
   hdisk1          00000103000d1a78     rootvg
   hdisk2          000040445043d9f3     alt_disk_432
   ```

```
hdisk3          00076443210a72ea     alt_disk_432
hdisk4          0000875f48998649     None
hdisk5          000005317c58000e     None
```

6. To "wake_up" an original rootvg, after booting from the new alternate disk:
   `alt_disk_install −W hdisk0`

   Illustrated Example

   ```
   # lspv
   hdisk0          000040445043d9f3     old_rootvg
   hdisk1          00076443210a72ea     rootvg
   # alt_disk_install −W hdisk0
   # lspv
   hdisk0          000040445043d9f3     altinst_rootvg
   hdisk1          00076443210a72ea     rootvg
   ```

   At this point, the "altinst_rootvg" volume group is varied−on and the /alt_inst file systems will be mounted.

7. To "put−to−sleep" a volume group that had experienced a "wake−up":
   `alt_disk_install −S`

   Illustrated Example

   ```
   # lspv
   hdisk0          000040445043d9f3     altinst_rootvg
   hdisk1          00076443210a72ea     rootvg
   # alt_disk_install −S
   # lspv
   hdisk0          000040445043d9f3     altinst_rootvg
   hdisk1          00076443210a72ea     rootvg
   ```

   The "altinst_rootvg" is no longer varied−on and the /alt_inst file systems are no longer mounted. If it's necessary for the "altinst_rootvg" volume group name to be changed back to "old_rootvg", this can be done with the "−v" flag.

Files

/usr/sbin/alt_disk_install Contains the alt_disk_install command

Related Information

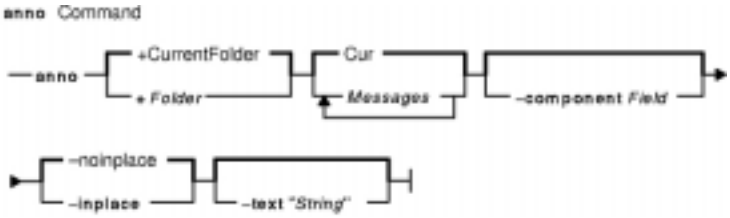
The **bootlist**, **nim**, **lspv**, and **bosboot** commands.

anno Command

Purpose

Annotates messages.

Syntax



anno [*+Folder*] [*Messages*] [**−component** *Field*] [**−inplace** | **−noinplace**] [**−text** "*String*"]

Description

The **anno** command annotates messages with text and dates. If you enter the **anno** command without any flags, the system responds with the following prompt:

```
Enter component name:
```

Typing a component name and pressing the Enter key annotates the component name and system date to the top of the message being processed. You cannot annotate an existing field. You can only add lines to the top of a message file. The annotation fields can contain only alphanumeric characters and dashes.

> **Note:** To simply add distribution information to a message, use the **dist**, **forw**, or **repl** commands.

Flags

−component *Field*
: Specifies the field name for the annotation text. The *Field* variable must consist of alphanumeric characters and dashes. If you do not specify this flag, the **anno** command prompts you for the name of the field.

+Folder
: Identifies the message folder that contains the message to annotate. The default is the current folder.

−help
: Lists the command syntax, available switches (toggles), and version information.
> **Note:** For MH (Message Handler), the name of this flag must be fully spelled out.

−inplace
: Forces annotation to be done in place in order to preserve links to the annotated messages.

Messages
: Specifies what messages to annotate. This parameter can specify several messages, a range of messages, or a single message. If several messages are specified, the first message annotated becomes the current message. Use the following references to specify messages:

Number
> Number of the message. When specifying several messages, separate each number with a comma. When specifying a range, separate the first and last

number in the range with a hyphen.

Sequence
A group of messages specified by the user. Recognized values include:

all
All messages in the folder.

cur *or* **.** *(period)*
Current message. This is the default.

first
First message in a folder.

last
Last message in a folder.

next
Message following the current message.

prev
Message preceding the current message.

−noinplace Prevents annotation in place. This flag is the default.

−text "*String*" Specifies the text to be annotated to the messages. The text must be enclosed with quotation marks.

Profile Entries

The following entries can be made to the *UserMhDirectory*/**.mh_profile** file:

`Current−Folder:` Sets the default current folder.

`Path:` Specifies the location of a user's MH (Message Handler) directory.

Examples

1. To annotate the message being processed with the date and time, enter:

   ```
   anno
   ```

   The following prompt is displayed on your screen:

   ```
   Enter component name: _
   ```

   After responding to this prompt, type:

   ```
   Date
   ```

   Press Enter. The component name you entered becomes the prefix to the date and time on the message. The caption appended to the message is similar to the following:

   ```
   Date: Tues, 28 Mar 89 13:36:32 −0600
   ```

2. To annotate the message being processed with the date, time, and a message, enter:

   ```
   anno −component NOTE −text "Meeting canceled."
   ```

   A two−line caption similar to the following is appended to the message:

   ```
   NOTE: Mon, 15 Mar 89 10:19:45 −0600
   NOTE: Meeting canceled.
   ```

3. To annotate message `25` in the `meetings` folder, enter:

```
anno +meetings 25 −component NOTE −text "Meeting delayed
until Friday."
```

   The top of message 25 is annotated with a caption similar to the following:

```
NOTE: Wed, 19 Jun 87 15:20:12 −0600
NOTE: Meeting delayed until Friday.
```

   Note: Do not press the Enter key until the entire message has been entered, even though the message may be wider than the screen.

Files

$HOME/.mh_profile	Contains the MH user profile.
/usr/bin/anno	Contains **anno** command.

Related Information

The **dist** command, **forw** command, **repl** command.

The **mh_alias** file format, **mh_profile** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

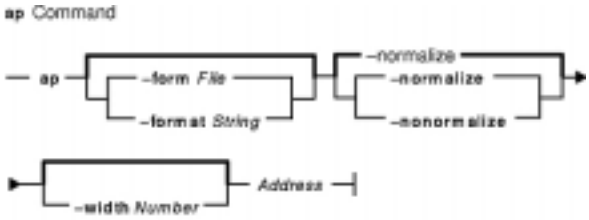
Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

ap Command

Purpose

Parses and reformats addresses.

Syntax



ap [**−form** *File* | **−format** *String*] [**−normalize** | **−nonormalize**] [**−width***Number*] *Address*

Description

The **ap** command parses and reformats addresses. The **ap** command is not started by the user. The **ap** command is called by other programs. The command is typically called by its full path name, **/usr/lib/mh/ap**.

The **ap** command parses each string specified by the address parameter and attempts to reformat it. The default output format for the **ap** command is the ARPA RFC 822 standard. When the default format is used, the **ap** command displays an error message for each string it is unable to parse.

Alternate file and string formats are specified by using the **−form** and **−format** flags.

Flags

−form *File*	Reformats the address string specified by the *Address* parameter into the alternate format described in the *File* variable.
−format *String*	Reformats the address string specified by the *Address* parameter into the alternate format specified by the *String* variable. The default format string follows: `%<{error}%{error}:%{Address}%:%(putstr(proper{ Address}))%>`
−help	Lists the command syntax, available switches (toggles), and version information. **Note:** For MH, the name of this flag must be fully spelled out.
−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 *Number*	Sets the maximum number of columns the **ap** command uses to display dates and error messages. The default is the width of the display.

Files

/etc/mh/mtstailor	Contains the MH tailor file.
$HOME/.mh_profile	Contains the MH user profile.

Related Information

The **ali** command, **dp** command, **scan** command.

The **.mh_alias** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

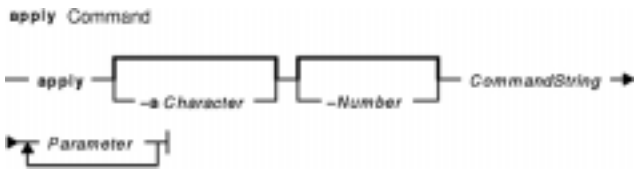
Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

apply Command

Purpose

Applies a command to a set of parameters.

Syntax



apply [−**a***Character*] [−*Number*] *CommandString Parameter* ...

Description

The **apply** command runs a command string specified by the *CommandString* parameter on each specified value of the *Parameter* parameter in turn. Normally, *Parameter* values are chosen individually; the optional −*Number* flag specifies the number of *Parameter* values to be passed to the specified command string. If the value of the *Number* variable is 0, the command string is run without parameters once for each *Parameter* value.

If you include character sequences of the form %*n* (where *n* is a digit from 1 to 9) in *CommandString*, they are replaced by the *n*th unused *Parameter* value following the *CommandString* parameter when the command string is executed. If any such sequences occur, the **apply** command ignores the −*Number* flag, and the number of parameters passed to *CommandString* is the maximum value of *n* in the *CommandString* parameter.

You can specify a character other than % (percent sign) to designate parameter substitution character strings with the **−a** flag; for example, `−a@` would indicate that the sequences @1 and @2 would be replaced by the first and second unused parameters following the *CommandString* parameter.

Notes:

1. Because pattern−matching characters in *CommandString* may have undesirable effects, it is recommended that complicated commands be enclosed in ' ' (single quotation marks).
2. You cannot pass a literal % (percent sign) followed immediately by any number without using the **−a** flag.

Flags

−**a***Character*	Specifies a character (other than %) to designate parameter substitution strings.
−*Number*	Specifies the number of parameters to be passed to *CommandString* each time it is run.

Examples

1. To obtain results similar to those of the **ls** command, enter:

```
apply echo *
```

2. To compare the file named **a1** to the file named **b1**, and the file named **a2** to the file named **b2**, enter:

   ```
   apply −2 cmp a1 b1 a2 b2
   ```

3. To run the **who** command five times, enter:

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

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

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

Related Information

The **xargs** command.

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

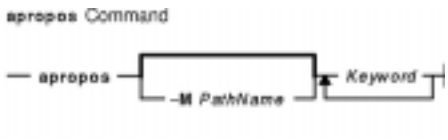
Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

apropos Command

Purpose

Locates commands by keyword lookup.

Syntax



apropos [**−M***PathName*] *Keyword* ...

Description

The **apropos** command shows the manual sections that contain any of the keywords specified by the *Keyword* parameter in their title. The **apropos** command considers each word separately and does not take into account if a letter is in uppercase or lowercase. Words that are part of other words are also displayed. For example, when looking for the word `compile`, the **apropos** command also finds all instances of the word `compiler`. The database containing the keywords is **/usr/share/man/whatis**, which must first be generated with the **catman −w** command.

If the output of the **apropos** command begins with a name and section number, you can enter **man***Section Title*. For example, if the output of the **apropos** command is `printf(3)`, you can enter `man 3 printf` to obtain the manual page on the **printf** subroutine.

The **apropos** command is equivalent to using the **man** command with the **−k** option.

> **Note:** When the **/usr/share/man/whatis** database is built from the HTML library using the **catman −w** command, section 3 is equivalent to section 2 or 3. See the **man** command for further explanation of sections.

Flag

−M*PathName* Specifies an alternative search path. The search path is specified by the *PathName* parameter, and is a colon−separated list of directories.

Examples

1. To find the manual sections that contain the word `password` in their titles, enter:

   ```
   apropos password
   ```

2. To find the manual sections that contain the word `editor` in their titles, enter:

   ```
   apropos editor
   ```

File

/usr/share/man/whatis Contains the **whatis** database.

Related Information

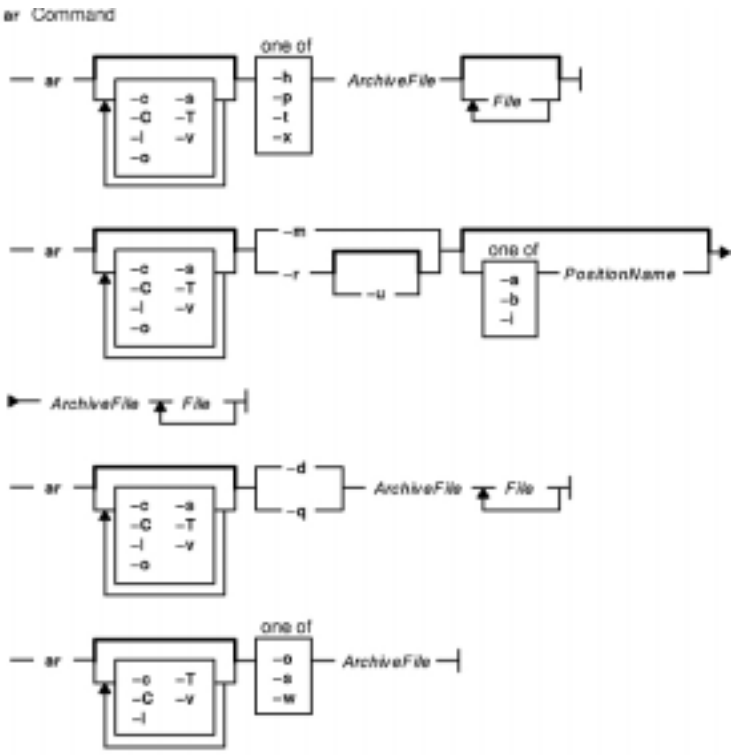
The **catman** command, **man** command, **whatis** command.

ar Command

Purpose

Maintains the indexed libraries used by the linkage editor.

Syntax



ar [**−c**] [**−l**] [**−g** | **−o**] [**−s**] [**−v**] [**−C**] [**−T**] { **−h** | **−p** | **−t** | **−x** }
[**−X** {**32**|**64**|**32_64**}]*ArchiveFile* [*File ...*]

ar [**−c**] [**−l**] [**−g** | **−o**] [**−s**] [**−v**] [**−C**] [**−T**] { **−m** | **−r** [**−u**] } [{ **−a** | **−b** | **−i** }
[**−X** {**32**|**64**|**32_64**}] { *PositionName* }] *ArchiveFile File ...*

ar [**−c**] [**−l**] [**−g** | **−o**] [**−s**] [**−v**] [**−C**] [**−T**] { **−d** | **−q** }
[**−X** {**32**|**64**|**32_64**}]*ArchiveFileFile ...*

ar [**−c**] [**−l**] [**−v**] [**−C**] [**−T**] { **−g** | **−o** | **−s** | **−w** } [**−X** {**32**|**64**|**32_64**}]*ArchiveFile*

Description

The **ar** command maintains the indexed libraries used by the linkage editor. The **ar** command combines one or more named files into a single archive file written in **ar** archive format. When the **ar** command creates a library, it creates headers in a transportable format; when it creates or updates a library, it rebuilds the symbol table. See the **ar** file format entry for information on the format and structure of indexed archives and symbol tables.

There are two file formats that the **ar** command recognizes. The Big Archive Format, **ar_big**, is the default file format and supports both 32−bit and 64−bit object files. The Small Archive Format can be used to create

archives that are recognized on version of AIX older than 4.3, see the **−g** flag. If a 64−bit object is added to a small format archive, **ar** first converts it to the big format, unless **−g** is specified. By default, **ar** only handles 32−bit object files; any 64−bit object files in an archive is silently ignored. To change this behavior, use the **−X** flag or set the **OBJECT_MODE** environment variable.

Flags

In an **ar** command, you can specify any number of optional flags from the set **cClosTv**. You must specify one flag from the set of flags **dhmopqrstwx**. If you select the **−m** or **−r** flag, you may also specify a positioning flag (**−a**, **−b**, or **−i**); for the **−a**, **−b**, or **−i** flags, you must also specify the name of a file within *ArchiveFile* (*PositionName*), immediately following the flag list and separated from it by a blank.

−a *PositionName* Positions the named files after the existing file identified by the *PositionName* parameter.

−b *PositionName* Positions the named files before the existing file identified by the *PositionName* parameter.

−c Suppresses the normal message that is produced when *library* is created.

−C Prevents extracted files from replacing like−named files in the file system.

−d Deletes the named files from the library.

−g Orders the members of the archive to ensure maximum loader efficiency with a minimum amount of unused space. In almost all cases, the **−g** flag physically positions the archive members in the order in which they are logically linked. The resulting archive is always written in the small format, so this flag can be used to convert a big−format archive to a small−format archive. Archives that contain 64−bit **XCOFF** objects cannot be created in or converted to the small format.

−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, the **ar** command sets the time stamps of all member headers.

−i *PositionName* Positions the named files before the existing file identified by the *PositionName* parameter (same as the **−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.

−o Orders the members of the archive to ensure maximum loader efficiency with a minimum amount of unused space. In almost all cases, the **−o** flag physically positions the archive members in the order in which they are logically linked. The resulting archive is always written in the big archive format, so this flag can be used to convert a small−format archive to a big−format archive.

−p Writes to standard output the contents of the named in the *Files* parameter, or all files specified in the *ArchiveFile* parameter if you do not specify any files.

−q Adds the named files to the end of 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), the **−r** flag replaces only files modified since they were last added to the library file.

If a named file does not already appear in the library, the **ar** command 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 the **ar** command 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, the **−t** flag lists all files in the library.
−T	Allows file name truncation if the archive member name is longer than the file system supports. This option has no effect because the file system supports names equal in length to the maximum archive member name of 255 characters.
−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. 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.
−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, the **−x** flag copies all files out of the library. This process does not alter the library.
−X*mode*	Specifies the type of object file **ar** should examine. The *mode* must be one of the following: *32* Processes only 32−bit object files *64* Processes only 64−bit object files *32_64* Processes both 32−bit and 64−bit object files The default is to process 32−bit object files (ignore 64−bit objects). The *mode* can also be set with the **OBJECT_MODE** environment variable. For example, **OBJECT_MODE=64** causes **ar** to process any 64−bit objects and ignore 32−bit objects. The **−X** flag overrides the **OBJECT_MODE** variable.
ArchiveFile	Specifies an archive file name; required.
MemberName ...	Names of individual archive members.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To create a library, enter:

   ```
   ar −v −q lib.a strlen.o strcpy.o
   ```

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

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

   ```
   ar −v −t lib.a
   ```

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

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

   ```
   ar −v −r lib.a strlen.o strcat.o
   ```

   This command replaces the members `strlen.o` and `strcat.o`. If `lib.a` was created as shown in example 1, then 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, enter:

   ```
   ar −v −r −b strlen.o lib.a strcmp.o
   ```

   This command adds the `strcmp.o` file, placing the new member before the `strlen.o` member.

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

   ```
   ar −v −r −u lib.a strcpy.o
   ```

   This command 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, enter:

   ```
   ar −v −m −a strcmp.o lib.a strcat.o strcpy.o
   ```

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

7. To extract library members, enter:

   ```
   ar −v −x lib.a strcat.o strcpy.o
   ```

   This command 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, enter:

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

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

9. To delete a member, enter:

   ```
   ar −v −d lib.a strlen.o
   ```

   This command deletes the member `strlen.o` from the `lib.a` library.

10. To create an archive library from multiple shared libraries created with the **ld** command, enter :

    ```
    ar −r −v libshr.a shrsub.o shrsub2.o shrsub3.o ...
    ```

    This command creates an archive library named `libshr.a` from the shared libraries named `shrsub.o`, `shrsub2.o`, `shrsub3.o`, and so on. To compile and link the `main` program using the `libshr.a` archive library, use the following command:

```
cc −o main main.c −L/u/sharedlib −lshr
```

The `main` program is now executable. Any symbols referenced by the `main` program that are contained by the `libshr.a` archive library have been marked for deferred resolution. The **−l** flag specifies that the `libshr.a` library be searched for the symbols.

11. To list the contents of **lib.a**, ignoring any 32−bit object file, enter:
```
ar −X64 −t −v lib.a
```

12. To extract all 32−bit object files from **lib.a**, enter:
```
ar −X32 −x lib.a
```

13. To list all files in **lib.a**, whether 32−bit, 64−bit, or non−objects, enter:
```
ar −X32_64 −t −v lib.a
```

File

/tmp/ar* Contains temporary files.

Related Information

The **ld** command, **lorder** command, **make** command, **nm** command, **strip** command.

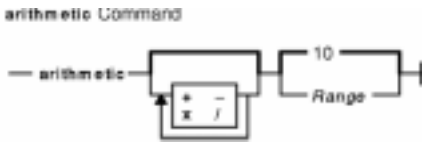
The **a.out** file format, **ar** file format (Big), **ar** file format (Small).

arithmetic Command

Purpose

Tests arithmetic skills.

Syntax



arithmetic [+] [−] [**x**] [/] [*Range*]

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. After a set of 20 problems, the **arithmetic** command 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.

To quit the game, press the Interrupt (Ctrl−C) key sequence; the **arithmetic** command displays the final game statistics and exits.

Flags

The optional flags modify the action of the **arithmetic** command. These flags are:

+ Specifies addition problems.

− Specifies subtraction problems.

x Specifies multiplication problems.

/ Specifies division problems.

Range A decimal number that specifies the permissible range of numbers. This range goes up to and includes 99. For addition and multiplication problems, the range applies to all numbers (except answers). For subtraction and division problems, the range applies only to the answers. At the start of the game, 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.

If you do not select any flags, the **arithmetic** command selects addition and subtraction problems and a default range of 10. 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:

```
arithmetic
```

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

```
arithmetic  +x/  50
```

File

/usr/games Location of the system's games.

Related Information

The **back** command, **bj** command, **craps** command, **fish** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command, **turnon** command, **wump** command.

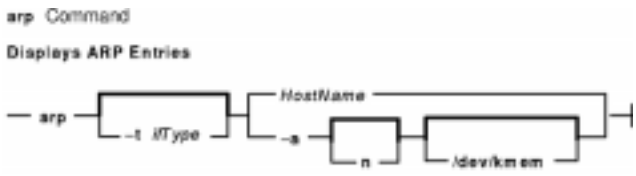
arp Command

Purpose

Displays and modifies address resolution, including ATM (Asyncronous Transfer Mode) interfaces.

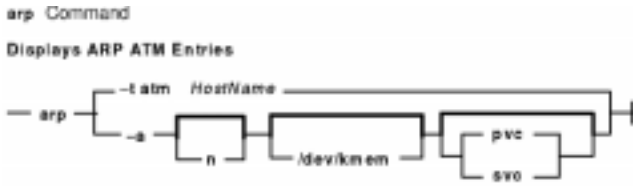
Syntax

To Display ARP Entries



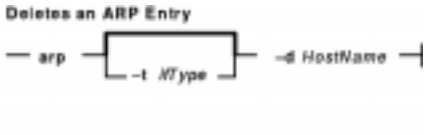
arp {[**−t***ifType*] *HostName* | **−a**[**n**] [**/dev/kmem**] }

To Display ARP ATM Entries



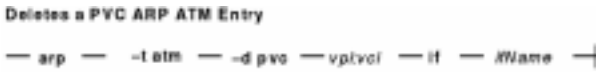
arp { **−t atm***HostName* | **−a**[**n**] [**/dev/kmem**] [**pvc** | **svc**] }

To Delete an ARP Entry



arp [**−t***ifType*] **−d** *HostName*

To Delete a PVC ARP ATM Entry

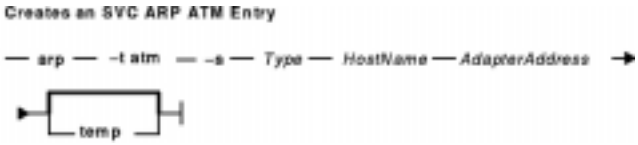


arp −t atm−dpvc*vpi:vci***if***ifName*

To Create an ARP Entry

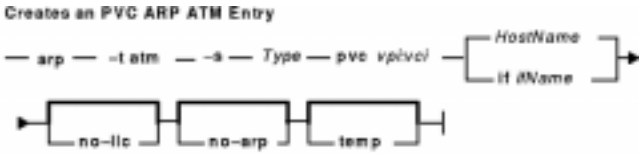
arp [−**t***ifType*] −**s** *Type HostNameAdapterAddress* [*Route*] [**temp**] [**pub**]

To Create an SVC ARP ATM Entry



arp −t atm −s *Type HostNameAdapterAddress* [**temp**]

To Create a PVC ARP ATM Entry



arp −t atm −s *Type* **pvc***vpi:vci* { *HostName* | **if***ifName }* [**no−llc**] [**no−arp**] [**temp**]

To Import ARP Entries from Another File

arp [−**t***ifType*] −**f** *FileName* [*Type*]

Description

The **arp** command displays and modifies the Internet−to−adapter address translation tables used by the Address Resolution Protocol. The **arp** command displays the current ARP entry for the host specified by the *HostName* variable. The host may be specified by name or number, using Internet dotted decimal notation.

Flags

−a

Used as {[**−t** *ifType*] *HostName* | **−a** [**n**] [**/dev/kmem**] }

Displays all of the current ARP entries. Use the **crash** command to look at KMEM or UMUnix variables. Specify the **−a /dev/kmem** flag to display ARP information for kernel memory. The '**n**' modifier causes hostname lookups to be suppressed.

Used as { **−t atm***HostName* | **−a** [**n**] [**/dev/kmem**] [**pvc** | **svc**] }

The **pvc** specification will display only ATM PVC (Permanent Virtual Circuits) types of virtual circuits, **svc** specification will display only ATM SVC (Switched

Virtual Circuits) types of virtual circuits. If the **pvc** | **svc** parameter is omitted, all ATM virtual circuits will be displayed.

−d Used as [**−t** *ifType*] **−d** *HostName*

Deletes an entry for the host specified by the *HostName* variable if the user has root user authority.

Used as **−t atm−dpvc***vpi:vci***if***ifName*

Deletes a PVC ARP entry by specifying *vpi:vci* rather than hostname. The *vpi:vc*i variables specify the virtual circuit that is to be deleted. The *ifname* variable specifies the name of the ATM interface on which the virtual circuit is to be deleted.

−f *FileName* `[Type]` Causes the file specified by the *FileName* variable to be read and multiple entries to be set in the ARP tables. Entries in the file should be in the form:

```
[Type] HostName AdapterAddress [Route] [temp] [pub]
```

where

Type
: Specifies the type of hardware address. If the address type is specified when invoking **arp** from the command line, it should not be specified in the file entries. Otherwise it should be specified in each file entry. Valid hardware address types are:
  - ◊ `ether` for an Ethernet interface
  - ◊ `802.3` for an 802.3 interface
  - ◊ `fddi` for a Fiber Distributed Data interface
  - ◊ `802.5` for a Token−Ring interface

HostName
: Specifies the remote host.

AdapterAddress
: Specifies the hardware address of the adapter for this host as 6 hexadecimal bytes separated by colons. Use the **netstat −v** command to display the local hardware address.

Route
: Specifies the route for a Token−Ring interface or Fiber Distributed Data Interface (FDDI) as defined in the Token−Ring or FDDI header.

temp
: Specifies that this ARP table entry is temporary. The table entry is permanent if this argument is omitted.

pub
: Specifies that this table entry is to be published, and that this system will act as an ARP server responding to requests for `HostName`, even though the host address is not its own.
  Note: The **−f** flag is not supported for ATM.

−s Used as [**−t** *ifType*] **−s** *Type HostNameAdapterAddress* [*Route*] [**temp**] [**pub**]

Creates an ARP entry of the type specified by the *Type* variable for the host specified by the *HostName* variable with the adapter address specified by the *AdapterAddress* variable. The adapter address is given as 6 hexadecimal bytes separated by colons. The line must be in the following format:

```
Type HostName AdapterAddress [Route] [temp] [pub]
```

where the *Type*, *HostName*, *AdapterAddress*, *Route*, **temp**, and **pub** parameters have the same purpose and definitions as the parameters for the **−f** flag.

Used as **−t atm −s** *Type HostNameAdapterAddress* [**temp**]

Creates a SVC type of ARP entry for the remote host, specified by the *HostName* variable, with the ATM address specified by the *ATMAddress* variable. The ATM address is given as 20 hexadecimal bytes separated by colons. Creation of this entry causes this IP station to not use ARP server mechanism to resolve IP addresses.

Used as **−t atm −s** *Type* **pvc***vpi:vci* { *HostName* | **if***ifName }* [**no−llc**] [**no−arp**] [**temp**]

Creates a PVC type of ARP entry for the remote host, specified by the *HostName* variable, with the PVC specified by the *vpi:vci*. Either destination *Hostname* or the local *ifname* needs to be specified. The **no−llc** flag is used to indicate that LLC/SNAP encapsulation will not be used on this virtual circuit, in this case, the destination *Hostname* needs to be specified. The **no−arp** flag is used to indicate that ARP protocol will not be used on this virtual circuit, in this case, the destination *Hostname* needs to be specified.

The *temp* parameter specifies that this ARP table entry is temporary, the table entry is permanent if this argument is omitted.

−t *ifType*

The **−t** *iftype* flag is used to indicate the type of Network interface. It is optional for the following types of interfaces:

- et for IEEE 802.3 Ethernet (inet, xns)
- tr for Token−Ring (inet, xns)
- xt for X.25 (inet)
- sl for serial line IP (inet)
- lo for loopback (inet)
- op for serial (inet)

The **−t atm** flag is required for the following interfaces:

- at for ATM

Examples

1. To add a single entry to the **arp** mapping tables until the next time the system is restarted, enter:

   ```
   arp −s 802.3 host2 0:dd:0:a:85:0 temp
   ```

2. To delete a map table entry for the specified host with the **arp** command, enter:

   ```
   arp   −d   host1 flag
   ```

3. To display arp entries for atm host `host1` , enter:

   ```
   arp −t atm −a host1
   ```

4. To add a PVC arp entry for atm host `host2`, enter:

   ```
   arp   −t atm   −s atm pvc 0:20 host2
   ```

5. To add a PVC arp entry for an interface `at0`, enter:

   ```
   arp   −t atm   −s atm pvc 0:20 if at0
   ```

Related Information

The **crash** command, **ifconfig** command, **netstat** command.

The **inetd** daemon.

TCP/IP Protocols in *AIX Version 4.3 System Management Guide: Communications and Networks*.

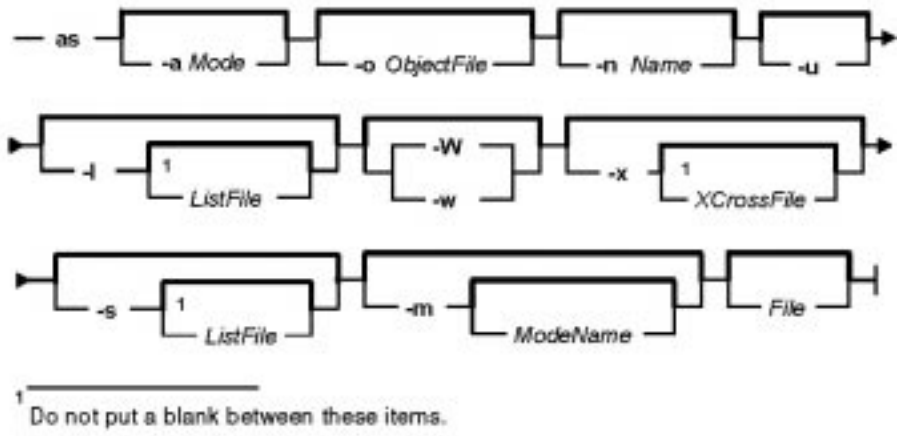
as Command

Purpose

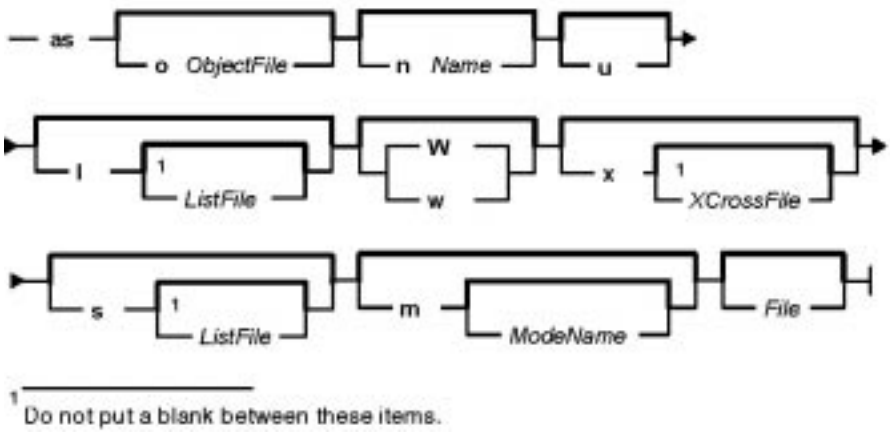
Reads and assembles a source file.

Syntax

as Command



as Command



as [**−a** *Mode*] [**−o** *ObjectFile*] [**−n** *Name*] [**−u**] [**−l** [*ListFile*]] [**−W** | **−w**] [**−x**[*XCrossFile*]] [**−s** [*ListFile*]] [**−m** *ModeName*] [*File*]

Description

The **as** command reads and assembles the named *File* (by convention, this file ends with a **.s** suffix). If you do not specify a *File*, the **as** command reads and assembles standard input. It stores its output, by default, in a file named **a.out**. The output is stored in the XCOFF file format.

All flags for the **as** command are optional.

Flags

−a *Mode*
Specifies the mode in which the **as** command operates. By default, the **as** command operates in 32−bit mode, but the mode can be explicitly set by using the flag **−a32** for 32−bit mode operation or **−a64** for 64−bit mode operation.

any
Specifies the indiscriminate mode. The assembler generates object code for any recognized instruction, regardless of architecture. This mode is used primarily for operating system development and for testing and debugging purposes.

> **Note:** All POWER/PowerPC incompatibility errors are ignored when using the **any** assembly mode, and no warnings are generated.

A35
Specifies the A35 mode. A source program can contain only instructions for the A35.

File
Specifies the source file. If no file is specified, the source code is taken from standard input.

−l[*ListFile*]
Produces an assembler listing. If you do not specify a file name, a default name is produced by replacing the suffix extension of the source file name with a **.lst** extension. By convention, the source file suffix is a **.s**. For example:

```
sourcefile.xyz
```

produces a default name of:

```
sourcefile.lst
```

If the source code is from standard input and the **−l** flag is used without specifying an assembler−listing file name, the listing file name is **a.lst**.

−m*ModeName*
Indicates the assembly mode. This flag has lower priority than the **.machine** pseudo−op.

If this flag is not used and no **.machine** pseudo−op is present in the source program, the default assembly mode is used. The default assembly mode has the POWER/PowerPC intersection as the target environment, but treats all POWER/PowerPC incompatibility errors (including instructions outside the POWER/PowerPC intersection and invalid form errors) as instructional warnings.

If an assembly mode that is not valid is specified and no **.machine** pseudo−op is present in the source program, an error is reported and the default assembly mode is used for instruction validation in pass 1 of the assembler.

If the **−m** flag is used, the *ModeName* variable can specify one of the following values:

""
Explicitly specifies the default assembly mode that has the POWER/PowerPC intersection as the target environment, but treats instructions outside the POWER/PowerPC intersection and invalid form errors as instructional warnings. A space is required between **−m** and the null string argument (two double quotation marks).

com
Specifies the POWER/PowerPC intersection mode. A source program can contain only instructions that are common to both POWER and PowerPC; any other instruction causes an error. Any instruction with an invalid form causes errors,

terminates the assembly process, and results in no object code being generated.

Note: Certain POWER instructions are supported by the PowerPC 601 RISC Microprocessor, but do not conform to the PowerPC architecture. These instructions cause errors when using the **com** assembly mode.

−n *Name* Specifies the name that appears in the header of the assembler listing. By default, the header contains the name of the assembler source file.

−o *ObjectFile* Writes the output of the assembly process to the specified file instead of to the **a.out** file.

ppc Specifies the PowerPC mode. A source program can contain only PowerPC instructions. Any other instruction causes an error.

Notes:

1. The PowerPC optional instructions are implemented in each PowerPC processor and do not belong to the **ppc** mode. These instructions generate an error if they appear in a source program that is assembled using the **ppc** assembly mode.
2. Certain instructions conform to the PowerPC architecture, but are not supported by the PowerPC 601 RISC Microprocessor.

ppc64 Specifies the PowerPC 64−bit mode. A source program can contain 64−bit PowerPC instructions.

pwr Specifies the POWER mode. A source program can contain only instructions for the POWER implementation of the POWER architecture.

pwr2(pwrx) Specifies the POWER2 mode. A source program can contain only instructions for the POWER2 implementation of the POWER architecture. **pwr2** is the preferred value. The alternate assembly mode value **pwrx** means the same thing as **pwr2**.

Note: The POWER implementation instruction set is a subset of the POWER2 implementation instruction set.

−s[*ListFile*] Indicates whether or not a mnemonics cross−reference for POWER and PowerPC is included in the assembler listing. If this flag is omitted, no mnemonics cross−reference is produced. If this flag is used, the assembler listing will have POWER mnemonics if the source contains PowerPC mnemonics, and will have PowerPC mnemonics if the source contains POWER mnemonics.

The mnemonics cross−reference is restricted to instructions that have different mnemonics in POWER and PowerPC, but that have the same op code, function, and input operand format.

Because the **−s** flag is used to change the assembler−listing format, it implies the **−l** flag. If both option flags are used and different assembler−listing file names (specified by the *ListFile* variable) are given, the listing file name specified by the *ListFile* variable used with the **−l** flag is used. If an assembler−listing file name is not specified with either the **−l** or **−s** flag, a default assembler listing file name is produced by replacing the suffix extension of the source file name with a **.lst** extension.

−u Accepts an undefined symbol as an extern so that an error message is not displayed. Otherwise, undefined symbols are flagged with error messages.

−W Turns off all warning message reporting, including the instructional warning messages (the POWER and PowerPC incompatibility warnings).

−w Turns on warning message reporting, including reporting of instructional warning messages (the POWER and PowerPC incompatibility warnings).

Note: When neither **−W** nor **−w** is specified, the instructional warnings are reported, but other warnings are suppressed.

−x[*XCrossFile*] Produces cross reference output. If you do not specify a file name, a default name is produced by replacing the suffix extension of the source file name with an **.xref** extension. Conventionally, the suffix is a **.s**. For example:

```
sourcefile.xyz
```

produces a default name of:

```
sourcefile.xref
```

> **Note:** The assembler does not generate an object file when the **−x** flag is used.

601 Specifies the PowerPC 601 RISC Microprocessor mode. A source program can contain only instructions for the PowerPC 601 RISC Microprocessor.

> **Note:** The PowerPC 601 RISC Microprocessor design was completed before the PowerPC architecture. Therefore, some PowerPC instructions may not be supported by the PowerPC 601 RISC Microprocessor.
>
> **Attention:** It is recommended that the **601** assembly mode not be used for applications that are intended to be portable to future PowerPC systems. The **com** or **ppc** assembly mode should be used for such applications.

The PowerPC 601 RISC Microprocessor implements the PowerPC architecture plus some POWER instructions are not included in the PowerPC architecture. This allows existing POWER applications to run with acceptable performance on PowerPC systems. Future PowerPC systems will not have this feature. The **601** assembly mode may result in applications that will not run on existing POWER systems and that may not have acceptable performance on future PowerPC systems, because the **601** assembly mode permits the use of all the instructions provided by the PowerPC 601 RISC Microprocessor.

603 Specifies the PowerPC 603 RISC Microprocessor mode. A source program can contain only instructions for the PowerPC 603 RISC Microprocessor.

604 Specifies the PowerPC 604 RISC Microprocessor mode. A source program can contain only instructions for the PowerPC 604 RISC Microprocessor.

Environment Variables

OBJECT_MODE

The assembler respects the setting of the OBJECT_MODE environment variable. If neither **−a32** or **−a64** is used, the environment is examined for this variable. If the value of the variable is anything other than the values listed in the following table, an error message is generated and the assembler exits with a non−zero return code. The implied behavior corresponding to the valid settings are as follows:

OBJECT_MODE=**32**	Produce 32−bit object code. The default machine setting is **com**.
OBJECT_MODE=**64**	Produce 64−bit object code (XCOFF64 files). The default machine setting is **ppc64**.
OBJECT_MODE=**32_64**	Invalid.
OBJECT_MODE=*anything else*	Invalid.

Examples

1. To produce a listing file named file.lst and an object file named **file.o**, enter:
   ```
   as −l −o file.o file.s
   ```

2. To produce an object file named **file.o** that will run on the 601 processor and generate a cross reference for POWER and PowerPC mnemonics in an assembler listing file named **file.lst**, enter:

```
as   -s   -m   601   -o   file.o   file.s
```

3. To produce an object file named **file.o** using the default assembly mode and an assembler listing file named **xxx.lst** with no mnemonics cross reference, enter:
```
as   -lxxx.lst   -o   file.o   file.s
```

Files

/usr/ccs/bin/as	Contains the **as** command
a.out	The default output file.

Related Information

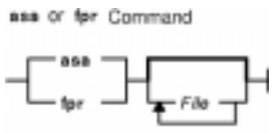
The **ld** command, **m4** command.

asa or fpr Command

Purpose

Prints FORTRAN files to AIX line−printer conventions.

Syntax



{ **asa** | **fpr** } [*File* ...]

Description

The **asa** and **fpr** commands print FORTRAN files to conform to AIX line−printer conventions. Both commands work like a filter to transform files formatted according to FORTRAN carriage control conventions into files formatted according to AIX line−printer conventions.

The *File* variable specifies the name of the input file that the **asa** and **fpr** commands read instead of the standard input. The **asa** and **fpr** commands read the file, replace the carriage control characters with recognizable AIX characters, and print the file to standard output.

Both commands read the first character of each line from the input file, interpret the character, and space the line according to the definition of the first character. If the first character is either a **Blank**, a **0**, a dash (−) , a **1**, or a plus sign (+), either command does the following:

Blank Advances the carriage one line and prints the input line.
0 Advances the carriage two lines and prints the input line.
− Advances the carriage three lines and prints the input line.
1 Advances the carriage to the top of the next page.
+ Does not advance the carriage and starts printing the input line in the first space of the output file.

The commands interpret a blank line as if its first character is a blank and delete a blank that appears as a carriage control character. It treats lines that begin with characters other than the defined control characters as if they begin with a blank character. The first character of a line is not printed. If any such lines appear, an appropriate diagnostic appears in the standard error.

Note: Results are undefined for input lines longer than 170 characters.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. Use the **fpr** command in the following manner to change the carriage control characters in an `a.out` file produced by a FORTRAN compiler into carriage control characters and print the resulting file:

   ```
   a.out | fpr | qprt
   ```

2. Use the **asa** command in the following manner to run the `f77.output` file through the **asa** command to change carriage control characters from FORTRAN to the operating system and print the resulting file.

   ```
   asa f77.output | qprt
   ```

Files

/usr/ucb/fpr Contains the **fpr** command.

/usr/bin/asa Contains the **asa** command.

Related Information

The **fsplit** command, **qprt** command, **struct** command.

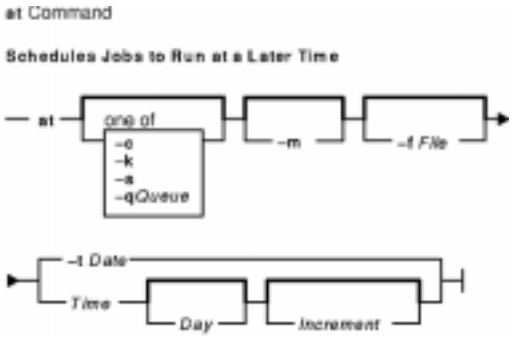
at Command

Purpose

Runs commands at a later time.

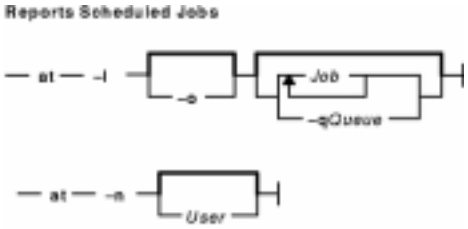
Syntax

To Schedule Jobs to Run at a Later Time



at [**−c** | **−k** | **−s** | **−q***Queue*] [**−m**] [**−f***File*] { **−t** *Date* |*Time* [*Day*] [*Increment*] }

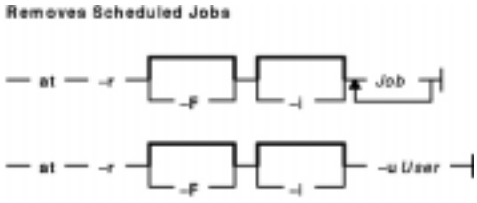
To Report Scheduled Jobs



at −l [**−o**] [*Job ...* | **−q***Queue*]

at−n [*User*]

To Remove Scheduled Jobs



at−r [**−F**] [**−i**] *Job ...*

at−r [**−F**] [**−i**] **−u***User*

Description

The **at** command reads from standard input the names of commands to be run at a later time and allows you

to specify when the commands should be run.

The **at** command mails you all output from standard output and standard error for the scheduled commands, unless you redirect that output. It also writes the job number and the scheduled time to standard error.

When the **at** command is executed, it retains the current process environment. It does not retain open file descriptors, traps, and priority.

The **/var/adm/cron/at.allow** and **/var/adm/cron/at.deny** files control which users can use the **at** command. A person with root user authority can create, edit, or delete these files. Entries in these files are user login names with one name to a line. The following is an example of an **at.allow** file:

```
root
nick
dee
sarah
```

If the **at.allow** file exists, only users whose login names appear in it can use the **at** command. A system administrator can explicitly stop a user from using the **at** command by listing the user's login name in the **at.deny** file. If only the **at.deny** file exists, any user whose name does not appear in the file can use the **at** command.

A user cannot use the **at** command if one of the following is true:

- The **at.allow** file and the **at.deny** file do not exist (allows root user only).
- The **at.allow** file exists but the user's login name is not listed in it.
- The **at.deny** file exists and the user's login name is listed in it.

If the **at.allow** file does not exist and the **at.deny** file does not exist or is empty, only someone with root user authority can submit a job with the **at** command.

To schedule a job to run at a later time, you must specify a time to start the job. You may specify the time by using either the **−t***Date* flag or the *Time*, *Day*, and *Increment* parameters.

The *Date* variable to the **−t** flag is specified using the following format:

[[*CC*]*YY*]*MMDDhhmm*[**.***SS*]

The digits in the *Date* variable are defined as follows:

CC	Specifies the first two digits of the year (the century).
YY	Specifies the second two digits of the year.
MM	Specifies the month of the year (01 through 12).
DD	Specifies the day of the month (01 through 31).
hh	Specifies the hour of the day (00 through 23).
mm	Specifies the minute of the hour (00 through 59).
SS	Specifies the second of the minute (00 through 59).

Both the *CC* and *YY* digits are optional. If neither is given, the current year is assumed. If the *YY* digits are specified but the *CC* digits are not, the *CC* digits are defined as follows:

- If the value of the *YY* digits is between 70 and 99, the value of the *CC* digits is assumed to be 19.
- If the value of the *YY* digits is between 00 and 37, the value of the *CC* digits is assumed to be 20.
- The default value of *SS* is 00.

The resulting time is affected by the value of the **TZ** environment variable.

The *Time* parameter may be specified as 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 **T_FMT** item in the **LC_TIME** locale category specifies the order of 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 one of the following suffixes:

- **am**
- **pm**
- **zulu**

If you do not specify **am** or **pm**, the **at** command uses a 24−hour clock. These suffixes can follow the time as a separate argument or separated with spaces. The **am** and **pm** suffixes are defined values from the **AM_STR** and **PM_STR** items in the **LC_TIME** locale category. The suffix **zulu** indicates that the time is **GMT** (Greenwich Mean Time).

The **at** command also recognizes the following keywords as special values for the *Time* parameter:

- **noon**
- **midnight**
- **now**
- **A** for AM
- **P** for PM
- **N** for noon
- **M** for midnight

You may specify the optional *Day* parameter as either a month name and a day number (and possibly a year number preceded by a comma), or a day of the week. The **D_FMT** item in the **LC_TIME** locale category specifies the order of the month and day (by default, month followed by day). The **DAY_1** through **DAY_7** items in the **LC_TIME** locale category specify long day names. The **ABDAY_1** through **ABDAY_7** items in the **LC_TIME** locale category specify short day names. The **MON_1** through **MON_12** items in the **LC_TIME** locale category specify long month names. The **ABMON_1** through **ABMON_12** items in the **LC_TIME** locale category specify short month names. By default, the long name is fully spelled out; the short name is abbreviated to two or more characters for weekdays, and three characters for months.

The **at** command recognizes **today** and **tomorrow** as special default values for the *Day* parameter. The **today** value is the default *Day* if the specified time is later than the current hour; the **tomorrow** value 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.

Flags

−c	Requests that the **csh** command be used for executing this job.
−f*File*	Uses the specified file as input rather than using standard input.
−F	Suppresses delete verification. Use this flag with the **−r** flag.
−i	Specifies interactive delete. Use this flag with the **−r** flag.
−k	Requests that the **ksh** command be used for executing this job.
−l	Reports your scheduled jobs. If you have root user authority, you can get jobs issued by other users.
−m	Mails a message to the user about the successful execution of the command.
−n [*User*]	Reports the number of files in your queue. If you have root user authority, you can get

information about another user's queue.

−o Lists jobs in schedule order. This flag is useful only with the **−l** flag.

−q*Queue* Specifies the queue in which to schedule a job for submission. When used with the **−l** flag, the report is limited to the queue specified by the *Queue* variable. By default, **at** jobs are scheduled in the **a** queue. The **b, c** and **d** queues are reserved for **batch** jobs, **cron** jobs, and **sync** jobs respectively.

−q a
Queues **at** jobs.

−q b
Queues **batch** jobs. The **batch** command calls the **at** command with this flag.
Note: When using the **b** queue, commands are read from standard input. Also, the **now** keyword is used for the *Time* parameter, regardless of what you specify on the command line.

−q e
Queues **ksh** jobs. Equivalent to the **−k** flag.

−q f
Queues **csh** jobs. Equivalent to the **−c** flag.

−r *Job*... Removes *Job*s previously scheduled by the **at** or **batch** commands, where *Job* is the number assigned by the **at** or **batch** commands. If you do not have root user authority (see the **su** command), you can remove only your own jobs. The **atrm** command is available to the root user to remove jobs issued by other users or all jobs issued by a specific user.

−s Requests that the **bsh** command (Bourne shell) be used for executing this job.

−t *Date* Submits the job to be run at the time specified by the *Date* variable.

−u*User* Deletes all jobs for the specified user. If used with the **−r** flag, do not specify a *Job* variable (the correct syntax is **at −r −u** *User*).

Parameters

Day Specifies the optional *Day* parameter as either a month name and a day number (and possibly a year number preceded by a comma), or a day of the week.

Increment The optional *Increment* parameter 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**]
- The special word **next** followed by a one of the following words:
  - **minute**[**s**]
  - **hour**[**s**]
  - **day**[**s**]
  - **week**[**s**]
  - **month**[**s**]
  - **year**[**s**]

Security

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the **at** command will generate the following audit record (event) every time the command is executed:

Event	Information

AT_JobAdd Lists **at** jobs that were run, the time the task was completed, and the user who issued the command.

See "Setting Up an Auditing System" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more details about how to properly select and group audit events, and how to configure audit event data collection.

Exit Status

This command returns the following exit values:

0 The **at** command successfully submitted, removed, or listed a job or jobs.

>0 An error occurred.

Examples

1. To schedule the command from the terminal, enter a command similar to one of the following:

   If uuclean is in your current directory, enter:

   ```
   at   5   pm   Friday
   uuclean
   <Ctrl-D>

   at   now   next   week
   uuclean
   <Ctrl-D>
   ```

   If uuclean is in **$HOME/bin/uuclean**, enter:

   ```
   at   now   +   2   days
   $HOME/bin/uuclean
   <Ctrl-D>
   ```

   > **Note:** When entering a command name as the last item on the command line, a full path name must be given if the command is not in the current directory, and the **at** command will not accept any arguments.

2. To run the **uuclean** command at 3:00 in the afternoon on the 24th of January, enter any one of the following commands:

   ```
   echo   uuclean   |   at   3:00   pm   January   24

   echo   uuclean   |   at   3pm   Jan   24

   echo   uuclean   |   at   1500   jan   24
   ```

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

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

4. To list the jobs you have sent to be run later, enter:

   ```
   at   -l
   ```

5. To cancel a job, enter:

```
at   -r   ctw.635677200.a
```

This cancels job `ctw.635677200.a`. Use the **at -l** command to list the job numbers assigned to your jobs.

Files

/var/adm/cron/FIFO	A named pipe that sends messages to the **cron** daemon when new jobs are submitted with the **crontab** or **at** commands.
/usr/bin/at	Contains the **at** command.
/var/adm/cron	Contains the main **cron** directory.
/var/adm/cron/at.allow	Specifies the list of allowed users.
/var/adm/cron/at.deny	Specifies the list of denied users.

/var/spool/cron/atjobs Contains the spool area directory for **at**.

Related Information

The **atq** command, **atrm** command, **auditpr** command, **batch** command, **bsh** command, **kill** command, **ksh** command, **mail** command, **nice** command, **ps** command, **sh** command, **su** command.

The **cron** daemon.

The **environment** file.

Auditing Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices* provides more information about audits and audit events.

Input and Output Redirection Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes how the operating system processes input and output.

National Language Support Overview for Programming in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains collating sequences, equivalence classes, and locale.

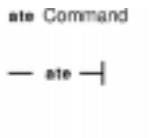
Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes what shells are, the different types of shells, and how shells affect the way commands are interpreted.

ate Command

Purpose

Starts the Asynchronous Terminal Emulation (ATE) program.

Syntax



ate

Description

The **ate** command starts the Asynchronous Terminal Emulation (ATE) program. The ATE program establishes a connection between a workstation and a remote computer. A workstation acts as a terminal connected to the remote computer. Using ATE the user can connect to, and exchange data with, remote databases and other systems.

> **Note:** Users must be a member of the UNIX−to−UNIX Copy Program (uucp) group in order to use ATE. A user with root authority uses System Management Interface Tool (SMIT) to install individual users in groups.

ATE establishes the connection and allows users to record and control the session. After logging in to the remote system, users execute programs, issue commands, and use files on the remote system as a local user. ATE also enables a workstation to emulate a VT100 terminal.

The ATE program uses menus and subcommands. From the menus, users issue subcommands to connect to a remote system, receive and transfer files, and execute commands. The Unconnected Main Menu displays any time users issue the **ate** command. The Connected Main Menu displays when users press the MAINMENU_KEY (usually the Ctrl−V key sequence) while connected to another system. The **connect** subcommand makes the connection.

The ATE program supports three control key sequences: the CAPTURE_KEY (usually Ctrl−B), PREVIOUS_KEY (usually CTRL−R), and MAINMENU_KEY (usually CTRL−V). These control keys do not function until the ATE program is started. The control keys and other ATE defaults can be changed by editing the **ate.def** file format.

Examples

To start the ATE program, enter:

```
ate
```

The ATE Unconnected Main Menu displays.

Subcommands

alter	Temporarily changes data transmission characteristics in the ATE program.
break	Interrupts current activity on a remote system.
connect	Connects to a remote computer.
directory	Displays the ATE dialing directory.
help	Provides help information for the ATE subcommands.
modify	Temporarily modifies local settings used for terminal emulation.
perform	Allows the user to issue workstation operating system commands while using ATE.
quit	Exits the Asynchronous Terminal Emulation (ATE) program.
receive	Receives a file from a remote system.
send	Sends a file to a remote system.
terminate	Terminates an ATE connection to a remote system.

alter Subcommand

a [**l***CharacterLength*] [**s***StopBit*] [**p***Parity*] [**r***BaudRate*] [**d***Device*] [**i***DialPrefix*] [**f***DialSuffix*] [**w***Seconds*] [**a***RedialAttempts*] [**t***TransferProtocol*] [**c***PacingType*]

> **Note:** The default values of the **alter** subcommand flags can be permanently changed by editing the **ate.def** file format.

The **alter** subcommand is accessed from the Asynchronous Terminal Emulation (ATE) Connected or Unconnected Main Menu. Issuing the **ate** command from the command line displays the Unconnected Main Menu. The **alter** subcommand temporarily changes these data transmission characteristics:

- Data character length
- Baud rate
- Stop and parity bits
- Port name
- Modem dialing prefixes and suffixes
- Waiting time and retry limits
- File transfer protocol
- Pacing character or delay time

The settings return to the defaults as defined in the **ate.def** file format when the user exits ATE.

When issued without flags from either of the ATE main menus, the **alter** subcommand displays the Alter Menu. To bypass the Alter Menu, enter the **alter** subcommand, followed by the appropriate flags, at the command prompt on either ATE main menu.

The **alter** subcommand can change more than one feature at a time. To change the value of more than one variable, type the first flag followed by the new value, followed by a space, then the second flag and second value, and so on.

To permanently change the settings affected by the **alter** subcommand, customize the **ate.def** file format.

The Alter Menu

The Alter Menu displays the current settings of the changeable characteristics with the **alter** subcommand. Enter the letter **a** after the command prompt on either the ATE Connected or Unconnected Main Menu to view the Alter Menu.

The Alter Menu contains the following columns:

Column Names	Contents
COMMAND	Flag that changes the value of a variable
DESCRIPTION	Description of the variable that the flag affects
CURRENT	Current value of the variable
POSSIBLE CHOICES	Possible values of the variable

To change the value of a variable, enter the flag (from the COMMAND column) and new value (from the POSSIBLE CHOICES column) at the command prompt on the Alter Menu.

To return to one of the ATE main menus from the Alter Menu, press the Enter key.

Flags

a *RedialAttempts* Specifies the maximum number of times the ATE program redials for a connection. If the *RedialAttempts* variable is 0, no redial attempt occurs.

Options: 0 (none) or a positive integer

Default: 0

c *PacingType* Specifies the type of **pacing** protocol used.

Default: 0 (no pacing)

> **Note:** The *PacingType* variable has no effect when the **xmodem** protocol is used.

The *PacingType* can be either of the following:

Character
: Signal to transmit a line. The signal can be any ASCII character.

  When the **send** subcommand encounters a line−feed character while transmitting data, it waits to receive the pacing character before sending the next line.

  When the **receive** subcommand is ready to receive data, it sends the pacing character and then waits 30 seconds to receive data. The **receive** subcommand sends a pacing character again whenever it finds a carriage−return character in the data. The **receive** subcommand ends when it receives no data for 30 seconds.

Interval
: Number of seconds the system waits between each line it transmits. The value of the *Interval* variable must be an integer. The default value is 0 indicating a pacing delay of 0 seconds.

d *Device* Specifies the name of the asynchronous port used to connect to a remote system.

Options: Locally created port names. The first 8 characters of the port name display in the Alter Menu.

Default: tty0

f *DialSuffix* Specifies the dial suffix that must follow the telephone number when autodialed with a modem. Consult the modem documentation for the proper dial command.

Options: 0 (none) or a valid modem suffix. The first 8 characters display in the Alter Menu.

Default: no default

i *DialPrefix* Specifies the dial prefix that must precede the telephone number when autodialed with a modem. Consult the modem documentation for the proper dial commands.

Options: ATDT, ATDP, or other values depending on the type of modem used. The first 8 characters display in the Alter Menu.

Default: ATDT

l *CharacterLength* Specifies the number of bits in a data character. This length must match the length expected by the remote system.

Options: 7 or 8

Default: 8

p *Parity* Checks whether a character was successfully transmitted to or from a remote system. Must match the parity of the remote system.

For example, if the user selects even parity, when the number of 1 bits in the character is odd, the parity bit is turned on to make an even number of 1 bits.

Options: 0 (none), 1 (odd), or 2 (even)

Default: 0

r*BaudRate* Specifies the baud rate, or bits transmitted per second (bps). The speed must match the speed of the modem and that of the remote system.

Options: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2400, 4800, 9600, or 19200

Default: 1200

s *StopBit* Specifies the number of stop bits appended to a character to signal the end of that character during data transmission. This number must match the number of stop bits used by the remote system.

Options: 1 or 2

Default: 1

t *TransferProtocol* Defines the type of asynchronous protocol that transfers files during a connection.

p

File transfer protocol controls the data transmission rate by waiting for either a specified character or a certain number of seconds between line transmissions. This helps prevent loss of data when the transmission blocks are either too large or sent too quickly for the system to process.

x

An 8−bit file transfer protocol to detect data transmission errors and retransmit the data.

Options: **p** (**pacing)**, or **x** (**xmodem**)

Default: **p**

w *Seconds* **wait**

Specifies the number of seconds between redial attempts. The wait period does not begin until the connection attempt times out or until it is interrupted. If the **attempts** flag is set to 0, no redial attempt occurs.

Options: 0 (none) or a positive integer

Default: 0

Examples

1. To display the Alter Menu, enter the **alter** subcommand at the command prompt on either ATE main menu:

   ```
   a
   ```

   The Alter Menu is displayed.

2. To alter transmission settings from the Alter Menu, enter the appropriate flags at the command prompt on the Alter Menu:

- To change the value for the **rate** flag, enter:

```
r 9600
```

For the current session of ATE, the baud rate is changed to `9600` bps.

- To change the value of the **wait** flag, enter:

```
w 7
```

For the current session of ATE, the wait time for redial changes to `7` seconds.

- To bypass the Alter Menu when using the **alter** command, type the command abbreviation **a**, followed by the appropriate flags, at the prompt on one of the ATE main menus. For example, to change the **rate**, **wait**, and **attempt** values, enter the following at the prompt on either ATE main menu:

```
a r 9600 w 5 a 1
```

For the current session of ATE, the baud rate changes to `9600` bps, the wait time for redial changes to `5` seconds, and the maximum number of redial attempts changes to `1` attempt.

break Subcommand

b

The **break** subcommand sends a break signal to the remote system connected to the terminal by the Asynchronous Terminal Emulation (ATE) program. The **break** subcommand interrupts current activity on the remote system. Issue the **break** subcommand from the ATE Connected Main Menu.

> **Attention:** The **break** subcommand may disconnect the current session. The system may lose data.

Example

To interrupt the current session, at the remote system login screen, press the MAINMENU_KEY (usually the Ctrl−V key sequence). When the ATE Connected Main Menu displays, enter:

```
b
```

A break signal is sent to the remote system, and the ATE Unconnected Main Menu displays. Now exit the ATE program or issue other ATE subcommands.

connect Subcommand

c [*TelephoneNumber* | *PortName*]

The ATE **connect** subcommand enables users to connect to a remote computer using Asynchronous Terminal Emulation (ATE). Issue the **connect** subcommand from the ATE Unconnected Main Menu. The connection can be made between two machines connected by cable or by telephone line. Users establish connection in one of three ways:

direct	Uses an established cabled link to another system.
manually dialed	Uses a telephone number dialed by the user.
automatically dialed	Uses a modem to dial a specified telephone number (a modem−dialed connection).

If the system login is not disabled, attempts to connect to another computer return an error. To disable the workstation port that handles system login by remote users, a user with root authority must use the **pdisable** command. Once the workstation port is secure from remote logins, the user must then ensure the remote system is ready to receive calls.

No connection is established if the line is busy, if the party does not answer, or if the user specified an unrecognized number. If any of these conditions exist, a message is displayed.

If a busy signal is received while trying to connect to a remote workstation, press the PREVIOUS_KEY (usually the Ctrl−R key sequence), and enter the *TelephoneNumber* parameter again.

Once the connection is established, ATE displays a message indicating the name of the port used for the connection.

Parameters

PortName	Specifies the name of the port used for a direct connection.
TelephoneNumber	Specifies the telephone number used to establish a modem connection.

Examples

1. To establish a direct connection, at the command line of the ATE Unconnected Main Menu, enter:

   ```
   c tty0
   ```

   This command establishes a direct connection using port `tty0`. After connection is established, a message displays, followed by a login screen. Enter the requested login information and press the MAINMENU_KEY (usually the Ctrl−V key sequence) to display the ATE Connected Main Menu.

2. To establish a manually dialed connection, at the command line of the ATE Unconnected Main Menu, enter:

   ```
   c
   ```

   The ATE program prompts the user for information necessary to establish a manually dialed connection, such as a telephone number or modem to use. After connection is established, ATE displays a message giving the port name used for the connection, followed by a login screen. Enter the requested login information and press the MAINMENU_KEY (usually the Ctrl−V key sequence) to display the ATE Connected Main Menu.

3. To establish an automatically dialed connection, at the command line of the ATE Unconnected Main Menu, enter:

   ```
   c  2229999
   ```

This example dials the telephone number 222−9999. After connection is established, a message displays indicating the port used for the connection, followed by a login screen. Enter the requested login information and press the MAINMENU_KEY (usually the Ctrl−V key sequence) to display the ATE Connected Main Menu.

directory Subcommand

d

The ATE **directory** subcommand displays a dialing directory. Users establish a connection to a remote computer by selecting one of the directory entries from the displayed directory. The **directory** subcommand is issued from the ATE Unconnected Main Menu. The **directory** subcommand uses the information contained in the dialing directory to establish an automatically dialed (modem−dialed) connection.

When ATE starts, it checks the current directory for an **ate.def** file format. If an **ate.def** file format does not exist in the current directory, it creates one. The initial location of the dialing directory is **/usr/lib/dir**, but this value can be changed by editing the **ate.def** file format. If users specify a different dialing directory in the **ate.def** file format, that directory is used.

The dialing directory contains entries for remote systems called with the ATE program in the format:

NamePhoneRateLengthStopBitParityEchoLinefeed

These fields give the name of the entry (usually the person or company whose computer the phone number reaches), the telephone number, and other information the ATE program uses to establish the connection.

See "Dialing Directory File Format for ATE" in *AIX Files Reference* for more information about dialing directory entries.

When an entry displays on the screen using the **directory** subcommand, the entry is preceded by an entry number. Select the entry to establish a connection to by entering its entry number in response to a prompt.

Example

To display a dialing directory, at the command line of the Unconnected Main Menu, enter:

```
d
```

The dialing directory specified in the **ate.def** file format displays and prompts the user for an entry number. Enter the number of the dialing directory entry to establish a connection with. ATE establishes the connection and displays a message indicating the port name used.

See How to Set up an ATE Dialing Directory in *AIX Version 4.3 System User's Guide: Communications and Networks*.

help Subcommand

h [**a**] [**b**] [**c**] [**d**] [**m**] [**p**] [**q**] [**r**] [**s**] [**t**]

The ATE **help** subcommand provides help information for the ATE subcommands. Issue the **help** subcommand from either the Unconnected or Connected Main Menu of ATE. Help information is

available for all the ATE subcommands, and can be requested for several subcommands at the same time.

When issuing the **help** subcommand, ATE displays a description of each subcommand requested and instructions for using the subcommand. Help information for each subcommand displays individually, in the order requested. After reading each help message, press Enter to view the next page of help text. At the end of the help text, press Enter to return to the main menu.

Issue the **help** subcommand with the first letter of an ATE subcommand for help information. These are the names for the ATE subcommands:

Name	ATE Subcommand
a	**alter** subcommand
b	**break** subcommand
c	**connect** subcommand
d	**directory** subcommand
m	**modify** subcommand
p	**perform** subcommand
q	**quit** subcommand
r	**receive** subcommand
s	**send** subcommand
t	**terminate** subcommand

Examples

1. To receive help information for a single subcommand, enter the following at one of the ATE main menus:

   ```
   h c
   ```

   Help information displays for the **connect** (`c`) subcommand. After viewing the help information, press the Enter key, and ATE displays the menu from which the **help** subcommand was issued.

2. To receive help information for multiple subcommands, enter the following at one of the ATE main menus:

   ```
   h r s
   ```

The help information for the **receive** subcommand (`r`) displays first. After viewing the help information, press the Enter key. Help information for the **send** subcommand (`s`) displays. After viewing the help information, press the Enter key, and ATE displays the menu from which the **help** subcommand was issued.

modify Subcommand

m [**n** *CaptureFileName*] [**e**] [**l**] [**v**] [**w**] [**x**]

> **Note:** The default *CaptureFileName* and the initial settings of the other **modify** subcommand flags can be permanently changed in the **ate.def** file format.

The **modify** subcommand is accessed from the Asynchronous Terminal Emulation (ATE) Connected or Unconnected Main Menu. The **modify** subcommand temporarily changes how ATE functions on the local system in the following ways:

- Changes the name of the capture file that receives incoming data.
- Switches (toggles) the following features on or off:
  - ♦ Add a line−feed character at the end of each line of incoming data.

- Use echo mode.
- Emulate a DEC VT100 terminal at the console.
- Write incoming data to a capture file as well as to the display.
- Use an **Xon/Xoff** (transmitter on/off) signal.

The settings return to the default values as defined in the **ate.def** file format when the user exits ATE.

When issued without flags from either of the ATE main menus, the **modify** subcommand displays the Modify Menu. The Modify Menu can be bypassed by entering **m** (the **modify** subcommand abbreviation), followed by the appropriate flags, at the command prompt on either ATE main menu.

The **modify** subcommand can change more than one feature at a time. To change the **name** variable, enter the **n** flag followed by the new file name. All other variables are switches that can be turned on or off by typing the flag. Typing the flag switches, or toggles, the value.

To permanently change the settings affected by the **modify** subcommand, customize the **ate.def** file format in the directory running ATE.

Modify Menu

The Modify Menu displays the current settings of the features changeable with the **modify** subcommand. To display the Modify Menu, enter the letter **m** after the command prompt on either the ATE Connected Main Menu or the ATE Unconnected Main Menu.

The Modify Menu contains the following columns:

Column Names	Contents
COMMAND	Flag to enter to change a value
DESCRIPTION	Description of the variable the flag affects
CURRENT	Current value of the variable
POSSIBLE CHOICES	Possible values of the variable

To change the value of a flag other than the **name** flag, enter the flag (from the COMMAND column) at the command prompt on the Modify Menu. The flag value toggles to the alternate setting. To change the name of the capture file, enter the letter **n** (the **name** flag), followed by the new file name, at the prompt on the Modify Menu.

To return to the ATE Connected or Unconnected Main Menu from the Modify Menu, press the Enter key.

Flags

e **echo**

Displays the input typed by the user.

With a remote computer that supports echoing, each character sent returns and displays on the screen. When the **echo** flag is on, each character is displayed twice: first when it is entered and again when it returns over a connection. When the **echo** flag is off, each character displays only when it returns over the connection.

Options: On or off

Default: Off

l **linefeed**

Adds a line−feed character after every carriage−return character in the incoming data stream.

Options: On or off

Default: Off

n *CaptureFileName* **name**

Specifies the file name for incoming data when the **write** flag is on, or when the CAPTURE_KEY (usually the Ctrl−B key sequence) is pressed during a connection.

Options: Any valid file name. The first 18 characters display in the Modify Menu.

Default: **capture**

v **VT100**

The local console emulates a DEC VT100 terminal so DEC VT100 codes can be used with the remote system. With the **VT100** flag off, the local console functions like a workstation.

Options: On or off

Default: Off

> **Note:** No keys on the console keyboard are remapped. In addition, some DEC VT100 codes, such as 132 columns, double−height and double−width lines, origin mode, and graphics characters generated from a 10−key keypad, are not supported.

w **write**

Routes incoming data to the capture file (specified by the **name** flag) as well as to the display. The **write** command functions like the CAPTURE_KEY key sequence during a connection. Carriage return and line−feed combinations are converted to line−feed characters before being written to the capture file. In an existing file, data is appended to the end of the file.

Options: On or off

Default: Off

x **Xon/Xoff**

Controls data transmission at a port using the **Xon/Xoff** protocol, as follows:

- When an **Xoff** signal is received, transmission stops.
- When an **Xon** signal is received, transmission resumes.
- An **Xoff** signal is sent when the receive buffer is nearly full.
- An **Xon** signal is sent when the buffer is no longer full.

Options: On or off

Default: On

Note: If you use a variable value with any flag other than the **name** flag, the following error message displays:

```
828-003 not 'command-name' command is not valid.
Enter the first letter of a command
from the list on the menu.
```

This error message indicates either an incorrect letter was entered or a value that is not valid was included.

Examples

1. To display the Modify Menu, enter the **modify** subcommand at the command prompt on either ATE main menu:

   ```
   m
   ```

   The Modify Menu displays.

2. To modify settings from the Modify Menu, enter the appropriate flag at the command prompt at the bottom of the Modify Menu:
   - To toggle the values of the **linefeed** flag, at the prompt on the Modify Menu enter:

     ```
     l
     ```

   The value of the **linefeed** flag is switched to the alternate setting.
3. To change the **name** variable to `schedule`, at the prompt on the Modify Menu enter:

   ```
   n schedule
   ```

   Any data saved is now put into the `schedule` file.

4. To bypass the Modify menu when using the **modify** subcommand, type the **m** subcommand (the **modify** subcommand abbreviation), followed by the appropriate flags, at the command prompt on either ATE main menu:
   - To toggle the values of the **linefeed** and **echo** flags, at the prompt on either ATE main menu enter:

     ```
     m l e
     ```

   The values of the **linefeed** and **echo** flags are switched to the alternate settings. Display the Modify Menu to view the current settings of the flags.
5. To change the **name** variable to `schedule` and toggle the values of the **write** and **Xon/Xoff** flags, at the prompt on either ATE main menu enter:

   ```
   m n schedule w X
   ```

Any data saved is now put into the `schedule` file, and the values of the **write** and **Xon/Xoff** flags are switched to the alternate settings. Display the Modify Menu to view the settings of the flags.

perform Subcommand

p [*Command*]

The ATE **perform** subcommand allows the user to issue workstation operating system commands while using Asynchronous Terminal Emulation (ATE). Issue the **perform** subcommand from the ATE Unconnected or Connected Main Menu. *Command* specifies a valid workstation operating system command.

Examples

1. To issue a workstation operating system command, at the command line of the ATE Unconnected or Connected Main Menu, enter:

   ```
   p
   ```

   ATE prompts the user to enter a command. ATE executes the specified command. After the command finishes, ATE displays the menu from which the **perform** subcommand was issued.

2. To specify the command to be executed, at the command line of the ATE Unconnected or Connected Main Menu, enter:

   ```
   p cat mystuff
   ```

ATE executes the **cat** command, which displays the `mystuff` file. After the **cat** command finishes, ATE displays the menu from which the **perform** subcommand was issued.

quit Subcommand

q

The ATE **quit** subcommand exits the Asynchronous Terminal Emulation (ATE) program. Issue the **quit** subcommand from the ATE Unconnected or Connected Main Menu. Issuing the **quit** subcommand ends the ATE program and displays the command prompt.

Example

To exit the ATE program, from the command line of either ATE main menu, enter:

```
q
```

The ATE program ends and the command prompt displays.

receive Subcommand

r*FileName*

The ATE **receive** subcommand enables your system to receive a file from a remote system. The ATE **receive** subcommand is issued from the ATE Connected Main Menu.

The ATE **receive** subcommand uses the **xmodem** file transfer protocol, which enables your system to receive data from a remote system, a block at a time, with error checking. The remote system must be set to send the file before your system can receive. Use the **xmodem** command with the **−s** flag on the remote system to enable the remote system to send the file. Then issue the **receive** subcommand. *FileName* names the file where the received data is stored.

Example

To receive a file sent from the remote system, at the command line of the ATE Connected Main Menu, enter:

```
r myfile
```

The data is received from the remote system and is stored in the `myfile` file.

send Subcommand

s [*FileName*]

The ATE **send** subcommand sends a file to a remote system. Issue the ATE **send** subcommand from the ATE Connected Main Menu once a connection is established. The ATE **connect** subcommand establishes the connection and prepares the remote system to receive files.

The **send** subcommand uses the **xmodem** file transfer protocol, sending data to a remote system, a block at a time, with error checking. Issue the **xmodem** command with the **−r** flag on the remote system to enable the remote system to receive the file. Then issue the **send** subcommand. *FileName* names the file to send to the remote system.

Examples

1. To send a file to a remote system, at the command line of the ATE Connected Main Menu, enter:

   ```
   s
   ```

   ATE prompts the user for the name of the file to send to the remote system.

2. To specify a file to send to the remote system, at the command line of the ATE Connected Main Menu, enter:

   ```
   s mystuff
   ```

The `mystuff` file is sent to the remote system.

terminate Subcommand

t

The ATE **terminate** subcommand ends an Asynchronous Terminal Emulation (ATE) connection to a remote system and returns to the ATE Unconnected Main Menu. Issue the **terminate** subcommand from the ATE Connected Main Menu.

Example

To terminate the current session, from the remote system login screen, press the MAINMENU_KEY (usually the Ctrl−V key sequence). When the ATE Connected Main Menu displays, enter:

```
t
```

A terminate signal is sent to the remote system, the session ends, and ATE displays the Unconnected Main Menu. Now issue other ATE subcommands or exit ATE.

File

/usr/lib/dir Contains the default dialing directory.

Related Information

The **ate.def** file format contains ATE default values.

ATE Overview in *AIX Version 4.3 System User's Guide: Communications and Networks* describes the ATE program, its menus, and its control keys.

How to Edit the ATE Default File in *AIX Version 4.3 System User's Guide: Communications and Networks* explains how to permanently change ATE defaults.

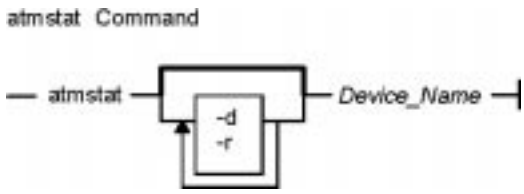
ATE Overview for System Management in *AIX Version 4.3 System User's Guide: Communications and Networks* discusses tasks involved in managing ATE and lists the aspects of ATE that can be customized.

atmstat Command

Purpose

Shows Asynchronous Transfer Mode adapters statistics.

Syntax



atmstat [**−d−r**] *Device_Name*

Description

The **atmstat** command displays Asynchronous Transfer Mode (ATM) adapter statistics. The user can optionally specify that the device−specific statistics be displayed in addition to the device generic statistics. If no flags are specified, only the device generic statistics are displayed. For information on statistic from the **atmstat** command, see ATM Adapter Statistics in the *AIX Version 4.3 System User's Guide: Communications and Networks*.

If an invalid *Device_Name* is specified, the **atmstat** command produces an error message stating that it could not connect to the device.

Flags

−d Displays detailed statistics.

−r Resets all the statistics back to their initial values. This flag can only be issued by privileged users.

Parameters

Device_Name The name of the ATM device, for example, **atm0**.

Examples

To display the adapter generic statistics for **atm0**, enter:

```
atmstat atm0
```

This produces the following output on a microchannel machine:

```
ATM STATISTICS (atm0) :
Device Type: Turboways 155 MCA ATM Adapter
Hardware Address: 08:00:5a:99:88:d5
Elapsed Time: 2 days 23 hours 38 minutes 18 seconds

Transmit Statistics:                          Receive Statistics:
--------------------                          -------------------
```

```
Packets: 50573                                  Packets: 0
Bytes: 2225182                                  Bytes: 0
Interrupts: 0                                   Interrupts: 12904
Transmit Errors: 0                              Receive Errors: 0
Packets Dropped: 0                              Packets Dropped: 0
                                                Bad Packets: 0
Max Packets on S/W Transmit Queue: 0
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 0

Cells Transmitted: 50573                        Cells Received: 0
Out of Xmit Buffers: 0                          Out of Rcv Buffers: 0
Current HW Transmit Queue Length: 0             CRC Errors: 0
Current SW Transmit Queue Length: 0             Packets Too Long: 0
                                                Incomplete Packets: 0
                                                  Cells Dropped: 0

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Loss of Signals: 0
Adapter Reset Count: 0
Driver Flags: Up Running Simplex
        64BitSupport
Virtual Connections in use: 2
Max Virtual Connections in use: 2
Virtual Connections Overflow: 0
SVC UNI Version: auto_detect

Turboways ATM Adapter Specific Statistics:
------------------------------------------------
Packets Dropped - No small DMA buffer: 0
Packets Dropped - No medium DMA buffer: 0
Packets Dropped - No large DMA buffer: 0
Receive Aborted - No Adapter Receive Buffer: 0
Transmit Attempted - No small DMA buffer: 0
Transmit Attempted - No medium DMA buffer: 0
Transmit Attempted - No large DMA buffer: 0
Transmit Attempted - No MTB DMA buffer: 0
Transmit Attempted - No Adapter Transmit Buffer: 0
Max Hardware transmit queue length: 12
Small Mbuf in Use: 0
Medium Mbuf in Use: 0
Large Mbuf in Use: 64
Huge Mbuf in Use: 0
MTB Mbuf in Use: 0
Max Small Mbuf in Use: 0
Max Medium Mbuf in Use: 0
Max Large Mbuf in Use: 64
Max Huge Mbuf in Use: 0
MTB Mbuf in Use: 0
Small Mbuf overflow: 0
Medium Mbuf overflow: 0
Large Mbuf overflow: 0
Huge Mbuf overflow: 0
MTB Mbuf overflow: 0
```

This produces the following output on a PCI machine:

```
--------------------                            ------------------
Packets: 299                                    Packets: 294
Bytes: 9727                                     Bytes: 10123
Interrupts: 0                                   Interrupts: 297
Transmit Errors: 0                              Receive Errors: 0
Packets Dropped: 0                              Packets Dropped: 0
                                                Bad Packets: 0
Max Packets on S/W Transmit Queue: 0
```

```
S/W Transmit Queue Overflow: 0
Current S/W+H/W Transmit Queue Length: 2

Cells Transmitted: 450                      Cells Received: 457
Out of Xmit Buffers: 0                      Out of Rcv Buffers: 0
Current HW Transmit Queue Length: 2         CRC Errors: 0
Current SW Transmit Queue Length: 0         Packets Too Long: 0
                                            Incomplete Packets: 0
                                              Cells Dropped: 5

General Statistics:
-------------------
No mbuf Errors: 0
Adapter Loss of Signals: 0
Adapter Reset Count: 0
Driver Flags: Up Running Simplex
        64BitSupport
Virtual Connections in use: 4
Max Virtual Connections in use: 5
Virtual Connections Overflow: 0
SVC UNI Version: uni3.1

IBM PCI 155 Mbps ATM Adapter Specific Statistics:
-------------------------------------------------
Total 4K byte Receive Buffers: 96   Using: 64
```

Related Information

The **entstat** command, **fddistat** command, **netstat** command, **tokstat** command.

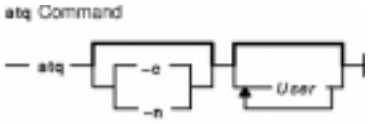
ATM Adapter Statistics in the *AIX Version 4.3 System User's Guide: Communications and Networks*.

atq Command

Purpose

Displays the queue of jobs waiting to be run.

Syntax



atq [**c** | **−n**] [*User* ...]

Description

The **atq** command displays the current user's queue of jobs that are waiting to be run at a later date, sorted in the order the jobs will be run. These jobs were created with the **at** command. If the user is root and *User* name is specified, the **atq** command displays only jobs belonging to that user.

Flags

−c Sorts the queue by the time that the **at** command was issued.
−n Displays only the number of jobs currently in the queue.

Examples

In order to look at the queue created by the **at** command, enter:

```
atq
```

If there are jobs in the queue, a message similar to the following appears:

```
root.635623200.a          Wed     Feb 21  12:00:00 1990
root.635670000.a          Thu     Feb 22  01:00:00 1990
```

Files

/usr/bin/atq Contains the **atq** program.
/var/spool/cron/atjobs Specifies the spool area.

Related Information

The **at** command, **atrm** command.

The **cron** daemon.

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output.

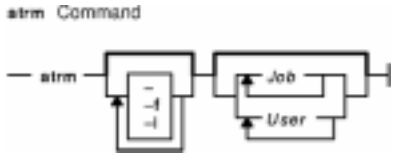
Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes what shells are, the different types of shells, and how shells affect the way commands are interpreted.

atrm Command

Purpose

Remove jobs spooled by the **at** command.

Syntax



atrm [**−f**] [**−i**] [**−**] [*Job* ... | *User* ...]

Description

The **atrm** command removes jobs that were created with the **at** command. If one or more job numbers is specified, the **atrm** command 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 the **atrm** command is useful only if you have root user authority.

Flags

− Removes all jobs belonging to the user invoking the **atrm** command.
−f Suppresses all information about the jobs being removed.
−i Prompts before a job is removed. Enter `y` to remove the job.

Examples

To remove job number `root.62169200.a` from the **at** command queue, enter:

```
atrm root.621619200.a
```

Files

/usr/bin/atrm Contains the **atrm** program file.
/var/spool/cron/atjobs Specifies the spool area.

Related Information

The **at** command, **atq** command.

The **cron** daemon.

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output.

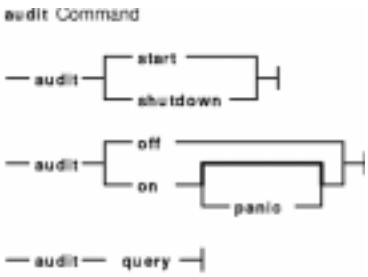
Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes what shells are, the different types of shells, and how shells affect the way commands are interpreted.

audit Command

Purpose

Controls system auditing.

Syntax



audit { **start** | **shutdown** }

audit { **off** | **on** [**panic**] }

audit query

Description

The **audit** command controls system auditing through its several keywords. One keyword must be included each time the command is given. The **start** keyword and the **shutdown** keyword start and stop the auditing system and reset the system configuration. The **off** keyword and the **on** keyword suspend and restart the audit system without affecting the system configuration. The **query** keyword lets you query the current status.

The auditing system follows the instructions established in the following configuration files:

- **/etc/security/audit/config**
- **/etc/security/audit/events**
- **/etc/security/audit/objects**
- **/etc/security/audit/bincmds**
- **/etc/security/audit/streamcmds**

Each of these files is described in "Files" section . For information on configuring the audit system, see "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

Keywords

start Starts the audit subsystem. The command reads the instructions in the configuration files and does the following:

object auditing
Writes the audit event definitions in the **/etc/security/audit/objects** file into the kernel to define the object auditing events.

event auditing
Writes the audit class definitions in the **/etc/security/audit/config** file into the kernel to define the audit

classes.

bin auditing
Starts the **auditbin** daemon according to the configuration information in the bin stanza in the **/etc/security/audit/config** file, if the start stanza contains `binmode=on`.

stream auditing
Invokes the audit stream commands as defined in the stream stanza in the **/etc/security/audit/config** file, if the start stanza contains `streammode=on`.

Attention: Invocation of stream auditing from **/etc/inittab** should be avoided.

user auditing
Audits all users currently logged in to the system, if they are configured in the users stanza of the **/etc/security/audit/config** file.

audit logging
Enables the audit logging component as defined in the start stanza in the **/etc/security/audit/config** file.

shutdown
Terminates the collection of audit records and resets the configuration information by removing the definition of classes from the kernel tables. All the audit records are flushed from the kernel buffers into the bin files or audit streams, according to the specifications for the backend commands, which are contained in the **/etc/security/audit/bincmds** file for binmode auditing, and in the **/etc/security/audit/streamcmds** file for streammode auditing. The collection of audit data stops until the next **audit start** command is given.

off
Suspends the auditing system, but leaves the configuration valid. Data collection pauses until the **audit on** command is given.

on [**panic**]
Restarts the auditing system after a suspension, if the system is properly configured (for example, if the **audit start** command was used initially and the configuration is still valid). If auditing is already started when the command is given, only bin data collection can be changed.

If you specify the **panic** option, the system will shut down if bin data collection is enabled but cannot be written to a bin file. If binmode is not enabled, the system will shut down.

query
Displays the current status of the audit subsystem, in the following format:

```
auditing on {panic} | auditing off

bin manager off |  is process number pid

audit events:
    audit class: audit event, audit event...
audit objects:
    object name: object mode: audit event
```

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be **setuid** to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/security/audit/config**
r	**/etc/security/audit/objects**
x	**/usr/sbin/auditbin**
x	**/usr/sbin/auditstream**

Examples

1. To start the audit process, configure the audit system as described in "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices*, and add the following line to the system initialization file (the **/etc/rc** file):

   ```
   /usr/sbin/audit start
   ```

   The audit process starts, as configured, each time the system is initialized.

2. To terminate the operation of the auditing process, enter:

   ```
   /usr/sbin/audit shutdown
   ```

   Data collection stops until the **audit start** command is given again. The configuration of classes in the operating system kernel is lost.

   Note: The **audit shutdown** command should be in the **/etc/shutdown** file as well.

3. To suspend the audit subsystem, enter:

   ```
   /usr/sbin/audit off
   ```

4. To restart an audit process that was suspended by the **audit off** command, enter:

   ```
   /usr/sbin/audit on
   ```

   The suspended state ends and audit records are generated again, as long as the system is configured correctly.

5. To display the current status of the auditing system, enter:

   ```
   /usr/sbin/audit query
   ```

   An example of an **audit query** status message follows:

   ```
   auditing on

   bin manager is process number 123

   audit events:
       authentication- USER_Login, USER_Logout
       administration- USER_Create, GROUP_Create

   audit objects:
       /etc/security/passwd :
         r = AUTH_Read
       /etc/security/passwd :
         w = AUTH_Write
   ```

The query tells you that audit records will be written when the specified users log in or log out, when the specified administrators create a user or a group, and when the system receives an authorized read or write instruction for the **/etc/security/passwd** file.

Files

/usr/sbin/audit	Contains the path of the **audit** command.
/etc/rc	Contains the system initialization commands.
/etc/security/audit/config	Contains audit configuration information.
/etc/security/audit/events	Lists the audit events and their tail format specifications.
/etc/security/audit/objects	Lists the audit events for each file (object).
/etc/security/audit/bincmds	Contains shell commands for processing audit bin data.
/etc/security/audit/streamcmds	Contains auditstream commands.

Related Information

The **auditbin** daemon, **auditcat** command, **auditconv** command, **auditpr** command, **auditselect** command, **auditstream** command, **login** command, **logout** command, **su** command.

The **audit** subroutine, **auditbin** subroutine, **auditevents** subroutine, **auditlog** subroutine, **auditproc** subroutine.

For general information on auditing, refer to Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

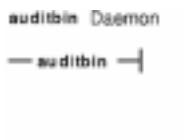
To see the steps you must take to establish an Auditing System, refer to Setting up Auditing in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

auditbin Daemon

Purpose

Manages bins of audit information.

Syntax



auditbin

Description

The **auditbin** daemon in the audit subsystem manages **bin1** and **bin2**, temporary bin files that alternately collect audit event data. The command also delivers bins of data records to backend commands for processing.

As audit events occur, the operating system kernel writes a record to a bin file. When a bin file is full, the **auditbin** daemon reads the **/etc/security/audit/bincmds** file and delivers the bin records to the backend commands defined in the file. Each line of the **/etc/security/audit/bincmds** file contains one or more commands with input and output that can be piped together or redirected. The **auditbin** daemon searches each command for the **$bin** string and the **$trail** string and substitutes the path names of the current bin file and the system trail file for these strings.

The **auditbin** daemon ensures that each command encounters each bin at least once, but does not synchronize access to the bins. When all the commands have run, the bin file is ready to collect more audit records.

If a command is unsuccessful, the **auditbin** daemon stops delivering data records and sends a message to the **/dev/tty** device every 60 seconds until the root user or a member of the audit group stops the command.

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be **setuid** to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/security/audit/config**
r	**/etc/security/audit/bincmds**
rw	Defined audit bins and trail file
x	All audit bin processing commands

Examples

1. To configure the **auditbin** daemon, edit the start and bin stanzas of the **/etc/security/audit/config** file to include the following attribute definitions:

```
start:
         binmode = on

bin:
         trail = /audit/trail
         bin1 = /audit/bin1
         bin2 = /audit/bin2
         binsize = 25000
         cmds = /etc/security/audit/bincmds
```

2. To define the commands that process the audit trail, edit the **/etc/security/audit/bincmds** file to include one or more command lines, such as the following:

```
/usr/sbin/auditcat −p −o $trail $bin

/usr/sbin/auditselect −e "event == USER_Login" \
$bin | /usr/sbin/auditpr >> /etc/log
```

The first command line appends compressed audit bins to the audit trail file. The second line selects `USER_Login` records from each bin file, passes them to the `auditpr` command for formatting, and appends the records to the `/etc/log` file.

Files

/usr/sbin/auditbin	Specifies the path to the **auditbin** daemon.
/audit/bin*x*	Specifies the path to the default bin collection files, with *x* indicating the bin number.
/etc/security/audit/config	Contains audit system configuration information.
/etc/security/audit/events	Contains the audit events of the system.
/etc/security/audit/objects	Contains audit events for audited objects (files).
/etc/security/audit/bincmds	Contains the **auditbin** backend commands.
/etc/security/audit/streamcmds	Contains the **auditstream** commands.

Related Information

The **audit** command, **auditcat** command, **auditconv** command,**auditpr** command, **auditselect** command, **auditstream** command.

The **audit** subroutine, **auditbin** subroutine.

Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

To see the steps you must take to establish an Auditing System, refer to Setting up Auditing in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

auditcat Command

Purpose

Writes bins of audit records.

Syntax

auditcat [**−p** | **−u**] [**−o** *OutFile*] [**−r**] [*InFile*]

Description

The **auditcat** command is part of the audit subsystem, and is one of several backend commands that process the audit data records.

The **auditcat** command reads bin files of audit records from standard input or from the file specified by the *InFile* parameter. The command then processes the records and writes its output to standard output or to the file specified by the *0utFile* parameter. The output can be compressed or not, depending on the flag selected.

One major use of the command is appending compressed bin files to the end of the system audit trail file.

If the **/etc/security/audit/bincmds** file includes **$bin** as the input file, input comes from the current bin file, **bin1** or **bin2**. If the **/etc/security/audit/bincmds** file includes **$trail** as the output file, the records are written to the end of the system audit trail file.

If a bin file is not properly formed with a valid header and tail, an error is returned. See the **auditpr** command for information about audit headers and tails and the **auditbin** command for information on error recovery.

Flags

−o *OutFile* Specifies the audit trail file to which the **auditcat** command writes records. If you specify **$trail** as the file for the *OutFile* parameter, the **auditbin** daemon substitutes the name of the system audit trail file.

−p Specifies that the bin files be compressed (packed) upon output. The default value specifies that the bins not be compressed.

−r Requests recovery procedures. File names for both the *InFile* and *OutFile* parameters must be specified for recovery to occur, so the command syntax must be **auditcat −o** *OutFile* **−r** *InFile*. The command checks to see if the bin file specified for the *InFile* parameter is appended and if not, appends the bin file to the file specified by the *OutFile* parameter. If the bin file is incomplete, the **auditcat** command adds a valid tail and then appends the bin file to the file specified by the *OutFile* parameter.

−u Specifies that compressed trail files be uncompressed upon output.

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be setuid to the root user and have the **trusted computing base** attribute.

Examples

To configure the system to append audit bin data to the system audit trail file, add the following line to the **/etc/security/audit/bincmds** file:

```
/usr/sbin/auditcat –o $trail $bin
```

When the **auditbin** daemon calls the **auditcat** command, the daemon replaces the **$bin** string with the path name of the current bin file, and replaces the **$trail** string with the name of the default audit trail file.

Files

/usr/sbin/auditcat	Specifies the path to the **auditcat** command.
/etc/security/audit/config	Contains audit system configuration information.
/etc/security/audit/events	Contains the audit events of the system.
/etc/security/audit/objects	Contains audit events for audited objects (files).
/etc/security/audit/bincmds	Contains auditbin backend commands.

Related Information

The **audit** command, **auditconv** command, **auditpr** command, **auditselect** command.

auditbin daemon.

For general information on auditing, refer to Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

To see the steps you must take to establish an Auditing System, refer to Setting up Auditing in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

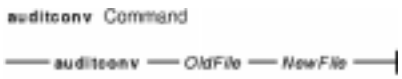
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

auditconv Command

Purpose

Converts pre−AIX version 4 format audit bins to AIX version 4 format.

Syntax



auditconv*OldFileNewFile*

Description

The **auditconv** command converts audit records which were generated by previous versions of the operating system into the format used by versions 4 and higher of the operating system.

Audit records are read from the file *OldFile*, and written to the file *NewFile*. Each audit record is updated with thread information, with a default thread identifier of zero.

Notes:

1. The *OldFile* and *NewFile* parameters must be different, and must not be currently in use by the audit system.
2. Versions 4.0 and higher of the operating system cannot work with pre−version 4 audit bins. Therefore, old bins must be converted using the **auditconv** command.

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be setuid to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/security/audit/events**
r	**/etc/passwd**
r	**/etc/group**

Example

To convert the old audit file **pre_v4_auditbin**, storing the results in **converted_auditbin**, enter the following command:

```
/usr/sbin/auditconv pre_v4_auditbin converted_auditbin
```

Files

/usr/sbin/auditconv	Specifies the path of the **auditconv** command.
/etc/security/audit/config	Contains audit system configuration information.
/etc/security/audit/events	Contains the audit events of the system.
/etc/security/audit/objects	Contains information about audited objects (files).
/etc/security/audit/bincmds	Contains auditbin backend commands.
/etc/security/audit/streamcmds	Contains auditstream commands.

Related Information

The **audit** command, **auditbin** daemon, **auditcat** command, **auditpr** command, **auditselect** command, **auditstream** command.

The **audit** subroutine.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

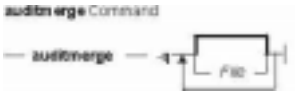
To see the steps you must take to establish an Auditing System, refer to Setting up Auditing in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

auditmerge Command

Purpose

Combines multiple audit trails into a single trail.

Syntax



/usr/sbin/auditmerge [**−q**] *file* [*file* ...]

Description

The **auditmerge** command combines multiple audit trail files from potentially multiple machines into a single audit trail file. For each file with records remaining, the record that has the oldest time stamp is added to the output. If a record is found that has a negative time change, an optional warning message may be emitted. Processing continues and any such records are output with their time values unmodified.

The **auditmerge** command also is capable of adding CPU ID values from the bin header to each output record. The CPU ID value is encoded in the bin header and trailer for bins with a version number more recent than AIX Version 4.3.1.

The **−q** flag is used to control outputting warning messages. When a record with a negative time change is first seen, a single warning message is output. That message contains the name of the file containing the record and the time difference. These messages are suppressed when the **−q** flag is given on the command line.

Flags

−q Used to control outputting warning messages.

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be setuid to the root user and have the **trusted computing base** attribute.

Examples

1. To merge two existing audit trails files from different hosts, enter:
   ```
   /usr/bin/auditmerge /audit/trail.calvin /audit/trail.hobbes > /audit/trail.merge
   ```

2. To merge two existing data files which were preselected for different user names, enter:
   ```
   /usr/bin/auditmerge /audit/trail.jim /audit/trail.julie > /audit/trail.both
   ```

3. To merge two data files without producing warnings about incorrect times, enter:
   ```
   /usr/bin/auditmerge −q /audit/jumbled.1 /audit/jumbled.2 > /audit/jumbled.output
   ```

Files

/etc/security/audit/hosts Contains the CPU ID to hostname mappings.

Related Information

The **auditpr** command, **auditstream** command, **auditselect** command.

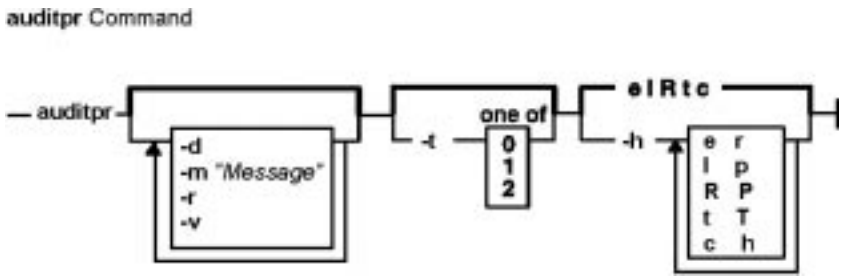
The **auditread** subroutine, **getaudithostattr** subroutine, **setaudithostdb** subroutine.

auditpr Command

Purpose

Formats bin or stream audit records to a display device or printer.

Syntax



auditpr [**−m** "*Message*"] [**−t** { **0** | **1** | **2** }] [**−h** { **e l R t c r p P T h** }] [**−r**] [**−v**]

Description

The **auditpr** command is part of the audit subsystem. This command reads audit records, in bin or stream format, from standard input and sends formatted records to standard output.

The output format is determined by the flags that are selected. If you specify the **−m** flag, a message is displayed before each heading. Use the **−t** and **−h** flags to change the default header titles and fields and the **−v** flag to append an audit trail. The **auditpr** command searches the local **/etc/passwd** file to convert user and group IDs to names.

An example of output using default header information follows:

```
event   login   status   time                                command
login   dick     OK       Fri  Feb;8   14:03:57    1990   login
 . . . . . tail portion . . . . .
```

For examples of audit tails, see the **/etc/security/audit/events** file where audit tail formats are defined.

Invalid records are skipped when possible, and an error message is issued. If the command cannot recover from an error, processing stops.

Flags

−d
Outputs data in microsecond resolutions. The output of the date in that format is:
`DD MMM YYYY hh:mm:ss.uuuuuu`

−h *Fields*
Selects the fields to display and the order in which to display them, by default **e**, **l, R**, **t**, and **c**. The legal values are:

- **e** The audit event
- **l** The user's login name
- **R** The audit status
- **t** The time the record was written

c The command name

r The real user name

p The process ID

P The ID of the parent process.

T The kernel thread ID. This is local to the process; different processes may contain threads with the same thread ID.

h The name of the host that generated the audit record. If there is no CPU ID in the audit record, the value **none** is used. If there is no matching entry for the CPU ID in the audit record, the 16 character value for the CPU ID is used instead.

−m "*Message*" Specifies a *Message* to be displayed with each heading. You must enclose the *Message* string in double quotation marks.

−r Suppresses ID translation to the symbolic name.

−t {**0** | **1** | **2**} Specifies when header titles are displayed. The default title consists of an optional message (see the **−m** flag) followed by the name of each column of output.

0 Ignores any title.

1 Displays a title once at the beginning of a series of records.

2 Displays a title before each record.

−v Displays the tail of each audit record, using the format specifications in the **/etc/security/audit/events** file.

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be **setuid** to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File
r	**/etc/security/audit/events**
r	**/etc/passwd**
r	**/etc/group**

Examples

1. To read the system audit tail file with default header titles and fields and an audit tail, enter:

   ```
   /usr/sbin/auditpr −v < /audit/trail
   ```

   The **/audit/trail** file must contain valid audit bins or records.

2. To format from an audit trail file all the audit events caused by user `witte`, enter:

   ```
   /usr/sbin/auditselect −e"login == witte"\
   /audit/trail | auditpr −v
   ```

   The resulting record is formatted with the default values (**e**, **c**, **l**, **R**, and **t**) and includes a tail.

3. To read records interactively from the audit device, enter:
   ```
   /usr/sbin/auditstream | /usr/sbin/auditpr −t0 −heRl
   ```

Files

/usr/sbin/auditpr	Specifies the path of the **auditpr** command.
/etc/security/audit/config	Contains audit system configuration information.
/etc/security/audit/events	Contains the audit events of the system.
/etc/security/audit/objects	Contains audit events for audited objects (files).
/etc/security/audit/bincmds	Contains auditbin backend commands.
/etc/security/audit/streamcmds	Contains auditstream commands.
/etc/security/audit/hosts	Contains the CPU ID to host name mappings.

Related Information

The **audit** command, **auditcat** command, **auditconv** command, **auditselect** command, **auditstream** command.

The **auditbin** daemon.

The **audit** subroutine.

The **events** file.

For general information on auditing, refer to Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

To see the steps you must take to establish an Auditing System, refer to Setting up Auditing in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

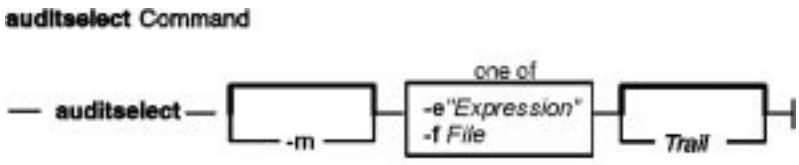
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

auditselect Command

Purpose

Selects audit records for analysis according to defined criteria.

Syntax



auditselect {**−e** "*Expression*" | **−f***File*} [*Trail*]

Description

The **auditselect** command is part of the audit subsystem. The command is called by the **auditbin** daemon if it is configured in the **/etc/security/audit/bincmds** file as a backend command for processing bin files.

The **auditselect** command selects audit records that match identified criteria and writes the records to standard output. With the **auditselect** command, you can filter the audit trail to obtain specific records for analysis or select specific records for long−term storage. The command takes stream or bin input from the file specified by the *Trail* parameter or from standard input. If you specify the **$bin** string as the value of the *Trail* parameter, the **auditbin** daemon substitutes the path name of the current bin file when it calls the **auditselect** command. The selection criteria can be entered as an expression or from the file specified by the **−f** flag. If the bin files are compressed, the **auditselect** command unpacks them prior to processing.

For stream data, configure both the **auditstream** command and the **auditselect** command in the **/etc/security/audit/streamcmds** file, or enter both commands from the command line.

Flags

−e"*Expression*"	Defines the selection criteria. The *Expression* parameter consists of one or more terms joined by logical operators.
−f*File*	Specifies the *File* that contains the selection criteria.

Creating Expressions

A valid expression consists of one or more terms joined by logical operators.

Logical Operators

Logical operators allow more than one term to be used in an expression. Normal precedence rules apply in evaluating expressions with more than one logical operator, and parentheses may be used to force the order of evaluation. The valid logical operators include the following:

&& (And) The expression `term1 && term2` is true (selected) if both term1 and term2 are true.

|| (Or) The expression `term1 || term2` is true (selected) if either term1 or term2 is true.

! (Not) The expression `!term1` is true (selected) if term1 is not true.

Terms

Each term of the expression has the following form:

```
Field Relational_Operator Value
```

Fields

Fields correspond to the information in the audit header of each record. Valid values for fields include the following:

`event` Name of the audit event, for example, `FILE_Open`.

`command` Name of the command that generated the audit event.

`result` Status of the audit event. The value of the `result` field must be one of the following:

- **OK**
- **FAIL**
- **FAIL_PRIV**
- **FAIL_AUTH**
- **FAIL_ACCESS**
- **FAIL_DAC** Indicates the event failed because of a discretionary access control (DAC) denial. Access Control Lists are a form of information repository that contain data relative to the rights of access (permission) to shared resources/objects. ACLs are categorized on DAC mechanism.

FAIL matches all other error codes.

`login` ID of the login user of the process that generated the audit event.

`real` ID of the real user of the process that generated the audit event.

`pid` ID of the process that generated the audit event.

`ppid` ID of the parent of the process that generated the audit event.

`tid` ID of the kernel thread that generated the event.

`time` Time of day the audit event was generated.

`date` Date the audit event was generated.

`host` Hostname of the machine that generated the record. The reserved name **UNKNOWN** can be used to match any machines that are not listed in the **/etc/security/audit/hosts** file.

Relational Operators

Relational operators are used to compare the field in the audit record to the specified value. Valid relational operators include:

`==` Equal to

`!=` Not equal to

`<` Less than

`>` Greater than

`>=` Greater than or equal to

`<=` Less than or equal to

Valid Terms

A valid term consists of a field, a relational operator, and a value. In addition, not all relational operators and values are valid for each field. The following are the valid combinations:

Field	Valid Operators	Valid Values
event	= =, ! =	Text string audit event name
result	= =, ! =	Text string audit status codes
command	= =, ! =	Text string command name
pid	all	Decimal integer process ID
ppid	all	Decimal integer process ID
login	all	Decimal integer user ID
login	= =, ! =	Text string user name
real	all	Decimal integer user ID
real	= =, ! =	Text string user name
tid	all	Decimal integer thread ID
time	all	String in the format specified by the current locale
date	all	String in the format specified by the current locale
host	= =, ! =	Text string host name or 16 character cpu ID

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be **setuid** to the root user and have the **trusted computing base** attribute.

Examples

Configuration

1. To select bin−collected data records that match the USER_SU or USER_Login audit events, add the **auditselect** command to the **/etc/security/audit/bincmds** file by entering:

   ```
   /usr/sbin/auditselect −e "event== USER_SU || event== \
   USER_Login" $bin >> /audit/trail.login
   ```

   While auditing is enabled, the records for each initiation of a user session are read from the current bin file and written to the **/audit/trail.login** file.

2. To select stream−collected data records that match a user login that was unsuccessful, add the **auditselect** command to the **auditstream** stanza in the **/etc/security/audit/streamcmds** file by entering:

   ```
   /usr/sbin/auditstream −c authentication | \
   /usr/sbin/auditselect −e "event == \
   USER_Login && result == FAIL" | \
   /usr/sbin/auditpr −t 2 −v >> /dev/lpr2
   ```

To produce a hardcopy audit trail, records of unsuccessful authentication events are written to the **/dev/lpr2** line printer.

Select Authentication or Login Events

1. To search an audit trail file for all events that involve authentication errors:

   ```
   /usr/sbin/auditselect -e "result == FAIL_AUTH"
   /audit/oldtrail | /usr/sbin/auditpr -t -helt -v
   ```

   The records of events that were unsuccessful because authentication was denied are printed. The header titles will be printed once, followed by the event, login ID, and time fields, and then the audit trail.

2. To select audit records that are generated when `smith` logs in during prime working hours during the first week in May of 1987, enter:

   ```
   /usr/sbin/auditselect -f /aaa/bbb \
   /audit/trail1987 | /usr/sbin/auditpr
   ```

   The `/aaa/bbb` file must contain the following line:

   ```
   command == login && login == smith &&
   time >= 08:00:00 && time <= 17:00:00 &&
   date >= 05/01/87 && date <= 05/05/87
   ```

String Comparison

1. To compare the name of the audit event to the `USER_Login` string, enter one of the following:

   ```
   "event == USER_Login"

   "event != USER_Login"
   ```

2. To find out if the **passwd** command generated the audit event, use:

   ```
   "command == passwd"
   ```

   To find out if the audit event was not generated by the **passwd** command, use:

   ```
   "command != passwd"
   ```

3. To compare the audit status to the `OK` result string, enter:

   ```
   "result == OK"
   ```

4. To compare the login or real user ID of the process that generated the audit event to a specific user ID (user ID `014` or the user name `carol`), enter one of the following:

   ```
   "login == 014"
   "login != carol"
   "login == 014 || login != carol"
   "real == carol"
   ```

5. To compare the ID of the process or the parent of the process that generated the audit event to the process ID `2006`, enter one of the following:

   ```
   "pid == 2006"
   "pid != 2006"
   "ppid ==  2006"
   ```

   Note: Although login and real user IDs and process IDs can be compared with the inequality operators (< =, > =, <, >), it is normally unnecessary to do this.

6. To compare the time the audit event was generated to the `08:03:00` time string, enter one of the following:

   ```
   "time == 08:03:00"
   "time != 08:03:00"
   "time <  08:03:00"
   "time <= 08:03:00"
   "time >  08:03:00"
   "time >= 08:03:00"
   ```

   Audit records are selected that fit the indicated comparison to the `08:03:00` time string. The time string must agree with the format specified by the current locale.

7. To compare the date that the audit event was generated to the `05/05/89` date string, enter one of the following:

   ```
   "date == 05/03/89"
   "date != 05/03/89"
   "date <  05/03/89"
   "date <= 05/03/89"
   "date >  05/03/89"
   "date >= 05/03/89"
   ```

Audit records are selected that fit the indicated comparison to the `05/05/89` date string. The date string must agree with the format specified by the current locale.

Files

/usr/sbin/auditselect	Specifies the path of the **auditselect** command.
/etc/rc	Contains the system initialization commands.
/etc/security/audit/config	Contains audit system configuration information.
/etc/security/audit/events	Contains the audit events of the system.
/etc/security/audit/objects	Contains audit events for audited objects (files).
/etc/security/audit/bincmds	Contains auditbin backend commands.
/etc/security/audit/streamcmds	Contains auditstream commands.
/etc/security/audit/hosts	Contains the CPU ID to hostname mappings.

Related Information

The **audit** command, **auditcat** command, **auditconv** command, **auditpr** command, **auditstream** command, **env** command.

auditbin daemon.

For general information on auditing, refer to Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

To see the steps you must take to establish an Auditing System, refer to Setting up Auditing in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

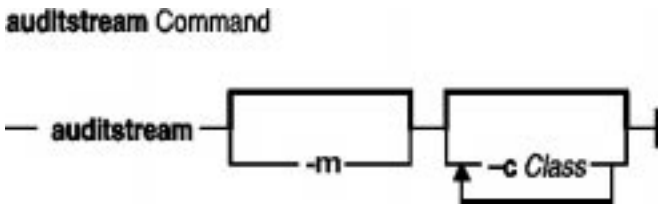
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

auditstream Command

Purpose

Creates a channel for reading audit records.

Syntax



auditstream [**−m**] [**−c***Class* ...]

Description

The **auditstream** command is part of the audit subsystem. This command reads audit records from the **/dev/audit** file (the audit device) and copies the records to standard output in binary format. You can select a subset of the audit records by specifying audit classes (defined in the **/etc/security/audit/config** file) with the **−c** flag; otherwise, all currently enabled audit classes are copied.

Audit stream data can be displayed and processed as it is generated. For example, the command output can be piped to an audit backend command for further processing or redirected to a file. Both the **auditselect** command, which selects data records according to defined criteria, and the **auditpr** command, which formats the records for viewing or for printing, are examples of backend commands.

The **auditstream** command can be called from the command line or be configured to run multiple times as part of the audit system configuration. For information on configuring the **auditstream** command, refer to "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* and to the **/etc/security/audit/config** file.

> **Note:** The **auditstream** command should be run in the background.

Flags

−c *Class* Specifies the audit classes to be copied. Each class must be configured in the **etc/security/audit/config** file as a list of comma−separated audit events. The default value is all the currently enabled audit events.

−m Includes the CPU ID in each audit record.

Security

Access Control: This command should grant execute (x) access to the root user and members of the audit group. The command should be **setuid** to the root user and have the **trusted computing base** attribute.

Files Accessed:

Mode	File

r **/dev/audit**

Examples

1. To configure the stream collection of audit data when the audit system is initialized, add the following to the stream stanza of the **/etc/security/audit/config** file:

   ```
   cmds  =  /etc/security/audit/streamcmds
   ```

   Then add the following to the start stanza:

   ```
   streammode=on
   ```

   Next, add to the /**etc/security/audit/streamcmds** file all the stream commands that should be executed when the auditing system is initialized. For example:

   ```
   /usr/sbin/auditstream  -c  authentication  |  \
   /usr/sbin/auditpr  -v  >  /dev/console

   /usr/sbin/auditstream  |  /usr/sbin/auditselect  -e  \
   "result  ==  FAIL_ACCESS"  |  \
   /usr/sbin/auditpr  -t  2  -v  >  /dev/lpr2
   ```

   The first command formats all records for events in the authentication class and writes them to the system console. The second command formats all records that resulted in an access denial and prints them on the printer **/dev/lp2**.

2. To record audit stream events on a line printer, enter:

   ```
   /usr/sbin/auditstream  |  /usr/sbin/auditselect  -e  "event  ==  \
   USER_Login  ||  event  ==  USER_SU"    |  \
   /usr/sbin/auditpr  -v  >  /dev/lp0  &
   ```

This command formats and writes all user login and **su** events to the line printer.

Files

/usr/sbin/auditstream	Specifies the path of the **auditstream** command.
/etc/rc	Contains the system startup routines.
/dev/audit	Specifies the audit device.
/etc/security/audit/config	Contains audit system configuration information.
/etc/security/audit/events	Contains the audit events of the system.
/etc/security/audit/objects	Contains audit events for audited objects (files).
/etc/security/audit/bincmds	Contains auditbin backend commands.
/etc/security/audit/streamcmds	Contains auditstream commands.
/etc/security/audit/hosts	Contains host and CPU IDs.

Related Information

The **audit** command, **auditcat** command, **auditconv** command, **auditpr** command, **auditselect** command.

The **auditbin** daemon.

For general information on auditing, refer to Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

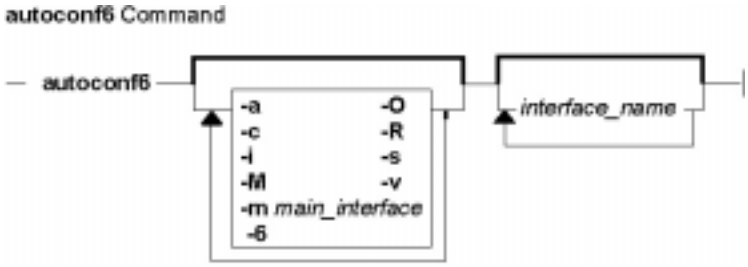
To see the steps you must take to establish an Auditing System, refer to Setting up Auditing in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

autoconf6 Command

Purpose

Automatically configures IPv6 network interfaces at boot time.

Syntax



autoconf6 [**−a**] [**−i**] [**−s**] [**−6**] [**−M**] [**−O**] [**−R**] [**−c**] [**−v**] [**−m***main_interface*] [*interface_name* ...]

Description

The **autoconf6** command is used at boot time to assign link−local addresses to ND−capable network interfaces. The **autoconf6** command initializes also the loopback interface, the automatic tunnels if needed, and adds some needed routes. It can also be used at any time to set link−local addresses and automatic tunnelling on newly configured ethernet−like interfaces.

Flags

−a	Configures and turns up all the acceptable interfaces.
−i	Configures and turns up the interfaces in the argument list. Without the **−a** and **−i** flags only the interfaces already up are configured.
−m*interface_name*	Specifies the main interface. You can also use the **no** command with argument **main_if6**.
−s	Installs the SIT interfaces and IPv4−compatible programs. Without this flag, the SIT interfaces are configured only if an SIT interface is already up.
−6	The SIT interface and IPv4−compatible interoperability are not installed or modified.
−M	(Debug) Do not modify existing IPv6 multicast routes.
−O	(Debug) Do not configure the loopback interface.
−R	(Debug) Do not install a default IPv6 route.
−c	Old compatibility flag for those who have bad LL addresses.
−v	Verbose output. The program displays what it is doing and/or what it is failing.
interface_name	Give to the program the name of the main IEEE LAN interface.

Messages

Messages indicate the different actions done and/or problems encountered by **autoconf6**.

Related Information

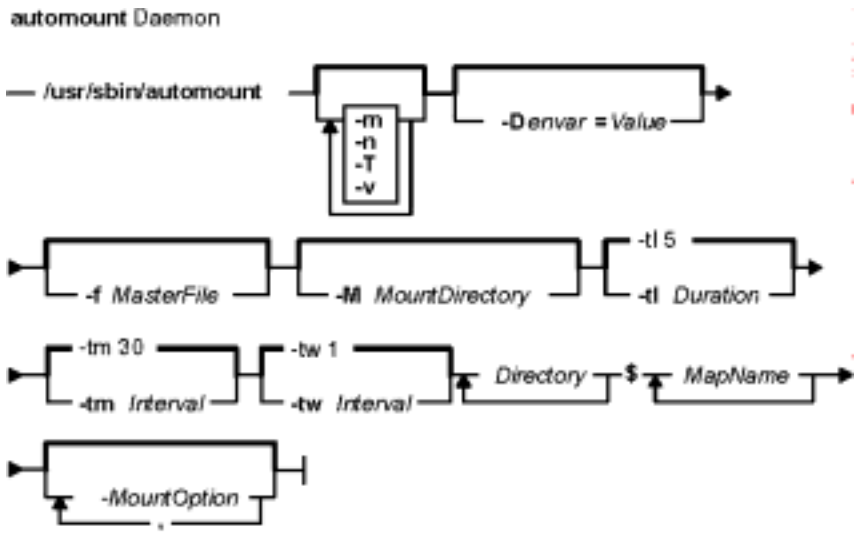
The **ifconfig** command, **ndpd−host** command, **ndpd−router** command, and **route** command.

automount Daemon

Purpose

Mounts automatic mount points.

Syntax



/usr/sbin/automount [**−m**] [**−n**] [**−T**] [**−v**] [**−D** *name*=*value*] [**−f** *MasterFile*]
[**−M** *MountDirectory*] [**−tl** *Duration*] [**−tm** *Interval*] [**−tw** *Interval*]
Directory ..**$**. *MapName* ... [*−MountOption* [**,***MountOption*] ...]

Description

The **automount** command is used as an administration tool for **AutoFS**. It installs **AutoFS** mount points and associates an **automount** map with each mount point. The **AutoFS** file system monitors attempts to access directories within it and notifies the **automountd** daemon. The daemon uses the map to locate a file system, which it then mounts at the point of reference within the **AutoFS** file system.

If the file system is not accessed within an appropriate interval (five minutes by default), the **automountd** daemon unmounts the file system.

The filename that contains the command line map information is **/tmp/autofs_cmdline**.

If the **automountd** daemon has not been started the **automount** command attempts to start it via **SRC**.

Maps

Updates to an map are transparent to the users because name−to−location binding is dynamic. This process eliminates the need to pre−mount shared file systems for applications containing hard−coded references to files.

See "How to Manage NIS automount Maps" in *AIX Version 4.3 System Management Guide: Communications and Networks* for more information about formatting map entries, multiple mounts, special maps, and the **auto_master/auto.master** NIS configuration map file.

The −*MountOptions* argument is a list of mount options. The list is preceded by a − (minus sign), and each option is separated by a comma. If this argument is supplied, the options usually become the default mount options for all entries in the map.

Notes:

1. Mount options provided in a map entry override the −*MountOptions* argument.
2. The **mount** command's **bg** background mount option is not recognized by the **automount** daemon.

The **automount** daemon is single−threaded. Any request delayed by a slow or unresponsive NFS server delays all subsequent automatic mount requests until the initial request completes. Programs that read the **/etc/mtab** file and then touch files that reside under automatic mount points introduce further entries to the file. Automatically mounted file systems are mounted with the **mount** command's **−t** type option set equal to *ignore*. These file systems do not appear in the output of either the **mount** command or the **df** command.

Environment Variables

Environment variables, specific only to the **automount** daemon, can be used in an **automount** map. When the daemon encounters an **automount** variable, the environment expands to account for the new variable. Environment variables are valid only for the automounter's environment, not for the operating system's environment.

References can be protected from affixed characters by enclosing the variable name in { } (curly braces).

> **Note:** Some NFS servers support mount options not supported by the AIX: *grpid*, *noauto*, *remount*, *quota*, *noquota*, *posix*, *nocto*, and *noac*. By default, the AIX version of the **automount** daemon ignores these listed options. To reverse the effect of this default, use the **AUTOMOUNT_BAD_OPTS** shell environment variable.

Configuration

The **automount** daemon normally consults the **auto.master** NIS configuration map for a list of initial *Directory*−to−*MapName* pairs, and sets up automatic mounts for them in addition to those given on the command line. If there are duplications, the command−line parameters take precedence.

> **Note:** This map contains the **automount** daemon parameter. The **automount** daemon does *not* look for an **auto.master** file on the local host.

See "How to Manage NIS automount Maps" in *AIX Version 4.3 System Management Guide: Communications and Networks* for more information about configuring the **auto.master** NIS map file.

Flags

−D *envar*=*Value*	Assigns a value to the indicated **automount** command environment variable. Supported for both implementations. See Note.
−f *MasterFile*	Reads the named local file, rather than the **master** NIS map file, for initialization. Supported for both implementations. See Note.
−m	Suppresses initialization of directory−mapname pairs listed in the **master** NIS database. Ignored in the AutoFS (AIX 4.3.1 and later) implementations.
−M *MountDirectory*	Ignored in the AutoFS (AIX 4.3.1 and later) implementations.
−n	Ignored in the AutoFS (AIX 4.3.1 and later) implementations.
−T	Traces automount activity for diagnostic purposes, displaying it on standard output. Supported for both implementations. See Note.
−t *Duration*	Specify a duration, in seconds, that a file system is to remain mounted when notin use.

	The default is 5 minutes. Supported only in the AutoFS (AIX 4.3.1 and later) implementations.
−tl *Duration*	Specifies a duration, in seconds, that a lookup name remains cached when not in use. The default is 5 minutes. Supported for both implementations. See Note.
−tm *Interval*	Specifies an interval in seconds, between attempts to mount a file system. The default is 30 seconds. Supported for both implementations. See Note.
−tw *Interval*	Specifies an interval, in seconds, between attempts to unmount file systems that have exceeded their cached times. The default is 60 seconds. Supported for both implementations. See Note.
−v	Displays on standard output verbose status and warning messages. Supported for both implementations. See Note.

Note: Some of the flags available for the **automount** command in previous versions are not supported in versions 4.3.1 and later. The **automount** command is a new implementation in 4.3.1 to support AutoFS. Some of these flags in the AIX 4.3.1 version of the **automount** command are accepted, but have no other purpose other than to maintain compatibility between the new AutoFS implementation of automatic mounting and the pre−AutoFS implementation.

The **−tl**, **−tm**, **−tw**, **−D**, and **−f** flags are supported for compatibility reasons, these flags are not supported in other vendor's current **automount** command implementations.

If you require full support of obsoleted arguments in AIX 4.3.1 and later you may still run the pre−AutoFS **automount** daemon by calling the **automount** command with the COMPAT_AUTOMOUNT environment value set to TRUE.

Files

/tmp_mount	Contains the directory under which file systems are dynamically mounted.
auto_master or **auto.master**	Contains the NIS configuration map for the **automount** daemon.

Related Information

The **df** command, **mount** command.

How to Manage NIS automount Maps in *AIX Version 4.3 System Management Guide: Communications and Networks* discusses map formatting, multiple mounts, special maps, and the **auto.master** NIS configuration map file.

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

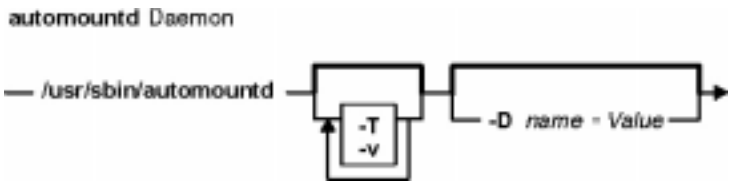
List of NFS Commands.

automountd Daemon

Purpose

AutoFS mount and unmount daemon.

Syntax



/usr/sbin/automountd−T−v−D*name*=*value*

Description

The **automountd** daemon is an RPC server that processes and answers requests from the local AutoFS filesystem kernel extension. It uses local files or name service maps to locate file systems to be mounted.

Maps

For a description on map files see the information on Maps in the automount daemon.

Flags

−D*name*=*Value*	Assigns a value to the indicated **automountd** daemon environment variable.
−T	Traces RPC server calls, displaying it on standard output.
−v	Displays on standard output verbose status and warning messages.

Related Information

The **df** command, **mount** command, **automount** daemon.

How to Manage NIS automount Maps in *AIX Version 4.3 Network Information Services (NIS and NIS+) Guide* discusses map formatting, multiple mounts, special maps, and the **auto_master/auto.master** NIS configuration map file.

List of NFS Commands.

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

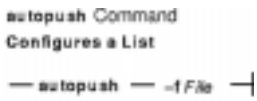
autopush Command

Purpose

Configures lists of automatically pushed STREAMS modules.

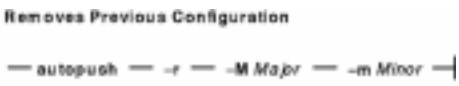
Syntax

To Configure A List Of Automatically Pushed Streams Modules



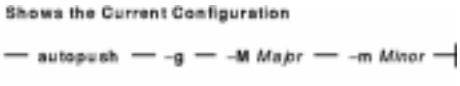
autopush **−f***File*

To Remove The Previous Configuration:



autopush−r−M*Major*−**m***Minor*

To Show The Current Configuration:



autopush−g−M *Major*−**m** *Minor*

Description

The **autopush** command configures the list of modules to be automatically pushed onto the stream when a device is opened. It can also remove a previous setting or obtain information on a setting.

Flags

−f *File* Sets up the **autopush** configuration for each driver according to the information stored in the specified file.

The file specified by the *File* parameter consists of lines consisting of at least four fields per line. Each field is separated by a character space as shown in the following example:

```
maj_ min_ last_min_ mod1 mod2 . . . modn
```

The first three fields are integers that specify the major device number, minor device number, and last minor device number. The subsequent fields represent the names of modules. If the value of

the `min_` field is −1, then all minor devices of a major driver specified by the `maj_` field are configured and the value of the `last_min_` field is ignored. If the value of the `last_min_` field is 0, then only a single minor device is configured. To configure a range of minor devices for a particular major, the value of the `min_` field must be less than the value of the `last_min_` field.

The last fields of a line in the **autopush** file represent the list of module names. Each module name is separated by a character space. The maximum number of modules that can be automatically pushed on a stream is eight, and they are pushed onto the stream in the order they are listed. Comment lines start with a # (pound sign).

−r Removes the previous configuration setting of a particular major and minor device number.

−g Obtains the current configuration setting of a particular major and minor device number. It also returns the starting minor device number if the request corresponds to a setting of a range.

−M*Major* Specifies a major device number.

−m*Minor* Specifies a minor device number.

AIX provides an enhancement to the **autopush** command that makes it easier to specify major numbers. The name of a driver can be specified instead of its major number anywhere the major number is normally used.

Parameters

File Contains at least the major device number, minor device number, last minor device number and modules.

Major Specifies a major device number.

Minor Specifies a minor device number.

Related Information

The **streamio** operations.

List of Streams Commands.

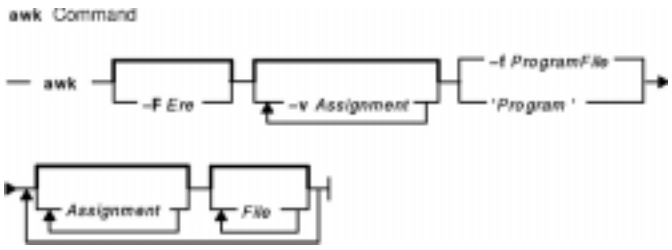
STREAMS Overview in *AIX Communications Programming Concepts*.

awk Command

Purpose

Finds lines in files that match patterns and then performs specified actions on them.

Syntax



awk [**−F** *Ere*] [**−v** *Assignment*] ... { **−f** *ProgramFile* | '*Program*' } [[*File* ... | *Assignment* ...]] ...

Description

The **awk** command utilizes a set of user−supplied instructions to compare a set of files, one line at a time, to extended regular expressions supplied by the user. Then actions are performed upon any line that matches the extended regular expressions.

The pattern searching of the **awk** command is more general than that of the **grep** command, and it allows the user to perform multiple actions on input text lines. The **awk** command programming language requires no compiling, and allows the user to use variables, numeric functions, string functions, and logical operators.

The following topics are covered in this article:

- Input for the awk Command
- Output for the awk Command
- File Processing with Records and Fields
- The awk Command Programming Language
    - ♦ Patterns
    - ♦ Actions
    - ♦ Variables
    - ♦ Special Variables
- Flags
- Examples

Input for the awk Command

The **awk** command takes two types of input: input text files and program instructions.

Input Text Files

Searching and actions are performed on input text files. The files are specified by:

- Specifying the *File* variable on the command line.
- Modifying the special variables **ARGV** and **ARGC**.
- Providing standard input in the absence of the *File* variable.

If multiple files are specified with the *File* variable, the files are processed in the order specified.

Program Instructions

Instructions provided by the user control the actions of the **awk** command. These instructions come from either the *`Program*' variable on the command line or from a file specified by the **−f** flag together with the *ProgramFile* variable. If multiple program files are specified, the files are concatenated in the order specified and the resultant order of instructions is used.

Output for the awk Command

The **awk** command produces three types of output from the data within the input text file:

- Selected data can be printed to standard output, without alteration to the input file.
- Selected portions of the input file can be altered.
- Selected data can be altered and printed to standard output, with or without altering the contents of the input file.

All of these types of output can be performed on the same file. The programming language recognized by the **awk** command allows the user to redirect output.

File Processing with Records and Fields

Files are processed in the following way:

1. The **awk** command scans its instructions and executes any actions specified to occur before the input file is read.

   The **BEGIN** statement in the **awk** programming language allows the user to specify a set of instructions to be done before the first record is read. This is particularly useful for initializing special variables.

2. One record is read from the input file.

   A record is a set of data separated by a record separator. The default value for the record separator is the new−line character, which makes each line in the file a separate record. The record separator can be changed by setting the **RS** special variable.

3. The record is compared against each pattern specified by the **awk** command's instructions.

   The command instructions can specify that a specific field within the record be compared. By default, fields are separated by white space (blanks or tabs). Each field is referred to by a field variable. The first field in a record is assigned the **$1** variable, the second field is assigned the **$2** variable, and so forth. The entire record is assigned to the **$0** variable. The field separator can be changed by using the **−F** flag on the command line or by setting the **FS** special variable. The **FS** special variable can be set to the values of: blank, single character, or extended regular expression.

4. If the record matches a pattern, any actions associated with that pattern are performed on the record.
5. After the record is compared to each pattern, and all specified actions are performed, the next record is read from input; the process is repeated until all records are read from the input file.
6. If multiple input files have been specified, the next file is then opened and the process repeated until all input files have been read.
7. After the last record in the last file is read, the **awk** command executes any instructions specified to occur after the input processing.

The **END** statement in the **awk** programming language allows the user to specify actions to be performed after the last record is read. This is particularly useful for sending messages about what work was accomplished by the **awk** command.

The awk Command Programming Language

The **awk** command programming language consists of statements in the form:

Pattern { *Action* }

If a record matches the specified pattern, or contains a field which matches the pattern, the associated action is then performed. A pattern can be specified without an action, in which case the entire line containing the pattern is written to standard output. An action specified without a pattern is performed for every input record.

Patterns

There are four types of patterns used in the **awk** command language syntax:

- Regular Expressions
- Relational Expressions
- Combinations of Patterns
- BEGIN and END Patterns.

Regular Expressions

The extended regular expressions used by the **awk** command are similar to those used by the **grep** or **egrep** command. The simplest form of an extended regular expression is a string of characters enclosed in slashes. For an example, suppose a file named `testfile` had the following contents:

```
smawley, andy
smiley, allen
smith, alan
smithern, harry
smithhern, anne
smitters, alexis
```

Entering the following command line:

```
awk '/smi/' testfile
```

would print to standard output of all records that contained an occurrence of the string `smi`. In this example, the program `'/smi/'` for the **awk** command is a pattern with no action. The output is:

```
smiley, allen
smith, alan
smithern, harry
smithhern, anne
smitters, alexis
```

The following special characters are used to form extended regular expressions:

Character	Function
+	Specifies that a string matches if one or more occurrences of the character or extended regular expression that precedes the + (plus) are within the string. The command line:

```
awk '/smith+ern/' testfile
```

prints to standard output any record that contained a string with the characters `smit`, followed by one or more `h` characters, and then ending with the characters `ern`. The output in this example is:

```
smithern, harry
smithhern, anne
```

? Specifies that a string matches if zero or one occurrences of the character or extended regular expression that precedes the ? (question mark) are within the string. The command line:

```
awk '/smith?/' testfile
```

prints to standard output of all records that contain the characters `smit,` followed by zero or one instance of the `h` character. The output in this example is:

```
smith, alan
smithern, harry
smithhern, anne
smitters, alexis
```

| Specifies that a string matches if either of the strings separated by the | (vertical line) are within the string. The command line:

```
awk '/allen
|
alan /' testfile
```

prints to standard output of all records that contained the string `allen` or `alan`. The output in this example is:

```
smiley, allen
smith, alan
```

( ) Groups strings together in regular expressions. The command line:

```
awk '/a(ll)?(nn)?e/' testfile
```

prints to standard output of all records with the string `ae` or `alle` or `anne` or `allnne`. The output in this example is:

```
smiley, allen
smithhern, anne
```

{*m*} Specifies that a string matches if exactly *m* occurrences of the pattern are within the string. The command line:

```
awk '/l{2}/' testfile
```

prints to standard output

```
smiley, allen
```

{*m*,} Specifies that a string matches if at least *m* occurrences of the pattern are within the string. The command line:

```
awk '/t{2,}/' testfile
```

prints to standard output:

```
smitters, alexis
```

{*m, n***}** Specifies that a string matches if between *m* and *n*, inclusive, occurrences of the pattern are within the string (where *m* <= *n*). The command line:

```
awk '/er{1, 2}/' testfile
```

prints to standard output:

```
smithern, harry
smithern, anne
smitters, alexis
```

[*String***]** Signifies that the regular expression matches any characters specified by the *String* variable within the square brackets. The command line:

```
awk '/sm[a-h]/' testfile
```

prints to standard output of all records with the characters `sm` followed by any character in alphabetical order from `a` to `h`. The output in this example is:

```
smawley, andy
```

[^ *String***]** A ^ (caret) within the [ ] (square brackets) and at the beginning of the specified string indicates that the regular expression *does not* match any characters within the square brackets. Thus, the command line:

```
awk '/sm[^a-h]/' testfile
```

prints to standard output:

```
smiley, allen
smith, alan
smithern, harry
smithhern, anne
smitters, alexis
```

~,!~ Signifies a conditional statement that a specified variable matches (tilde) or does not match (tilde, exclamation point) the regular expression. The command line:

```
awk '$1 ~ /n/' testfile
```

prints to standard output of all records whose first field contained the character `n.` The output in this example is:

```
smithern, harry
smithhern, anne
```

^ Signifies the beginning of a field or record. The command line:

```
awk '$2 ~ /^h/' testfile
```

prints to standard output of all records with the character `h` as the

first character of the second field. The output in this example is:

```
smithern, harry
```

$ Signifies the end of a field or record. The command line:

```
awk '$2 ~ /y$/' testfile
```

prints to standard output of all records with the character y as the last character of the second field. The output in this example is:

```
smawley, andy
smithern, harry
```

. (period) Signifies any one character except the terminal new−line character at the end of a space. The command line:

```
awk '/a..e/' testfile
```

prints to standard output of all records with the characters a and e separated by two characters. The output in this example is:

```
smawley, andy
smiley, allen
smithhern, anne
```

***** (asterisk) Signifies zero or more of any characters. The command line:

```
awk '/a.*e/' testfile
```

prints to standard output of all records with the characters a and e separated by zero or more characters. The output in this example is:

```
smawley, andy
smiley, allen
smithhern, anne
smitters, alexis
```

\ (backslash) The escape character. When preceding any of the characters that have special meaning in extended regular expressions, the escape character removes any special meaning for the character. For example, the command line:

```
/a\/\//
```

would match the pattern a //, since the backslashes negate the usual meaning of the slash as a delimiter of the regular expression. To specify the backslash itself as a character, use a double backslash. See the following item on escape sequences for more information on the backslash and its uses.

Recognized Escape Sequences

The **awk** command recognizes most of the escape sequences used in C language conventions, as well as several that are used as special characters by the **awk** command itself. The escape sequences are:

Escape Sequence	Character Represented
\"	\" (double−quotation) mark
\/	/ (slash) character

ddd	Character whose encoding is represented by a one−, two− or three−digit octal integer, where *d* represents an octal digit
\\	\ (backslash) character
\a	Alert character
\b	Backspace character
\f	Form−feed character
\n	New−line character (see following note)
\r	Carriage−return character
\t	Tab character
\v	Vertical tab.

Note: Except in the **gsub**, **match**, **split**, and **sub** built−in functions, the matching of extended regular expressions is based on input records. Record−separator characters (the new−line character by default) cannot be embedded in the expression, and no expression matches the record−separator character. If the record separator is not the new−line character, then the new−line character can be matched. In the four built−in functions specified, matching is based on text strings, and any character (including the record separator) can be embedded in the pattern so that the pattern matches the appropriate character. However, in all regular−expression matching with the **awk** command, the use of one or more NULL characters in the pattern produces undefined results.

Relational Expressions

The relational operators < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to), = = (equal to), and ! = (not equal to) can be used to form patterns. For example, the pattern:

```
$1 < $4
```

matches records where the first field is less than the fourth field. The relational operators also work with string values. For example:

```
$1 =! "q"
```

matches all records where the first field is not a `q`. String values can also be matched on collation values. For example:

```
$1 >= "d"
```

matches all records where the first field starts with a character that is `a`, `b`, `c`, or `d`. If no other information is given, field variables are compared as string values.

Combinations of Patterns

Patterns can be combined using three options:

- Ranges are specified by two patterns separated with a , (comma). Actions are performed on every record starting with the record that matches the first pattern, and continuing through and including the record that matches the second pattern. For example:

  ```
  /begin/,/end/
  ```

  matches the record containing the string `begin`, and every record between it and the record containing the string `end`, including the record containing the string `end`.

- Parentheses ( ) group patterns together.
- The boolean operators || (or), && (and), and ! (not) combine patterns into expressions that match if they evaluate true, otherwise they do not match. For example, the pattern:

```
$1 == "al" && $2 == "123"
```

matches records where the first field is `al` and the second field is `123`.

BEGIN and END Patterns

Actions specified with the **BEGIN** pattern are performed before any input is read. Actions specified with the **END** pattern are performed after all input has been read. Multiple **BEGIN** and **END** patterns are allowed and processed in the order specified. An **END** pattern can precede a **BEGIN** pattern within the program statements. If a program consists only of **BEGIN** statements, the actions are performed and no input is read. If a program consists only of **END** statements, all the input is read prior to any actions being taken.

Actions

There are several types of action statements:

- Action Statements
- Built−in Functions
- User−Defined Functions
- Conditional Statements
- Output Actions

Action Statements

Action statements are enclosed in { } (braces). If the statements are specified without a pattern, they are performed on every record. Multiple actions can be specified within the braces, but must be separated by new−line characters or ; (semicolons), and the statements are processed in the order they appear. Action statements include:

Arithmetical Statements

The mathematical operators + (plus), − (minus), / (division), ^ (exponentiation), * (multiplication), % (modulus) are used in the form:

```
Expression Operator Expression
```

Thus, the statement:

```
$2 = $1 ^ 3
```

assigns the value of the first field raised to the third power to the second field.

Unary Statements

The unary **−** (minus) and unary **+** (plus) operate as in the C programming language:

```
+Expression or −Expression
```

Increment and Decrement Statements

The pre−increment and pre−decrement statements operate as in the C programming language:

```
++Variable or −−Variable
```

The post−increment and post−decrement statements operate as in the C programming language:

```
Variable++ or Variable−−
```

Assignment Statements

The assignment operators += (addition), −= (subtraction), /= (division), and *= (multiplication) operate as in the C programming language, with the form:

```
Variable += Expression

Variable −= Expression

Variable /= Expression

Variable *= Expression
```

For example, the statement:

```
$1 *= $2
```

multiplies the field variable **$1** by the field variable **$2** and then assigns the new value to **$1**.

The assignment operators ^= (exponentiation) and %= (modulus) have the form:

```
Variable1^=Expression1
```

AND

```
Variable2%=Expression2
```

and they are equivalent to the C programming language statements:

```
Variable1=pow(Variable1, Expression1)
```

AND

```
Variable2=fmod(Variable2, Expression2)
```

where `pow` is the **pow** subroutine and `fmod` is the **fmod** subroutine.

String Concatenation Statements

String values can be concatenated by stating them side by side. For example:

```
$3 = $1 $2
```

assigns the concatenation of the strings in the field variables **$1** and **$2** to the field variable **$3**.

Built−In Functions

The **awk** command language uses arithmetic functions, string functions, and general functions. The close Subroutine statement is necessary if you intend to write a file, then read it later in the same program.

Arithmetic Functions

The following arithmetic functions perform the same actions as the C language subroutines by the same name:

atan2(*y*, *x* **)**	Returns arctangent of *y*/*x*.
cos(*x* **)**	Returns cosine of *x*; *x* is in radians.
sin(*x***)**	Returns sin of *x*; *x* is in radians.
exp(*x***)**	Returns the exponential function of *x*.
log(*x***)**	Returns the natural logarithm of *x*.
sqrt(*x***)**	Returns the square root of *x*.
int(*x***)**	Returns the value of *x* truncated to an integer.
rand()	Returns a random number *n*, with 0 <= *n* < 1.
srand([*Expr*] **)**	Sets the seed value for the **rand** function to the value of the *Expr* parameter, or use the time of day if the *Expr* parameter is omitted. The previous seed value is returned.

String Functions

The string functions are:

gsub(*Ere*, *Repl*, [*In*] **)**	Performs exactly as the **sub** function, except that all occurrences of the regular expression are replaced.
sub(*Ere*, *Repl*, [*In*] **)**	Replaces the first occurrence of the extended regular expression specified by the *Ere* parameter in the string specified by the *In* parameter with the string specified by the *Repl* parameter. The **sub** function returns the number of substitutions. An & (ampersand) appearing in the string specified by the *Repl* parameter is replaced by the string in the *In* parameter that matches the extended regular expression specified by the *Ere* parameter. If no *In* parameter is specified, the default value is the entire record (the **$0** record variable).
index(*String1*, *String2* **)**	Returns the position, numbering from 1, within the string specified by the *String1* parameter where the string specified by the *String2* parameter occurs. If the *String2* parameter does not occur in the *String1* parameter, a 0 (zero) is returned.
length [(*String***)]**	Returns the length, in characters, of the string specified by the *String* parameter. If no *String* parameter is given, the length of the entire record (the **$0** record variable) is returned.
blength [(*String***)]**	Returns the length, in bytes, of the string specified by the *String* parameter. If no *String* parameter is given, the length of the entire record (the **$0** record variable) is returned.
substr(*String*, *M*, [*N*] **)**	Returns a substring with the number of characters specified by the *N* parameter. The substring is taken from the string specified by the *String* parameter, starting with the character in the position specified by the *M* parameter. The *M* parameter is specified with the first character in the *String* parameter as number 1. If the *N* parameter is not specified, the length of the substring will be from the position specified by the *M* parameter until the end of the *String* parameter.
match(*String*, *Ere***)**	Returns the position, in characters, numbering from 1, in the string specified by the *String* parameter where the extended regular expression specified by the *Ere* parameter occurs, or else returns a 0 (zero) if the *Ere* parameter does not occur. The **RSTART** special variable is set to the return value. The **RLENGTH** special variable is set to the length of the matched string, or to −1 (negative one) if no match is found.

split(*String*, *A*, [*Ere*] **)**	Splits the string specified by the *String* parameter into array elements *A*[1], *A*[2], . . ., *A*[*n*], and returns the value of the *n* variable. The separation is done with the extended regular expression specified by the *Ere* parameter or with the current field separator (the **FS** special variable) if the *Ere* parameter is not given. The elements in the *A* array are created with string values, unless context indicates a particular element should also have a numeric value.
tolower(*String***)**	Returns the string specified by the *String* parameter, with each uppercase character in the string changed to lowercase. The uppercase and lowercase mapping is defined by the **LC_CTYPE** category of the current locale.
toupper(*String***)**	Returns the string specified by the *String* parameter, with each lowercase character in the string changed to uppercase. The uppercase and lowercase mapping is defined by the **LC_CTYPE** category of the current locale.
sprintf(*Format***,** *Expr***,** *Expr***,** . . . **)**	Formats the expressions specified by the *Expr* parameters according to the **printf** subroutine format string specified by the *Format* parameter and returns the resulting string.

General Functions

The general functions are:

close(*Expression***)**	Close the file or pipe opened by a **print** or **printf** statement or a call to the **getline** function with the same string−valued *Expression* parameter. If the file or pipe is successfully closed, a 0 is returned; otherwise a non−zero value is returned. The **close** statement is necessary if you intend to write a file, then read the file later in the same program.
system(*Command* **)**	Executes the command specified by the *Command* parameter and returns its exit status. Equivalent to the **system** subroutine.
Expression | **getline** [*Variable*]	Reads a record of input from a stream piped from the output of a command specified by the *Expression* parameter and assigns the value of the record to the variable specified by the *Variable* parameter. The stream is created if no stream is currently open with the value of the *Expression* parameter as its command name. The stream created is equivalent to one created by a call to the **popen** subroutine with the *Command* parameter taking the value of the *Expression* parameter and the *Mode* parameter set to a value of **r**. Each subsequent call to the **getline** function reads another record, as long as the stream remains open and the *Expression* parameter evaluates to the same string. If a *Variable* parameter is not specified, the **$0** record variable and the **NF** special variable are set to the record read from the stream.
getline [*Variable*] < *Expression*	Reads the next record of input from the file named by the *Expression* parameter and sets the variable specified by the *Variable* parameter to the value of the record. Each subsequent call to the **getline** function reads another record, as long as the stream remains open and the *Expression* parameter evaluates to the same string. If a *Variable* parameter is not specified, the **$0** record variable and the **NF** special variable are set to the record read from the stream.
getline [*Variable*]	Sets the variable specified by the *Variable* parameter to the next record of input from the current input file. If no *Variable* parameter is specified, **$0** record variable is set to the value of the record, and the **NF**, **NR**, and **FNR** special variables are also set.

Note: All forms of the **getline** function return 1 for successful input, zero for end of file, and −1 for an error.

User−Defined Functions

User−defined functions are declared in the following form:

```
function Name (Parameter, Parameter,...)  { Statements }
```

A function can be referred to anywhere in an **awk** command program, and its use can precede its definition. The scope of the function is global.

Function parameters can be either scalars or arrays. Parameter names are local to the function; all other variable names are global. The same name should not be used for different entities; for example, a parameter name should not be duplicated as a function name, or special variable. Variables with global scope should not share the name of a function. Scalars and arrays should not have the same name in the same scope.

The number of parameters in the function definition does not have to match the number of parameters used when the function is called. Excess formal parameters can be used as local variables. Extra scalar parameters are initialized with a string value equivalent to the empty string and a numeric value of 0 (zero); extra array parameters are initialized as empty arrays.

When invoking a function, no white space is placed between the function name and the opening parenthesis. Function calls can be nested and recursive. Upon return from any nested or recursive function call, the values of all the calling function's parameters shall be unchanged, except for array parameters passed by reference. The **return** statement can be used to return a value.

Within a function definition, the new−line characters are optional before the opening { (brace) and after the closing } (brace).

An example of a function definition is:

```
function average ( g,n)
  {
        for (i in g)
           sum=sum+g[i]
        avg=sum/n
        return avg
  }
```

The function average is passed an array, g, and a variable, n, with the number of elements in the array. The function then obtains an average and returns it.

Conditional Statements

Most conditional statements in the **awk** command programming language have the same syntax and function as conditional statements in the C programming language. All of the conditional statements allow the use of { } (braces) to group together statements. An optional new−line can be used between the expression portion and the statement portion of the conditional statement, and new−lines or ; (semicolon) are used to separate multiple statements in { } (braces). Six conditional statements in C language are:

if Requires the following syntax:

if (*Expression*) { *Statement }* [**else***Action*]

while Requires the following syntax:

while (*Expression*) { *Statement* }

for Requires the following syntax:

for (*Expression* ; *Expression* ; *Expression*) { *Statement* }

break Causes the program loop to be exited when the **break** statement is used in either a **while** or **for** statement.

continue Causes the program loop to move to the next iteration when the **continue** statement is used in either a **while** or **for** statement.

Five conditional statements in the **awk** command programming language that do not follow C−language rules are:

for...in Requires the following syntax:

for (*Variable* **in** *Array*) { *Statement* }

The **for**...**in** statement sets the *Variable* parameter to each index value of the *Array* variable, one index at a time and in no particular order, and performs the action specified by the *Statement* parameter with each iteration. See the **delete** statement for an example of a **for**...**in** statement.

if...in Requires the following syntax:

if (*Variable* **in** *Array*) { *Statement* }

The **if**...**in** statement searches for the existence of the *Array* element. The statement is performed if the *Array* element is found.

delete Requires the following syntax:

delete *Array* [*Expression*]

The **delete** statement deletes both the array element specified by the *Array* parameter and the index specified by the *Expression* parameter. For example, the statements:

```
for (i in g)
   delete g[i];
```

would delete every element of the `g[]` array.

exit Requires the following syntax:

exit [*Expression*]

The **exit** statement first invokes all **END** actions in the order they occur, then terminates the **awk** command with an exit status specified by the *Expression* parameter. No subsequent **END** actions are invoked if the **exit** statement occurs within an **END** action.

Requires the following syntax:

#Comment

The **#** statement places comments. Comments should always

	end with a new−line but can begin anywhere on a line.
next	Stops the processing of the current input record and proceeds with the next input record.

Output Statements

Two output statements in the **awk** command programming language are:

print Requires the following syntax:

print [*ExpressionList*] [*Redirection*] [*Expression*]

The **print** statement writes the value of each expression specified by the *ExpressionList* parameter to standard output. Each expression is separated by the current value of the **OFS** special variable, and each record is terminated by the current value of the **ORS** special variable.

The output can be redirected using the *Redirection* parameter, which can specify the three output redirections with the > (greater than), >> (double greater than), and the | (pipe). The *Redirection* parameter specifies how the output is redirected, and the *Expression* parameter is either a path name to a file (when *Redirection* parameter is > or >>) or the name of a command (when the *Redirection* parameter is a |).

printf Requires the following syntax:

printf*Format* [, *ExpressionList*] [*Redirection*] [*Expression*]

The **printf** statement writes to standard output the expressions specified by the *ExpressionList* parameter in the format specified by the *Format* parameter. The **printf** statement functions exactly like the **printf** command, except for the `c` conversion specification (%c). The *Redirection* and *Expression* parameters function the same as in the **print** statement.

For the `c` conversion specification: if the argument has a numeric value, the character whose encoding is that value will be output. If the value is zero or is not the encoding of any character in the character set, the behavior is undefined. If the argument does not have a numeric value, the first character of the string value will be output; if the string does not contain any characters the bahaviour is undefined.

Note: If the *Expression* parameter specifies a path name for the *Redirection* parameter, the *Expression* parameter should be enclosed in double quotes to insure that it is treated as a string.

Variables

Variables can be scalars, field variables, arrays, or special variables. Variable names cannot begin with a digit.

Variables can be used just by referencing them. With the exception of function parameters, they are not explicitly declared. Uninitialized scalar variables and array elements have both a numeric value of 0 (zero) and a string value of the null string (" ").

Variables take on numeric or string values according to context. Each variable can have a numeric value, a string value, or both. For example:

```
x = "4" + "8"
```

assigns the value of `12` to the variable `x`. For string constants, expressions should be enclosed in " " (double quotation) marks.

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

Field Variables

Field variables are designated by a **$** (dollar sign) followed by a number or numerical expression. The first field in a record is assigned the **$1** variable , the second field is assigned to the **$2** variable, and so forth. The **$0** field variable is assigned to the entire record. New field variables can be created by assigning a value to them. Assigning a value to a non−existent field, that is, any field larger than the current value of **$NF** field variable, forces the creation of any intervening fields (set to the null string), increases the value of the **NF** special variable, and forces the value of **$0** record variable to be recalculated. The new fields are separated by the current field separator (which is the value of the **FS** special variable). Blanks and tabs are the default field separators. To change the field separator, use the **−F** flag, or assign the **FS** special variable a different value in the **awk** command program.

Arrays

Arrays are initially empty and their sizes change dynamically. Arrays are represented by a variable with subscripts in [ ] (square brackets). The subscripts, or element identifiers, can be numbers of strings, which provide a type of associative array capability. For example, the program:

```
/red/  { x["red"]++ }
/green/ { y["green"]++ }
```

increments counts for both the `red` counter and the `green` counter.

Arrays can be indexed with more than one subscript, similar to multidimensional arrays in some programming languages. Because programming arrays for the **awk** command are really one dimensional, the comma−separated subscripts are converted to a single string by concatenating the string values of the separate expressions, with each expression separated by the value of the **SUBSEP** environmental variable. Therefore, the following two index operations are equivalent:

```
x[expr1, expr2,...exprn]
```

AND

```
x[expr1SUBSEPexpr2SUBSEP...SUBSEPexprn]
```

When using the **in** operator, a multidimensional *Index* value should be contained within parentheses. Except for the **in** operator, any reference to a nonexistent array element automatically creates that element.

Special Variables

The following variables have special meaning for the **awk** command:

ARGC The number of elements in the **ARGV** array. This value can be altered.

ARGV The array with each member containing one of the *File* variables or *Assignment* variables, taken in order from the command line, and numbered from 0 (zero) to **ARGC** −1. As each input file is finished, the next member of the **ARGV** array provides the name of the next input file, unless:

- The next member is an *Assignment* statement, in which case the assignment is evaluated.
- The next member has a null value, in which case the member is skipped. Programs can skip selected input files by setting the member of the **ARGV** array that contains that input file to a null value.

- The next member is the current value of **ARGV** [**ARGC** −1], which the **awk** command interprets as the end of the input files.

CONVFMT The **printf** format for converting numbers to strings (except for output statements, where the **OFMT** special variable is used). The default is "%.6g".

ENVIRON An array representing the environment under which the **awk** command operates. Each element of the array is of the form:

ENVIRON["*Environment VariableName***"]** = *EnvironmentVariableValue*

The values are set when the **awk** command begins execution, and that environment is used until the end of execution, regardless of any modification of the **ENVIRON** special variable.

FILENAME The path name of the current input file. During the execution of a **BEGIN** action, the value of **FILENAME** is undefined. During the execution of an **END** action, the value is the name of the last input file processed.

FNR The number of the current input record in the current file.

FS The input field separator. The default value is a blank. If the input field separator is a blank, any number of locale−defined spaces can separate fields. The **FS** special variable can take two additional values:

- With **FS** set to a single character, fields are separated by each single occurrence of the character.
- With **FS** set to an extended regular expression, each occurrence of a sequence matching the extended regular expression separates fields.

NF The number of fields in the current record, with a limit of 99. Inside a **BEGIN** action, the **NF** special variable is undefined unless a **getline** function without a *Variable* parameter has been issued previously. Inside an **END** action, the **NF** special variable retains the value it had for the last record read, unless a subsequent, redirected, **getline** function without a *Variable* parameter is issued prior to entering the **END** action.

NR The number of the current input record. Inside a **BEGIN** action the value of the **NR** special variable is 0 (zero). Inside an **END** action, the value is the number of the last record processed.

OFMT The **printf** format for converting numbers to strings in output statements. The default is "% .6g".

OFS The output field separator (default is a space).

ORS The output record separator (default is a new−line character).

RLENGTH The length of the string matched by the **match** function.

RS Input record separator (default is a new−line character). If the **RS** special variable is null, records are separated by sequences of one or more blank lines; leading or trailing blank lines do not result in empty records at the beginning or end of input; and the new−line character is always a field separator, regardless of the value of the **FS** special variable.

RSTART The starting position of the string matched by the **match** function, numbering from 1. Equivalent to the return value of the **match** function.

SUBSEP Separates multiple subscripts. The default is \031.

Flags

−f *ProgramFile* Obtains instructions for the **awk** command from the file specified by the *ProgramFile* variable. If the **−f** flag is specified multiple times, the concatenation of the files, in the order specified, will be used as the set of instructions.

−F *Ere* Uses the extended regular expression specified by the *Ere* variable as the field separator. The default field separator is a blank.

−v*Assignment* Assigns a value to a variable for the **awk** command's programming language. The *Assignment* parameter is in the form of *Name* = *Value*. The *Name* portion specifies the name of the variable and can be any combination of underscores, digits, and alphabetic characters,

but it must start with either an alphabetic character or an underscore. The *Value* portion is also composed of underscores, digits, and alphabetic characters, and is treated as if it were preceded and followed by a " (double−quotation character, similar to a string value). If the *Value* portion is numeric, the variable will also be assigned the numeric value.

The assignment specified by the **−v** flag occurs before any portion of the **awk** command's program is executed, including the **BEGIN** section.

Assignment Assigns a value to a variable for the **awk** command's programming language. It has the same form and function as the *Assignment* variable with the **−v** flag, except for the time each is processed. The *Assignment* parameter is processed just prior to the input file (specified by the *File* variable) that follows it on the command line. If the *Assignment* parameter is specified just prior to the first of multiple input files, the assignments are processed just after the **BEGIN** sections (if any). If an *Assignment* parameter occurs after the last file, the assignment is processed before the **END** sections (if any). If no input files are specified, the assignments are processed the standard input is read.

File Specifies the name of the file that contains the input for processing. If no *File* variable is specified, or if a − (minus) sign is specified, standard input is processed.

'*Program*' Contains the instructions for the **awk** command. If the **−f** flag is not specified, the *Program* variable should be the first item on the command line. It should be bracketed by ' ' (single quotes).

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

You can alter the exit status within the program by using the **exit** [*Expression*] conditional statement.

Examples

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

   ```
   awk 'length >72' chapter1
   ```

   This selects each line of the `chapter1` file that is longer than 72 characters and writes these lines to standard output, because no *Action* is specified. A tab character is counted as 1 byte.

2. To display all lines between the words `start` and `stop,` including `"start"` and `"stop",` enter:

   ```
   awk '/start/,/stop/' chapter1
   ```

3. To run an **awk** command program, `sum2.awk,` that processes the file, `chapter1`, enter:

   ```
   awk −f sum2.awk chapter1
   ```

   The following program, `sum2.awk`, computes the sum and average of the numbers in the second column of the input file, `chapter1`:

   ```
       {
          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`. All variables are initialized to the numeric value of 0 (zero) when first referenced. The pattern **END** before the second action causes those actions to be performed after all of the input file has been read. The **NR** special variable, which is used to calculate the average, is a special variable specifying the number of records that have been read.

4. To print the first two fields in opposite order, enter:

```
awk '{ print $2, $1 }' chapter1
```

5. The following **awk** program `sum3.awk` prints the first two fields of the file `chapter2` with input fields separated by comma and/or blanks and tabs, and then adds up the first column, and prints the sum and average:

```
BEGIN  {FS = ",|[ \t]+"}
       {print $1, $2}
       {s += $1}
END    {print "sum is",s,"average is", s/NR }
```

Related Information

Commands: **egrep**, **fgrep**, **grep**, **lex**, **printf**, **sed**.

Subroutines: **popen**, **printf**, **system**.

Books: Aho, A.V., Kernighan, B.W., and Weinberger, P.J. *The Awk Programming Language*. Bell Telephone Laboratories, Incorporated, 1988.

back Command

Purpose

Starts the backgammon game.

Syntax

back

Description

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

Important locations on the computer−generated board are:

- 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 prompted for `Instructions?` at the beginning of the game. During play, you are prompted for `move?`. Either enter a numerical move or press ? (question mark) key for a list of move choices.

When the game is finished, you are asked if you want to save game information. Entering `Y` stores game data in the **back.log** file in your current directory.

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

To quit the game, press the Interrupt (Ctrl−C) key sequence.

Files

/usr/games	Location of the system's games.
/usr/games/lib/backrules	Location of the rules file.
/tmp/b*	Location of the log temp file.
back.log	Contains data from previously played games.

Related Information

The **arithmetic** command, **bj** command, **craps** command, **fish** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command,

turnon command, **wump** command.

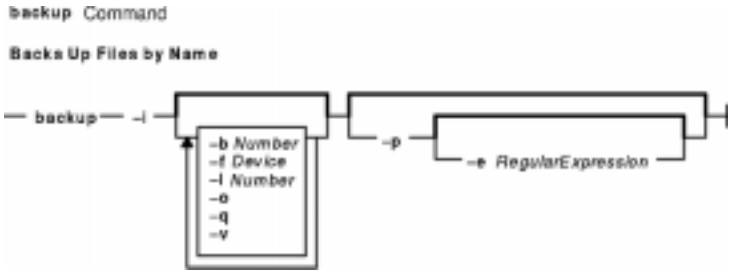
backup Command

Purpose

Backs up files and file systems.

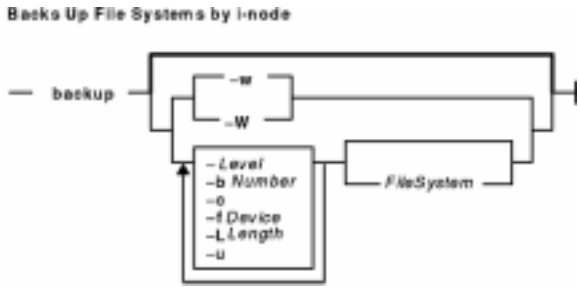
Syntax

To Back Up Files by Name



backup **−i** [**−b** *Number*] [**−p** [**−e** *RegularExpression*]] [**−f** *Device*] [**−l** *Number*] [**−o**] [**−q**] [**−v**]

To Back Up File Systems by i−node



backup [[−*Level*] [**−b** *Number*] [**−c**] [**−f** *Device*] [**−L** *Length*] [**−u**]] [*FileSystem*] | [**−w** | **−W**]

Description

The **backup** command creates copies of your files on a backup medium, such as a magnetic tape or diskette. The copies are in one of the two backup formats:

- Specific files backed up by name using the **−i** flag.
- Entire file system backed up by i−node using the *Level* and *FileSystem* parameters.

If you issue the **backup** command without any parameters, it defaults to a level 9 i−node backup of the root file system to the **/dev/rfd0** device. The default syntax is:

```
−9uf/dev/rfd0 /dev/rhd4
```

The default backup device is **/dev/rfd0**. If flags are specified that are not appropriate for the specified backup device, the **backup** command displays an error message and continues with the backup.

A single backup can span multiple volumes.

Notes:

1. Running the **backup** command results in the loss of all material previously stored on the selected output medium.
2. Data integrity of the archive may be compromised if a file is modified during system backup. Keep system activity at a minimum during the system backup procedure.
3. If a backup is made to a tape device with the device block size set to 0, it might be difficult to restore data from the tape unless the default write size was used with the **backup** command. The default write size for the **backup** command can be read by the **restore** command when the tape device block size is 0.

   In other words, the **−b** flag should not be specified when the tape device block size is 0. If the **−b** flag of the **backup** command is specified and is different from the default size, the same size must be specified with the **−b** flag of the **restore** command when the archived files are restored from the tape.

You can use a Web−based System Manager application (**wsm backup** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit backup** fast path to run this command.

Backing Up Files by Name

To back up by name, use the **−i** flag. The **backup** command reads standard input for the names of the files to be backed up.

File types can be special files, regular files, or directories. When the file type is a directory, only the directory is backed up. The files under the directory are not backed up, unless they are explicitly specified.

Notes:

1. Files are restored using the same path names as the archived files. Therefore, to create a backup that can be restored from any path, use full path names for the files that you want to back up.
2. When backing up files that require multiple volumes, do not enter the list of file names from the keyboard. Instead, pipe or redirect the list from a file to the **backup** command.

   When you enter the file names from the keyboard and the backup process needs a new tape or diskette, the command "loses" any file names already entered but not yet backed up. To avoid this problem, enter each file name only after the archived message for the previous file has been displayed. The archived message consists of the character `a` followed by the file name.

3. If you specify the **−p** flag, only files of less than 2GB are packed.

Backing Up File Systems by i−node

To back up a file system by i−node, specify the *−Level* and *FileSystem* parameters. When used in conjunction with the **−u** flag, the *−Level* parameter provides a method of maintaining a hierarchy of incremental backups for each file system. Specify the **−u** flag and set the *−Level* parameter to `n` to back up only those files that have been modified since the `n−1` level backup. Information regarding the date, time, and level of each incremental backup is written to the **/etc/dumpdates** file. The possible backup levels are 0 to 9. A level 0 backup archives all files in the file system. If the **/etc/dumpdates** file contains no backup information for a particular file system, specifying any level causes all files in that file system to be archived.

The *FileSystem* parameter can specify either the physical device name (block or raw name) or the name of the directory on which the file system is mounted. The default file system is the root (/) file system.

Users must have read access to the file system device (such as **/dev/hd4**) or have Backup authorization in order to perform backups by i_node.

Notes:

1. You must first unmount a file system before backing it up by i−node. If you attempt to back up a mounted file system, a warning message is displayed. The **backup** command continues, but the created backup may contain inconsistencies because of changes that may have occurred in the file system during the backup operation.
2. Backing up file systems by i−node truncates the **uid** or **gid** of files having a **uid** or **gid** greater than 65535. When restored, these files may have different values for the **uid** and **gid** attributes. To retain the values correctly, always back up by name files having a **uid** or **gid** greater than 65535.
3. You can archive only JFS (Journaled File System) file systems when backing up by i−node. Back up any non−JFS file systems by file name or by using other archive commands, such as the **pax**, **tar**, or **cpio** command.

Flags

−b *Number*
For backups by name, specifies the number of 512−byte blocks; for backups by i−node, specifies the number of 1024−byte blocks to write in a single output operation. When the **backup** command writes to tape devices, the default is 100 for backups by name and 32 for backups by i−node.

The write size is the number of blocks multiplied by the block size. The default write size for the **backup** command writing to tape devices is 51200 (100 * 512) for backups by name and 32768 (32 * 1024) for backups by i−node. The write size must be an even multiple of the tape's physical block size.

The value of the **−b** flag is always ignored when the **backup** command writes to diskette. In this case, the command always writes in clusters that occupy a complete track.

−c
Specifies that the tape is a cartridge, not a nine−track.

−e*RegularExpression*
Specifies that the files with names matching the regular expression are not to be packed. A regular expression is a set of characters, meta characters, and operators that define a string or group of strings in a search pattern. It can also be a string containing wildcard characters and operations that define a set of one or more possible strings. The **−e** flag is applied only when the **−p** flag is specified.

−f*Device*
Specifies the output device. To send output to a named device, specify the *Device* variable as a path name (such as `/dev/rmt0`). To send output to the standard output device, specify a − (minus sign). The − (minus) feature enables you to pipe the output of the **backup** command to the **dd** command.

You can also specify a range of archive devices. The range specification must be in the following format:

```
/dev/deviceXXX−YYY
```

where *XXX* and *YYY* are whole numbers, and *XXX* must always be less than *YYY*; for example, `/dev/rfd0−3`.

All devices in the specified range must be of the same type. For example, you can use a set of 8mm, 2.3GB tapes or a set of 1.44MB diskettes. All tape devices must be set

to the same physical tape block size.

If the *Device* variable specifies a range, the **backup** command automatically goes from one device in the range to the next. After exhausting all of the specified devices, the **backup** command halts and requests that new volumes be mounted on the range of devices.

−i Specifies that files be read from standard input and archived by file name. If relative path names are used, files are restored (with the **restore** command) relative to the current directory at restore time. If full path names are used, files are restored to those same names.

−L *Length* Specifies the length of the tape in bytes. This flag overrides the **−c**, **−d**, and **−s** flags. You can specify the size with a suffix of b, k, m, or g to represent Blocks (512 bytes), Kilo (1024 bytes), Mega (1024 Kilobytes), or Giga (1024 Megabytes), respectively. To represent a tape length of 2 Gigabytes, enter `−L 2g`. This flag only applies to AIX Version 4.2 and above.

Note: Use the **−L** flag for i−node backups only.

−l*Number* (lowercase L) Limits the total number of blocks to use on the diskette device. The value specified must be a non−zero multiple of the number of sectors per diskette track. This option applies to by−name backups only. See the format command for information on sectors per diskette track.

−o Creates a Version 2−compatible backup by name. This flag is required for compatibility with Version 2 systems because backups by name that are created by a version higher than 2 cannot be restored on Version 2 systems. To create a Version 2−compatible backup by name, use the **−o** flag along with other flags required for backups by name.

Files with attributes and values, such as user IDs and group IDs, that are too large for Version 2 systems will not be backed up. A message is displayed for each such file and each value that is too large.

−p Specifies that the files be packed, or compressed, before they are archived. Only files of less than 2GB are packed.

Note: This option should only be used when backing up files from an inactive filesystem. Modifying a file when a backup is in progress may result in corruption of the backup and an inability to recover the data. When backing up to a tape device which performs compression, this option can be omitted.

−q Indicates that the removable medium is ready to use. When you specify the **−q** flag, the **backup** command proceeds without prompting you to prepare the backup medium and press the Enter key to continue. This option applies only to the first volume; you are prompted for subsequent volumes. The **−q** flag applies only to backups by name.

−u Updates the **/etc/dumpdates** file with the raw device name of the file system and the time, date, and level of the backup. You must specify the **−u** flag if you are making incremental backups. The **−u** flag applies only to backups by i−node.

−v Causes the **backup** command to display additional information about the backup. When using the **−v** flag, the size of the file as it exists on the archive is displayed in bytes. Additionally, a total of these file sizes is displayed when all files have been processed. Directories are listed with a size of 0. Symbolic links are listed with the size of the symbolic link. Hard links are listed with the size of the file, which is how hard links are archived. Block and character devices, if they were backed up, are listed with a size of 0.

When the **−v** flag is not specified, the **backup** command displays only the names of the files being archived. This option is used only when backing up by file name.

−w Currently disabled. If the **−w** flag is specified, no other flags are applied.

−W	Displays, for each file system in the **/etc/dumpdates** file, the most recent backup date and level. If the **−W** option is specified, no other flags are applied.
−Level	Specifies the backup level (0 to 9). The default level is 9.

Exit Status

This command returns the following exit values:

0 Successful completion.
>**0** An error occurred.

Examples

1. To backup all the files and subdirectories in the `/home` directory using full path names, enter:

   ```
   find /home −print | backup −i −f /dev/rmt0
   ```

   The **−i** flag specifies that files will be read from standard input and archived by file name. The **find** command generates a list of all the files in the `/home` directory. The files in this list are full path names. The | (pipe symbol) causes this list to be read from standard input by the **backup** command. The **−f** flag directs the **backup** command to write the files to the `/dev/rmt0` tape device. Because the files are archived using full path names, they will be written to the same paths when restored.

2. To backup all the files and subdirectories in the `/home/mike` directory using relative path names, enter:

   ```
   cd /home
   find . −print | backup −i −v −q
   ```

   Each file name in the list generated by the **find** command is preceded by **./** (dot, slash). Because the files are backed up using relative path names, they will be written to the current directory when restored. The **−v** flag causes the **backup** command to display additional information about the backup. The files are written to the default backup device **/dev/rfd0**.

3. To backup the / (root) file system, enter:

   ```
   backup −0 −u −f /dev/rmt0 /
   ```

   The `0` level specifies that all the files in the / (root) file system be backed up. The **−u** flag causes the **backup** command to update the **/etc/dumpdates** file for this backup.

4. To backup all the files in the / (root) file system that have been modified since the last level 0 backup, enter:

   ```
   backup −1 −u −f /dev/rmt0 /
   ```

   If the **/etc/dumpdates** file does not have an entry for a level 0 backup of the / (root) system, all the files in the file system are backed up.

5. To run the **backup** command using the System Management Interface Tool (SMIT), enter:

   ```
   smit backup
   ```

Files

/etc/filesystems Contains file system mount information.

/etc/dumpdates	Specifies log for incremental by i−node backups.
/dev/rfd0	Specifies default backup device.
/dev/rhd4	Specifies device where the default file system (root) is located.
/usr/sbin/backup	Contains the **backup** command.

Related Information

The **dd** command, **find** command, **rdump** command, **restore** command.

The **dumpdates** file, **filesystems** file, **rmt** special file.

The Backup Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* provides information on different methods of backing up, restoring process, different types of backup media, and guidelines for backup policies.

The Directory Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* explains working with directories and path names.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains file system types, management, structure, and maintenance.

The Mounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains mounting files and directories, mount points, and automatic mounts.

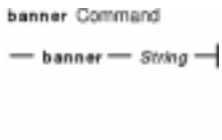
The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the structure, main menus, and tasks that are done with SMIT.

banner Command

Purpose

Writes ASCII character strings in large letters to standard output.

Syntax



banner*String*

Description

The **banner** command writes ASCII character *String*s to standard output in large letters. Each line in the output can be up to 10 uppercase or lowercase characters in length. On output, all characters appear in uppercase, with the lowercase input characters appearing smaller than the uppercase input characters.

Each word you input appears on a separate line on the screen. When you want to display more than one word to a line, use quotation marks to specify which words will appear on one line.

Examples

1. To display a banner at the workstation, enter:

   ```
   banner SMILE!
   ```

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

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

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

Files

/usr/bin/banner Contains the **banner** command.

Related Information

The **echo** command.

The Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output and how to use redirect and pipe symbols.

basename Command

Purpose

Returns the base file name of a string parameter.

Syntax

basename *String* [*Suffix*]

Description

The **basename** command reads the *String* parameter, deletes any prefix that ends with a / (slash) and any specified *Suffix* parameter, and writes the remaining base file name to standard output. The **basename** command applies the following rules in creating the base file name:

1. If the *String* parameter is a // (double slash), or if the *String* parameter consists entirely of slash characters, change the string to a single / (slash). Skip steps 2 through 4.
2. Remove any trailing / characters from the specified string.
3. If there are any / characters remaining in the *String* parameter, remove the prefix of the string up to and including the last / character.
4. If a *Suffix* parameter is specified and is identical to the characters remaining in the string, the string is not modified. For example, entering:

   ```
   K > basename /u/dee/desktop/cns.boo cns.boo
   ```

   results in:

   ```
   cns.boo
   ```

   If a *Suffix* parameter is specified and is not identical to all the characters in the string but is identical to a suffix in the string, the specified suffix is removed. For example, entering:

   ```
   K > basename /u/dee/desktop/cns.boo .boo
   ```

   results in:

   ```
   cns
   ```

Failure to find the specified suffix within a string is not considered an error.

The **basename** and **dirname** commands are generally used inside command substitutions within a shell script to specify an output file name that is some variation of a specified input file name.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

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

   ```
   basename $WORKFILE
   ```

   The command displays the base name of the value assigned to the shell variable `WORKFILE`. If the value of the `WORKFILE` variable is the **/home/jim/program.c** file, then the command displays `program.c`.

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

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

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

   Note: The ` (grave accent) specifies command substitution.

Files

/usr/bin/basename Contains the **basename** command.

Related Information

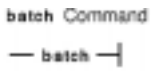
The **dirname** command, **sh** command.

batch Command

Purpose

Runs jobs when the system load level permits.

Syntax



batch

Description

The **batch** command reads from standard input the names of commands to be run at a later time and runs the jobs when the system load level permits. The **batch** command mails you all output from standard output and standard error for the scheduled commands, unless you redirect that output. It also writes the job number and the scheduled time to standard error.

When the **batch** command is executed, it retains variables in the shell environment, and the current directory; however, it does not retain open file descriptors, traps, and priority.

The **batch** command is equivalent to entering the **at −q b −m now** command. The **−q b** flag specifies the **at** queue for batch jobs.

Exit Status

This command returns the following exit values:

0 Successful completion
>0 An error occurred.

Examples

To run a job when the system load permits, enter:

```
batch <<!
longjob
!
```

This example shows the use of a "Here Document" to send standard input to the **batch** command.

Files

/usr/bin/batch	Contains the **batch** command.
/bin/batch	Symbolic link to the **batch** command.
/var/adm/cron	Indicates the main **cron** daemon directory.

/var/spool/cron/atjobs Indicates the spool area.

Related Information

at command, **bsh** command, **csh** command, **kill** command, **ksh** command, **mail** command, **nice** command, **ps** command.

Daemons: **cron**.

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output.

Korn Shell Special Commands and Bourne Shell Special Commands in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

National Language Support Overview for Programming in *AIX General Programming Concepts: Writing and Debugging Programs* explains collating sequences, equivalence classes, and locale.

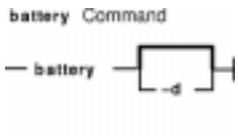
Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes what shells are, the different types of shells, and how shells affect the way commands are interpreted.

battery Command

Purpose

Controls or queries battery information.

Syntax



battery [**−d**]

Description

The **battery** command controls or queries the battery. If the **battery** command is invoked without **−d** option, the following battery information is displayed:

```
battery type: NiCd or NiMH
current battery usage: charging, discharging, in use, fully charged
battery capacity
current remaining capacity
full charge count
```

If the **battery** command is invoked with **−d** option, the following battery information is also displayed:

```
discharge quantity
discharge time
```

If you use 50% of a battery's capacity and charge it every time (about 20 to 30 times), then the battery cannot be used at more than 50% of its capacity. This is called the *memory effect of battery*. If, then, the battery is discharged (made empty) and then recharged, the battery can be used at 100% again.

Flags

−d Discharges the battery so you can reset the memory effect of battery.

Security

Access Control: Any User

Auditing Events: N/A

Examples

1. To show current battery status, enter:

   ```
   battery
   ```

   Something similar to the following displays:

```
battery type: NiMH
current battery usage: in use
battery capacity: 3200 (mAH)
current remaining capacity: 1800 (mAH) [57%]
full charge count: 3
```

Files

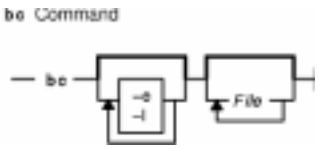
/usr/bin/battery Contains the **battery** command.

bc Command

Purpose

Provides an interpreter for arbitrary−precision arithmetic language.

Syntax



bc [**−c**] [**−l**] [*File ...*]

Description

The **bc** command is an interactive process that provides arbitrary−precision arithmetic. The **bc** command first reads any input files specified by the *File* parameter and then reads the standard input. The input files must be text files containing a sequence of commands, statements, or function definitions that the **bc** command can read and execute.

The **bc** command is a preprocessor for the **dc** command. It calls 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 standard output.

The **bc** command allows you to specify an input and output base for operations in decimal, octal, or hexadecimal. The default is decimal. The command also has a scaling provision for decimal point notation. The **bc** command always uses the . (period) to represent the radix point, regardless of any decimal point character specified as part of the current locale.

The syntax for the **bc** command is similar to that of the C language. You can use the **bc** command to translate between bases by assigning the **ibase** keyword to the input base and the **obase** keyword to the output base. A range of 2−16 is valid for the **ibase** keyword. The **obase** keyword ranges from 2 up to the limit set by the **BC_BASE_MAX** value defined in the **/usr/include/sys/limits.h** file. Regardless of the **ibase** and **obase** settings, the **bc** command recognizes the letters A−F as their hexadecimal values 10−15.

The output of the **bc** command is controlled by the program read. Output consists of one or more lines containing the value of all executed expressions without assignments. The radix and precision of the output are controlled by the values of the **obase** and **scale** keywords.

Further information about the way in which the **bc** command processes information from a source file is described in the following sections:

- Grammar
- Lexical Conventions
- Identifiers and Operators
- Expressions
- Statements
- Function Calls
- Functions in −I Math Library

Grammar

The following grammer describes the syntax for the **bc** program, where `program` stands for any valid program:

```
%token  EOF NEWLINE STRING LETTER NUMBER


%token  MUL_OP
/*      '*', '/', '%'                           */


%token  ASSIGN_OP
/*      '=', '+=', '-=', '*=', '/=', '%=', '^='  */


%token  REL_OP
/*      '==', '<=', '>=', '!=', '<', '>'          */


%token  INCR_DECR
/*      '++', '--'                              */


%token  Define    Break    Quit    Length
/*      'define', 'break', 'quit', 'length'       */


%token  Return    For    If    While    Sqrt
/*      'return', 'for', 'if', 'while', 'sqrt'    */


%token  Scale    Ibase    Obase    Auto
/*      'scale', 'ibase', 'obase', 'auto'         */


%start   program


%%


program           : EOF
                  | input_item program
                  ;


input_item        : semicolon_list NEWLINE
                  | function
                  ;


semicolon_list    : /* empty */
                  | statement
                  | semicolon_list ';' statement
```

```
                  | semicolon_list ';'
                  ;


statement_list    : /* empty */
                  | statement
                  | statement_list NEWLINE
                  | statement_list NEWLINE statement
                  | statement_list ';'
                  | statement_list ';' statement
                  ;


statement         : expression
                  | STRING
                  | Break
                  | Quit
                  | Return
                  | Return '(' return_expression ')'
                  | For '(' expression ';'
                       relational_expression ';'
                       expression ')' statement
                  | If '(' relational_expression ')' statement
                  | While '(' relational_expression ')' statement
                  | '{' statement_list '}'
                  ;

function          : Define LETTER '(' opt_parameter_list ')'
                       '{' NEWLINE opt_auto_define_list
                       statement_list '}'
                  ;


opt_parameter_list:/* empty */
                  | parameter_list
                  ;


parameter_list    : LETTER
                  | define_list ',' LETTER
                  ;


opt_auto_define_list
                  : /* empty */
                  | Auto define_list NEWLINE
                  | Auto define_list ';'
                  ;


define_list       : LETTER
                  | LETTER '[' ']'
                  | define_list ',' LETTER
                  | define_list ',' LETTER '[' ']'
                  ;


opt_argument_list : /* empty */
                  | argument_list
                  ;
```

```
argument_list     : expression
                  | argument_list ',' expression
                  ;


relational_expression
                  : expression
                  | expression REL_OP expression
                  ;


return_expression : /* empty */
                  | expression
                  ;


expression        : named_expression
                  | NUMBER
                  | '(' expression ')'
                  | LETTER '(' opt_argument_list ')'
                  | '-' expression
                  | expression '+' expression
                  | expression '-' expression
                  | expression MUL_OP expression
                  | expression '^' expression
                  | INCR_DECR named_expression
                  | named_expression INCR_DECR
                  | named_expression ASSIGN_OP expression
                  | Length '(' expression ')'
                  | Sqrt '(' expression ')'
                  | Scale '(' expression ')'
                  ;


named_expression  : LETTER
                  | LETTER '[' expression ']'
                  | Scale
                  | Ibase
                  | Obase
                  ;
```

Lexical Conventions

The following lexical conventions apply to the **bc** command:

1. The **bc** command recognizes the longest possible lexical token or delimiter beginning at a given point.
2. Comments begin with /* (slash, asterisk) and end with */ (asterisk, slash). Comments have no effect except to delimit lexical tokens.
3. The newline character is recognized as the **NEWLINE** token.
4. The **STRING** token represents a string constant. The string begins with " (double quotation mark) and terminates with " (double quotation mark). All characters between the quotation marks are taken literally. There is no way to specify a string that contains " (double quotation mark). The length of each string is limited to the maximum bytes set in the **BC_STRING_MAX** value, which is defined in the **limits.h** file.
5. Blank characters have no effect except as they appear in the **STRING** token or when used to delimit lexical tokens.
6. The \n (backslash, newline) character:

- delimits lexical tokens.
- is interpreted as a character sequence in **STRING** tokens.
- is ignored when part of a multiline **NUMBER** token.

7. A **NUMBER** token uses the following grammar:

```
NUMBER  : integer
        | '.' integer
        | integer '.'
        |integer '.' integer
        ;
integer : digit
        | integer digit
        ;
digit   : 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
        | 8 | 9 | A | B | C | D | E | F
        ;
```

   NUMBER token values are interpreted as numerals in the base specified by the **ibase** internal register value.

8. The value of a **NUMBER** token is interpreted as a numeral in the base specified by the value of the **ibase** internal register. Each of the digit characters has the value from 0 to 15 in the order listed here, and the period character presents the radix point. The behavior is undefined if digits greater than or equal to the value of the **ibase** register appear in the token. There is an exception for single−digit values being assigned to the **ibase** and **obase** registers themselves.
9. The following keywords are recognized as tokens:

```
auto    for     length   return sqrt
break   ibase   obase    scale  while
define  if      quit
```

10. Except within a keyword, any of the following letters are considered a **LETTER** token:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

11. The following single−character and two−character sequences are recognized as the **ASSIGN_OP** token:
    - = (equal sign)
    - += (plus, equal sign)
    - −= (minus, equal sign)
    - *= (asterisk, equal sign)
    - /= (slash, equal sign)
    - %= (percent, equal sign)
    - ^= (caret, equal sign)
12. The following single characters are recognized as the **MUL_OP** token:
    - * (asterisk)
    - / (slash)
    - % (percent)
13. The following single−character and two−character sequences are recognized as the **REL_OP** token:
    - == (double equal sign)
    - <= (less than, equal sign)
    - >= (greater than, equal sign)
    - != (exclamation point, equal sign)
    - < (less than)
    - > (greater than)
14. The following two−character sequences are recognized as the **INCR_DECR** token:
    - ++ (double plus sign)
    - −− (double hyphen)
15. The following single characters are recognized as tokens. The token has the same name as the character:

<newline>
((left parenthesis)
) (right parenthesis)
, (comma)
+ (plus)
− (minus)
; (semicolon)
[(left bracket)
] (right bracket)
^ (caret)
{ (left brace)
} (right brace)

16. The **EOF** token is returned when the end of input is reached.

Identifiers and Operators

There are three kinds of identifiers recognized by the **bc** command: ordinary identifiers, array identifiers, and function identifiers. All three types consist of single, lowercase letters. Array identifiers are followed by [ ] (left and right brackets). An array subscript is required except in an argument or auto list. Arrays are singly dimensioned and can contain up to the amount specified by the **BC_DIM_MAX** value. Indexing begins at 0. Therefore an array is indexed from 0 up to the value defined by **BC_DIM_MAX−1**. Subscripts are truncated to integers. Function identifiers must be followed by ( ) (left and right parentheses) and possibly by enclosing arguments. The three types of identifiers do not conflict.

The Operators in a bc Program table summarizes the rules for precedence and associativity of all operators. Operators on the same line have the same precedence. Rows are in order of decreasing precedence.

Operators in a bc Program	
Operator	**Associativity**
++, − −	not applicable
unary −	not applicable
^	right to left
*, /, %	left to right
+, binary −	left to right
=, +=, −=, *=, /=, ^=	right to left
==, <=, >=, !=, <, >	none

Each expression or named expression has a *scale*, which is the number of decimal digits maintained as the fractional portion of the expression.

Named expressions are places where values are stored. Named expressions are valid on the left side of an assignment. The value of a named expression is the value stored in the place named. Simple identifiers and array elements are named expressions; they have an initial value of zero and an initial scale of zero.

The internal registers **scale**, **ibase**, and **obase** are all named expressions. The scale of an expression consisting of the name of one of these registers is 0. Values assigned to any of these registers are truncated to integers. The **scale** register contains a global value used in computing the scale of expressions (as described below). The value of the **scale** register is limited to 0 <= **scale** <= {**BC_SCALE_MAX**} and has a default value of 0. The **ibase** and **obase** registers are the input and output number radix, respectively. The value of **ibase** is limited to 2 <= **ibase** <= 16. The value of **obase** is limited to 2 <= **obase** = {**BC_BASE_MAX**}

When either the **ibase** or **obase** registers are assigned a single−digit value from the list described in "Lexical Conventions" , the value is assumed in hexadecimal. For example:

```
ibase=A
```

sets to base ten, regardless of the current **ibase** register value. Otherwise, the behavior is undefined when digits greater than or equal to the value of the **ibase** register appear in the input. Both **ibase** and **obase** registers have initial values of 10.

Internal computations are conducted as if in decimal, regardless of the input and output bases, to the specified number of decimal digits. When an exact result is not achieved, for example:

```
scale=0; 3.2/1
```

the **bc** command truncates the result.

All numerical values of the **obase** register are output according to the following rules:

1. If the value is less than 0, output a − (hyphen).
2. Output one of the following, depending on the numerical value:
   - ♦ If the absolute value of the numerical value is greater than or equal to 1, output the integer portion of the value as a series of digits appropriate to the **obase** register (described in step 3). Next output the most significant non−zero digit, followed by each successively less significant digit.
   - ♦ If the absolute value of the numerical value is less than 1 but greater than 0 and the scale of the numerical value is greater than 0, it is unspecified whether the character 0 is output.
   - ♦ If the numerical value is 0, output the character 0.
3. If the scale of the value is greater than 0, output a . (period) followed by a series of digits appropriate to the following **obase** register values. The digits represent the most significant portion of the fractional part of the value, and *s* represents the scale of the value being output:
   - ♦ If the **obase** value is 10, output *s* number of digits.
   - ♦ If the **obase** value is greater than 10, output the number less than or equal to *s*.
   - ♦ If the **obase** value is less than 10, output a number greater than or equal to *s*.
   - ♦ For **obase** values other than 10, this should be the number of digits needed to represent a precision of 10s.
   - ♦ For **obase** values from 2 to 16, valid digits are the first **obase** of the single characters:

     ```
     0   1   2   3   4   5   6   7   8   9   A   B   C   D   E   F
     ```

   which represent the values 0 through 15, respectively.
4. For bases greater than 16, each digit is written as a separate multidigit decimal number. Each digit except the most significant fractional digit is preceded by a single space character. For bases 17 to 100, the **bc** command writes two−digit decimal numbers, for bases 101 to 1000 the **bc** command writes three−digit decimal numbers. For example, the decimal number 1024 in base 25 would be written as:

   ```
   01 15 24
   ```

   in base 125, as:

   ```
   008 024
   ```

Very large numbers are split across lines, with 70 characters per line in the POSIX locale. Other locales may split at different character boundaries. Lines that are continued must end with a \ (backslash).

Expressions

A numeric constant is an expression. The scale is the number of digits that follow the radix point in the input representing the constant, or 0 if no radix point appears.

The sequence (*expression*) is an expression with the same value and scale as *expression*. The parentheses can be used to alter the normal precedence.

The unary and binary operators have the following semantics:

−*expression*	The result is the negative of the expression. The scale of the result is the scale of the expression. The unary increment and decrement operators do not modify the scale of the named expression upon which they operate. The scale of the result is the scale of that named expression.
++*named_expression*	The named expression is incremented by 1. The result is the value of the named expression after incrementing.
− −*named_expression*	The named expression is decremented by 1. The result is the value of the named expression after decrementing.
named_expression++	The named expression is incremented by 1. The result is the value of the named expression before incrementing.
named_expression− −	The named expression is decremented by 1. The result is the value of the named expression before decrementing.

The exponentiation operator, ^ (caret), binds right to left.

expression ^*expression*	The result is the first *expression* raised to the power of the second *expression.* If the second expression is not an integer, the behavior is undefined. If a is the scale of the left expression and b is the absolute value of the right expression, the scale of the result is:
	
	`if b >= 0 min(a * b, max(scale, a))`
	`if b <     0 scale`

The multiplicative operators * (asterisk), / (slash), and % (percent) bind left to right.

expression * *expression*	The result is the product of the two expressions. If a and b are the scales of the two expressions, then the scale of the result is: `min(a+b,max(scale,a,b))`
expression / *expression*	The result is the quotient of the two expressions. The scale of the result is the value of **scale**.

expression % *expression*	For expressions a and b, a % b is evaluated equivalent to the following steps:
	1. Compute a/b to current scale.
	2. Use the result to compute:
	
	`a − (a / b) * b`
	
	to scale:
	
	`max(scale + scale(b), scale(a))`
	
	The scale of the result will be:

```
max(scale + scale(b), scale(a))
```

When **scale** is zero, the % operator is the mathematical remainder operator.

The additive operators + (plus) and − (minus) bind left to right.

expression + *expression*	The result is the sum of the two expressions. The scale of the result is the maximum of the scales of the expressions.
expression − *expression*	The result is the difference of the two expressions. The scale of the result is the maximum of the scales of the expressions.

The following assignment operators bind right to left:

- = (equal sign)
- += (plus, equal sign)
- −= (minus, equal sign)
- *= (asterisk, equal sign)
- /= (slash, equal sign)
- %= (percent, equal sign)
- ^= (caret, equal sign)

named−expression = *expression*	This expression results in assigning the value of the expression on the right to the named expression on the left. The scale of both the named expression and the result is the scale of the expression.

The compound assignment forms:

named−expression <*operator* >= *expression*

are equivalent to:

named−expression = *named−expression* <*operator* > *expression*

except that the named expression is evaluated only once.

Unlike all other operators, the following relational operators are only valid as the object of an **if** or **while** statement or inside a **for** statement:

- < (less than)
- > (greater than)
- <= (less than, equal sign)
- >= (greater than, equal sign)
- == (double equal sign)
- != (exclamation, equal sign)

expression1 < *expression2*	The relation is true if the value of *expression1* is strictly less than the value of *expression2*.
expression1 > *expression2*	The relation is true if the value of *expression1* is strictly greater than the value of *expression2*.
expression1 <= *expression2*	The relation is true if the value of *expression1* is less than or equal to the value of *expression2*.
expression1 >= *expression2*	The relation is true if the value of *expression1* is greater than or equal to the value of *expression2*.
expression1 == *expression2*	The relation is true if the values of *expression1* and *expression2* are equal.

expression1 != *expression2* The relation is true if the values of *expression1* and *expression2* are unequal.

Statements

When a statement is an expression, unless the main operator is an assignment, execution of the statement writes the value of the expression followed by a newline character.

When a statement is a string, execution of the statement writes the value of the string.

Statements separated by semicolons or newline characters are executed sequentially. In an interactive invocation of the **bc** command, each time a newline character is read that satisfies the grammatical production:

```
input_item : semicolon_list NEWLINE
```

the sequential list of statements making up the **semicolon_list** is executed immediately, and any output produced by that execution is written without any buffer delay.

If an **if** statement (**if** (*relation*) *statement*), the *statement* is executed if the relation is true.

The **while** statement (**while** (*relation*) *statement*) implements a loop in which the *relation* is tested. Each time the *relation* is true, the *statement* is executed and the *relation* retested. When the *relation* is false, execution resumes after *statement.*

A **for** statement (**for** (*expression; relation; expression*) *statement*) is the same as:

```
first-expression
while   (relation) {
   statement
   last-expression
}
```

All three expressions must be present.

The **break** statement causes termination for a **for** or **while** statement.

The **auto** statement (**auto***identifier***[**,*identifier* **]** ...) causes the values of the identifiers to be pushed down. The identifiers can be ordinary identifiers or array identifiers. Array identifiers are specified by following the array name by empty square brackets. The **auto** statement must be the first statement in a function definition.

The **define** statement:

```
define   LETTER ( opt_parameter_list )  {
   opt_auto_define_list
   statement_list
}
```

defines a function named LETTER. If the LETTER function was previously defined, the **define** statement replaces the previous definition. The expression:

```
LETTER ( opt_argument_list )
```

invokes the LETTER function. The behavior is undefined if the number of arguments in the invocation does not match the number of parameters in the definition. Functions are defined before they are invoked. A function is considered defined within its own body, so recursive calls are valid. The values of numeric

constants within a function are interpreted in the base specified by the value of the **ibase** register when the function is invoked.

The **return** statements (**return** and **return**(*expression*)) cause termination of a function, popping of its **auto** variables, and specify the result of the function. The first form is equivalent to return(0). The value and scale of an invocation of the function is the value and scale of the expression in parentheses.

The **quit** statement (**quit**) stops execution of a **bc** program at the point where the statement occurs in the input, even if it occurs in a function definition or in an **if**, **for**, or **while** statement.

Function Calls

A function call consists of a function name followed by parentheses containing a comma−separated list of expressions, which are the function arguments. A whole array passed as an argument is specified by the array name followed by [ ] (left and right brackets). All function arguments are passed by value. As a result, changes made to the formal parameters have no effect on the actual arguments. If the function terminates by executing a **return** statement, the value of the function is the value of the expression in the parentheses of the **return** statement, or 0 if no expression is provided or if there is no **return** statement.

The result of **sqrt**(*expression*) is the square root of the expression. The result is truncated in the least significant decimal place. The scale of the result is the scale of the expression or the value of **scale**, whichever is larger.

The result of **length**(*expression*) is the total number of significant decimal digits in the expression. The scale of the result is 0.

The result of **scale**(*expression*) is the scale of the expression. The scale of the result is 0.

There are only two storage classes in a **bc** program, global and automatic (local). Only identifiers that are to be local to a function need be declared with the **auto** keyword. The arguments to a function are local to the function. All other identifiers are assumed to be global and available to all functions. All identifiers, global and local, have initial values of 0. Identifiers declared as **auto** are allocated on entry to the function and released on returning from the function. Therefore they do not retain values between function calls. The **auto** arrays are specified by the array name followed by [ ] (left bracket, right bracket). On entry to a function, the old values of the names that appear as parameters and as automatic variables are pushed onto a stack. Until the function returns, reference to these names refers only to the new values.

References to any of these names from other functions that are called from this function also refer to the new value until one of those functions uses the same name for a local variable.

Functions in −l Math Library

The following functions are defined when you specify the **−l** flag:

s(*expression***)**	Specifies the sine of *expressionx*, where *expression* is in radians.
c(*expression***)**	Specifies the cosine of *expressionx*, where *expression* is in radians.
a(*expression***)**	Specifies the arctangent of *expressionx*, where *expression* is in radians.
l(*expression***)**	Specifies the natural logarithm of *expression*.
e(*expression***)**	Specifies the exponential of *expression*.
j(*expression***,***expression***)**	Specifies the Bessel function of integer order.

The scale of an invocation of each of these functions is the value of the **scale** keyword when the function is invoked. The behavior is undefined if any of these functions is invoked with an argument outside the domain

of the mathematical function.

Flags

−c Compiles the *File* parameter, but does not invoke the **dc** command.
−l (Lowercase L) Defines a library of math functions, and sets the **scale** variable to 20.

Exit Status

This command returns the following exit values:

0 Successful completion.
1 Encountered a syntax error or could not access the input file.
unspecified Any other error occurred.

Examples

1. You can use the **bc** command as a calculator. Depending on whether you set the **scale** variable and with what value, the system displays fractional amounts. Entering:

   ```
   bc
   1/4
   ```

   displays only `0`. To set the **scale** variable and add a comment, enter:

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

   The screen displays `0.2`. Entering:

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

   displays `0.250`. Entering:

   ```
   16+63/5
   ```

   displays `28.600`. Entering

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

   displays `15.800`. Entering

   ```
   71/6
   ```

   displays `11.833`.

   The **bc** command displays the value of each expression when you press the Enter key, except for assignments.

   When you enter the **bc** command expressions directly from the keyboard, press the End−of−File (Ctrl−D) key sequence to end the **bc** command session and return to the shell command line.

2. To write and run a C−like program, enter a command similar to the following:

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

The screen displays `7.389056098930650227`23. If you enter:

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

The screen displays `120`. If you enter:

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

The screen displays `3628800`.

This sequence interprets the **bc** program saved in the **prog.bc** file, and reads more of the **bc** command statements from the 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 **prog.bc** program file 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. When you enter the **bc** command expressions directly from the keyboard, press the End−of−File (Ctrl−D) key sequence to end the **bc** command session and return to the shell command line.

3. To convert an infix expression to Reverse Polish Notation (RPN), enter:

```
bc −c
(a * b) % (3 + 4 * c)
```

The screen displays:

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

This sequence compiles the **bc** command infix−notation expression into an expression that the **dc** command can interpret. The **dc** command evaluates extended RPN expressions. In the compiled output, the `l` before each variable name is the **dc** subcommand to load 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. When you enter the **bc** command expressions directly from the keyboard, press the End−of−File (Ctrl−D) key sequence to end the **bc** command session and return to the shell command line.

4. To assign in the shell an approximation of the first 10 digits of pi to the variable *x*, enter:

```
x=$(printf "%s\n" 'scale = 10; 104348/33215' | bc)
```

The following **bc** program prints the same approximation of pi, with a label, to standard output:

```
scale = 10
"pi equals "
104348 / 33215
```

5. To define a function to compute an approximate value of the exponential function (such a function is predefined if the −l (lowercase L) option is specified), enter:

```
scale = 20
define  e(x){
        auto a, b, c, i, s
        a = 1
        b = 1
        s = 1
        for (i = 1; 1 == 1; i++){
                a = a*x
                b = b*i
                c = a/b
                if (c == 0) {
                        return(s)
                }
                s = s+c
        }
}
```

   To print approximate values of the exponential function of the first 10 integers, enter:

```
for (i = 1; i <= 10; ++i) {
        e(i)
}
```

Files

/usr/bin/bc	Contains the **bc** command.
/usr/lib/lib.b	Contains the mathematical library.
/usr/bin/dc	Contains the desk calculator.

Related Information

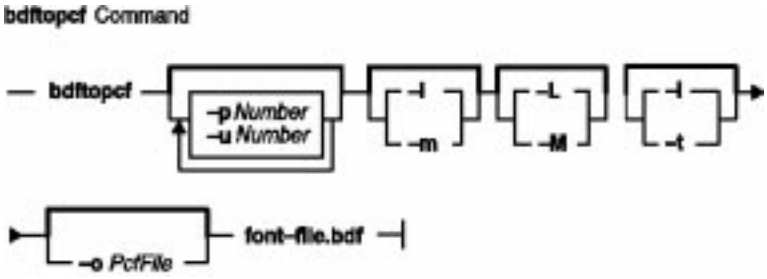
The **awk** command, **dc** command.

bdftopcf Command

Purpose

Converts fonts from Bitmap Distribution Format (bdf) to Portable Compiled Format (pcf).

Syntax



bdftopcf [**−i** | **−t**] [**−p** *Number*] [**−u** *Number*] [**−l** | **−m**] [**−L** | **−M**] [**−o** *PcfFile*] **font−file.bdf**

Description

The **bdftopcf** command is the font compiler which converts fonts from Bitmap Distribution Format to Portable Compiled Format. Fonts in Portable Compiled Format can be read by any architecture, although the file is structured to allow one particular architecture to read them directly without reformatting. This feature allows fast reading on the appropriate machine. In addition, the files remain portable to other machines, although they are read more slowly.

Flags

−p *Number* Sets the font glyph padding. Each glyph in the font has each scanline padded into a multiple of bytes specified by the *Number* variable, where *Number* is the value of 1, 2, 4, or 8 bytes.

−u *Number* Sets the font scanline unit. When the font bit order is different from the font byte order, the *Number* variable describes what units of data (in bytes) are to be swapped. The *Number* variable can be the value of 1, 2, or 4 bytes.

−m Sets the font bit order to MSB (most significant bit) first. Bits for each glyph are placed in this order. Thus, the left−most bit on the screen is the highest valued bit in each unit.

−l (lowercase L) Sets the font bit order to LSB (least significant bit) first. The left−most bit on the screen is the lowest valued bit in each unit.

−M Sets the font byte order to MSB (most significant byte) first. All multibyte data in the file, including metrics and bitmaps, are written most significant byte first.

−L Sets the font byte order to LSB (least significant byte) first. All multibyte data in the file, including metrics and bitmaps, are written least significant byte first.

−t Converts fonts into *terminal* fonts whenever possible. A terminal font has each glyph image padded to the same size. The Xserver can usually render these font types more quickly.

−i Inhibits the normal computation of ink metrics. When a font has glyph images that do not fill the bitmap image because the ``on'' pixels do not extend to the edges of the metrics, the **bdftopcf** command computes the actual ink metrics and places them in the **.pcf** file.

Note: The **−t** option inhibits the behavior of this flag.

−o *PcfFile* Specifies the name of an output file. By default, the **bdftopcf** command writes the **pcf** file to

standard output.

Examples

1. To convert fonts into terminal fonts whenever possible, enter:

   ```
   bdftocpf −t font−file.bdf
   ```

2. To set the glyph padding to a multiple of 4 bytes, enter:

   ```
   bdftocpf −p 4 font−file.bdf
   ```

Related Information

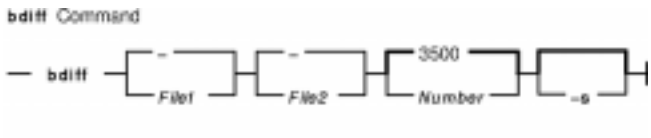
Calls and Functions Reference in *Technical Reference Volumes 1−3: Base Operating System and Extensions* .

bdiff Command

Purpose

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

Syntax



bdiff { *File1* | **−** } { *File2* | **−** } [*Number*] [**−s**]

Description

The **bdiff** command compares the files specified by the *File1* and *File2* parameters and writes information about their differing lines to standard output. If either file name is **−** (minus), the **bdiff** command reads standard input. The **bdiff** command is used like the **diff** command to find lines that must be changed in two files to make them identical. The primary purpose of this command is to permit processing of files that are too large for the **diff** command.

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

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

Flags

−s Suppresses error messages from the **bdiff** command. (Note that the **−s** flag does not suppress error messages from the **diff** command).

Examples

To display the differences between the `chap1` file and the `chap1.bak` file:

```
bdiff chap1 chap1.bak
```

Files

/usr/bin/bdiff Contains the **bdiff** command.

Related Information

The **diff** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

bellmail Command

Purpose

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

Syntax

To Display Messages

bellmail [**−e**] [**−f***File*] [**−p**] [**−q**] [**−r**]

To Send Messages

bellmail [**−t**] *User* ...

Description

The **bellmail** command with no flags writes to standard output, one message at a time, all stored mail addressed to your login name. 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 **bellmail** subcommands to control the disposition of the message.

Use the *User* parameter to attach a prefix to messages you send. The **bellmail** command prefaces each message with the sender's name, date and time of the message (its postmark), and adds the message to the user's mailbox. Specify the *User* parameter by pressing End Of File (the Ctrl−D key sequence) or entering a line containing only a . (period) after your message.

The action of the **bellmail** command can be modified by manipulating the **/var/spool/mail/***UserID* mailbox file in two ways:

- The default permission assignment for *others* is `all permissions denied (660)`. You may change this permission to `read/write`. When you change permissions from the default, the system preserves the file, even when it is empty, to maintain the desired permissions. You can no longer remove the file.
- You can edit the file to contain as its first line:

  ```
  Forward to person
  ```

This instruction causes all messages sent to the *User* parameter to be sent to the *Person* parameter instead. The `Forward to` feature is useful for sending all of a person's mail to a particular machine in a network

environment.

To specify a recipient on a remote system accessible through Unix−to−Unix Copy Program (UUCP), preface the *User* parameter with the system name and an ! (exclamation mark). The
[**−t**] *User*. . .**uucp** command contains additional information about addressing remote systems.

> **Note:** In order to use the remote mail function, UUCP must be completely configured.

If you are interested in writing your own third−party mail program, you may need to know the following locking mechanisms used by the **bellmail** command.

1. The **bellmail** command creates a *UserID***.lock** file in the **/var/spool/mail** directory that is opened by passing the **O_NSHARE** and **O_DELAY** flags to the **open** subroutine. If the *UserID***.lock** file is being held, your **bellmail** process sleeps until the lock is free.
2. The **bellmail** command locks **/var/spool/mail/***UserID* with the **lockf** subroutine.

Flags

−e Does not display any messages. This flag causes the **bellmail** command to return an exit value of 0 if the user has mail, or an exit value of 1 if there is no mail.

−f*File* Reads mail from the named *File* parameter instead of the default mail file, **/var/spool/mail/***UserID*.

−p Displays mail without prompting for a disposition code. This flag does not delete, copy, or forward any messages.

−q Causes the **bellmail** command to exit when you press Interrupt (the Ctrl−C key sequence). Pressing Interrupt (Ctrl−C) alone stops only the message being displayed. (In this case, the next message sometimes is not displayed until you enter the **p** subcommand.)

−r Displays mail in first−in, first−out order.

−t Prefaces each message with the names of all recipients of the mail. (Without this flag, only the individual recipient's name displays as addressee.)

The *User* parameter is a name normally recognized by the **login** command. If the system does not recognize one or more of the specified *User* parameters or if the **bellmail** command is interrupted during input, the **bellmail** command tries to save the message in the **dead.letter** file in the current directory. If the **bellmail** command cannot save the message to the **dead.letter** file, it saves the message in the **$HOME/dead.letter** file. Once in this file, the message can be edited and sent again.

> **Note:** The **bellmail** command uses the **$MAIL** environment variable to find the user's mailbox.

Subcommands

The following subcommands control message disposition:

+ Displays the next mail message (the same as pressing the Enter key).

− Displays the previous message.

!*Command* Runs the specified workstation command.

***** Displays a subcommand summary.

d Deletes the current message and displays the next message.

m *User* Forwards the message to the specified *User* parameter.

p Displays the current message again.

q Writes any mail not yet deleted to the **/var/spool/mail/***UserID* file and exits. Pressing End Of File (Ctrl−D) has the same effect.

s [*File*] Saves the message in the named *File* parameter instead of in the default mail file, **$HOME/mbox**.

w [*File*] Saves the message, without its postmark, in the specified *File* parameter instead of in the default mail file, **$HOME/mbox**.

x Writes all mail unchanged to **/var/spool/mail/***UserID* and exits.

Examples

1. To send mail to other users, enter:

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

   Press Ctrl−D at the end of the message. In this example, the system mails the message to users `tom` and `rachel`.

2. To send a file to another user, enter:

   ```
   bellmail lance <proposal
   ```

   In this example, the file `proposal` is sent to user `lance`.

3. To display your mail, enter:

   ```
   bellmail
   ```

   After the most recent message is displayed, a ? (question mark) indicates the **bellmail** command is waiting for one of the **bellmail** subcommands. Enter `help` or an * (asterisk) to list the subcommands available.

4. To save a message or a file to the default mail file, enter:

   ```
   bellmail
   ```

   This command displays each message mailed to you. Press the Enter key after the ? prompt until the desired file is displayed. When the appropriate file is displayed, enter:

   ```
   s
   ```

   In this example, the file is saved in the default mail file, **$HOME/mbox**.

5. To save a message or a file to a specific file, enter:

   ```
   bellmail
   ```

   This command displays each message mailed to you. Press the Enter key after the ? prompt until the desired file is displayed. When the appropriate file is displayed, enter:

   ```
   s mycopy
   ```

In this example, the file is saved in a file named `mycopy`, instead of in the default mail file.

Files

$HOME/dead.letter	Unmailable text.
$HOME/mbox	Your personal mailbox.

/usr/mail/*.lock	Lock for mail directory.
/var/spool/mail/*UserID*	Default system mailbox for *UserID*.
/usr/bin/bellmail	Bellmail program.

Related Information

The **mail** command, **uucp** command.

The **lockfx, lockf, or flock** subroutine, **open, openx, or creat** subroutine.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

Organizing Mail in *AIX Version 4.3 System User's Guide: Communications and Networks*.

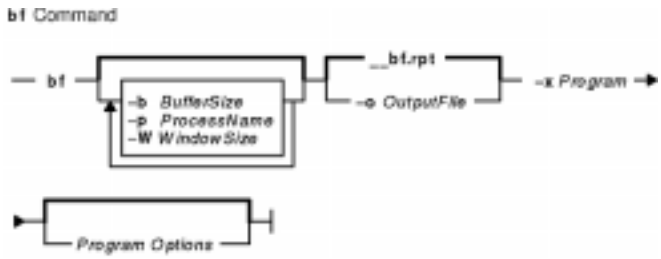
Editors Overview in *AIX INed Editor User's Guide*.

bf Command

Purpose

Analyzes the memory usage of applications.

Syntax



bf [**−b** *BufferSize*] [**−W** *WindowSize*][**−o** *OutputFile*] [**−p** *ProcessName*]{ **−x** *Program* [*Program Options*]}

Description

The **bf** (bigfoot) command traces the memory use of the applications. It provides a detailed trace, or *footprint*, of the memory page references for most processes on the system. The **bf** command captures references to all unpinned, and many pinned, pages.

> **Note:** The **bf** command tool currently only supports the operation of non−threaded 32−bit applications in AIX Version 4.3. No 64−bit application support exists.

The **bf** command turns page tracing on, runs the specified program, then turns page tracing off when the program completes. The **bf** command creates the **__bf.rpt** file, by default, in the current directory. The file contains unprocessed page−trace data gathered while page tracing was on. Use the **−o***OutputFile* option to specify a file name other than **__bf.rpt**.

Use the **bfrpt** command to postprocess the **bf** command's output.

You have various ways to limit the page−trace data captured in the **__bf.rpt** file.

- Use the **−b** flag to control the size of the pinned buffer that the **bf** command uses to store page−trace data.
- Use the **−p** flag to gather data only about the process specified in the *ProcessName* parameter. This flag causes the **bf** command to ignore processes with names other than *ProcessName*.
- Use the **−W** flag to provide another means of limiting captured page−trace data. This flag causes the **bf** command to capture repeated references to the same page but only logs the reference if the page's address is not already within the specified window of page references.

  For example, given a *WindowSize* of 3, and pages A, B, C, and D referenced in the following sequence: A B A C D A, the steps performed are:

  1. The window's contents is initially empty.
     ```
     [ * * * ]
     ```

  2. Page reference A is captured and logged.

```
[ * * A ]
```

3. Page reference B is captured and logged.
   ```
   [ * A B ]
   ```

4. Page reference A is captured but not logged because A is already in the window.
   ```
   [ * A B ]
   ```

5. Page reference C is captured and logged.
   ```
   [ A B C ]
   ```

6. Page reference D is captured, bumps A out, and is logged.
   ```
   [ B C D]
   ```

7. Page reference A is captured, bumps B out and is logged
   ```
   [ C D A ]
   ```

Therefore, the page−trace contains ABCDA not ABACDA. The second reference to A is not logged because A was within the window at the time of the reference, however, the third reference to A is logged.

> **Note:** The **bf** command causes a substantial reduction in system performance. Do not use it while doing any performance critical work

Flags

−b *BufferSize*
: Specifies the number of records to store in the BigFoot kernel buffer. If the buffer size is not large enough and an overflow occurs, the last record contains the process name "Buffer.Overflow."

−o*OutputFile*
: Specifies the name of the file to store the output in. The default is the **__bf.rpt** file.

−p*ProcessName*
: Filters out all processes other than those with the specified process name. The BigFoot kernel buffer stores only records of processes with the specified name. Specifying a process name reduces the size of the kernel buffer.

−W *WindowSize*
: Specifies the range of unpinned pages a page reference must be to a new (unique) page in order to be logged. The minimum size is 2 and the maximum size is 100. The default size is 10.

−x*Program*
: Specifies the program to run. This flag is required and must be the last argument on the **bf** command line. The program is forked as a child process. The **bf** command continues collecting page reference data until the child program ends or the buffer overlows.

Security

Access Control: You must have root authority to run this command.

Examples

1. To run the process, `/bin/ps −eaf`, and monitor page references until the process completes, enter:
   ```
   bf −b 10000  −p ps  −x /bin/ps −eaf
   ```

   The **−p** flag instructs the **bf** command to filter out all processes that are not named **ps** and the **−x** flag specifies the process to run.

2. To run the program **/directory/path/memory_prog** enter:

```
bf −b 80000  −x /directory/path/memory_prog
```

The **bf** command reports on all detected processes that referenced pages because filtering was not specified.

3. To fork the child processes `sleep 20`, wait until the sleep completes before turning off the page monitoring, and record the output in the file **bigfoot.rpt**, enter:
```
bf −b 12000 −o ./bigfoot.rpt  −x sleep 20
```

Files

/usr/bin/bf
Contains the **bf** command.

Related Information

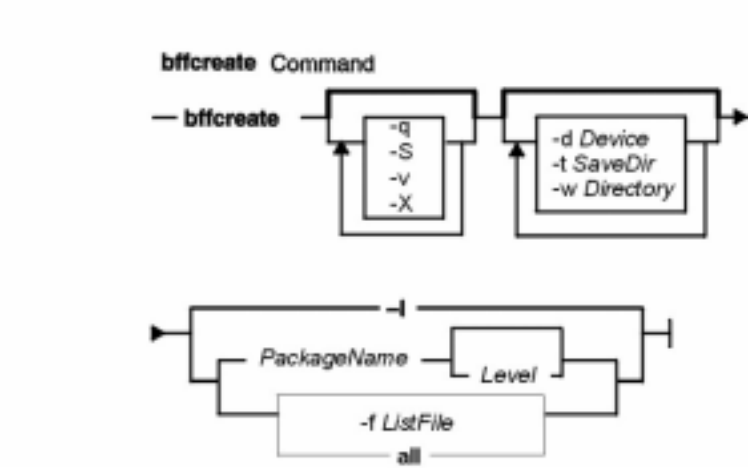
The **bfrpt** command, **rmss** command, and the **svmon** command.

bffcreate Command

Purpose

Creates installation image files in backup format.

Syntax



bffcreate [**−q**] [**−S**] [**−v**] [**−X**] [**−d***Device*] [**−t** *SaveDir*] [**−w***Directory*] { **−l** | *PackageName* [*Level*] ... | **−f** *ListFile* | **all** }

Description

The **bffcreate** command creates an installation image file in backup file format (bff) to support software installation operations.

The **bffcreate** command creates an installation image file from an installation image file on the specified installation media. Also, it automatically creates an installation image file from hyptertext images (such as those on the AIX documentation CD−ROMs). The **installp** command can use the newly created installation file to install software onto the system. The file is created in backup format and saved to the directory specified by *SaveDir*. The **.toc** file in the directory specified by the *SaveDir* parameter is updated to include an entry for the image file.

The **bffcreate** command determines the bff name according to this information:

Image Type	Target bff Name
Installation image	*package.v.r.m.f.I*
3.1 update	*package.v.r.m.f.service #*
3.2 update	*package.v.r.m.f.ptf*
*4.X update	*fileset.part.v.r.m.f.U*

*4.X updates contain one and only one *fileset*. In addition, Version 4 updates do not contain *ptf* ids.

package = the name of the software package as described by the *PackageName* parameter

v.r.m.f = version.release.modification.fix, the level associated with the software package. The

PackageName is usually not the same as the *fileset* name.

ptf = program temporary fix ID (also known as FixID)

The installation image file name has the form *Package.Level.I*. The *Package* is the name of the software package, as described for the *PackageName* parameter. *Level* has the format of *v.r.m.f*, where *v* = version, *r* = release, *m* = modification, *f* = fix. The *I* extension means that the image is an installation image rather than an update image.

Update image files containing an AIX 3.1 formatted update have a service number extension following the level. The *Servicenum* parameter can be up to 4 digits in length. One example is `xlccmp.3.1.5.0.1234`.

Update image files containing an AIX 3.2 formatted update have a *ptf* extension following the level. One example is `bosnet.3.2.0.0.U412345`.

AIX Version 4 update image file names begin with the *fileset* name, not the *PackageName*. They also have *U* extensions to indicate that they are indeed update image files, not installation images. One example of an update image file is `bos.rte.install.4.3.2.0.U.`

The **all** keyword indicates that installation image files are created for every installable software package on the device.

You can extract a single update image with the AIX Version 4 **bffcreate** command. Then you must specify the *fileset* name and the *v.r.m.f.* parameter. As in example 3 in the Examples section, the *PackageName* parameter must be the entire *fileset* name, `bos.net.tcp.client`, not just `bos.net`.

> **Attention:** Be careful when selecting the target directory for the extracted images, especially if that directory already contains installable images. If a *fileset* at a particular level exists as both an installation image and as an update image in the same directory, unexpected installation results can occur. In cases like this, **installp** selects the image it finds first in the table of contents (**.toc**) file. The image it selects may not be the one you intended and unexpected requisite failures can result. As a rule of thumb, you should extract maintenance levels to clean directories.

Flags

−d*Device*	Specifies the name of the device where the original image resides. The device can be a CD, tape, diskette, or a directory. If the image is contained on tape, the tape device must be specified as no−rewind−on−close and no−retention−on−open (**/dev/rmt*.1** for high−density tape and **/dev/rmt*.5** for low−density tape). The default device is /**dev**/**rfd0**.
−f*ListFile*	Reads a list of *PackageNames* and *Levels* from *ListFile*. *PackageNames*, each optionally followed by a level, should appear one per line of text. Any text following the second set of spaces or tabs on a line is ignored.
−l	Lists the *Package*, *Level*, *Image Type* (*I* for installation images and *U* for update images), and *Part(s)* of all packages on the media.
−q	Suppresses the request for media.
−t *SaveDir*	Specifies the directory where the installation image files are to be created. The **bffcreate** command creates the specified directory if it does not exist. If the **−t** flag is not specified, the files are saved in the **/usr/sys/inst.images** directory.
−v	Writes the name of the backup format file to standard output.
−w*Directory*	Specifies the directory where a temporary working directory can be created. The **bffcreate** command creates the specified directory if it does not exist. The default directory is **/tmp**.
−S	Suppresses multiple volume processing when the installation device is a CD−ROM.

Installation from a CD−ROM is always treated as a single volume, even if the CD−ROM contains information for a multiple volume CD set. This same suppression of multiple volume processing is performed if the **INU_SINGLE_CD** environment is set.

−X Automatically extends the file system if space is needed.

Security

Access Control: You must have root authority to run this command.

Examples

1. To create an installation image file from the **bos.net** software package on the tape in the **/dev/rmt0** tape drive and use **/var/tmp** as the working directory, enter:

   ```
   bffcreate −d /dev/rmt0.1
   −w /var/tmp bos.net
   ```

2. To create an installation image file from the **package** software package on the diskette in the **/dev/rfd0** diskette drive and print the name of the installation image file without being prompted, enter:

   ```
   bffcreate −q −v
   package
   ```

3. To create a single update image file from the **bos.net.tcp.client** software package on the CD in **/dev/cd0**, enter:

   ```
   bffcreate −d
   /dev/cd0 bos.net.tcp.client 4.2.2.1
   ```

4. To list the packages on the CD in **/dev/cd0**, enter:

   ```
   bffcreate −l
   −d /dev/cd0
   ```

5. To create installation and/or update images from a CD in **/dev/cd0** by specifying a list of *PackageNames* and *Levels* in a *ListFile* called my MyListFile, enter:

   ```
   bffcreate −d /dev/cd0
   −f MyListFile
   ```

Files

/usr/sbin/bffcreate	Contains the **bffcreate** command.
/usr/sys/inst.images	Contains files in backup format for use in installing or updating a complete set or subset of software packages.
/usr/sys/inst.images/.toc	The table of contents file for the default directory where a list of installation image files in the directory is maintained.

Related Information

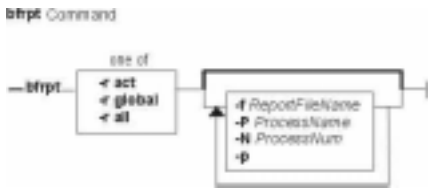
The **installp** command, **inutoc** command.

bfrpt Command

Purpose

Reports on the memory usage of applications.

Syntax



bfrpt{ **−ract** | **global** | **all** } [**−f** *ReportFileName*] [**−P** *ProcessName*] [**−N** *ProcessNum*] [**−p**]

Description

The **bfrpt** command filters **bf** page−trace files for two types of post processing. The first type reflects the global state of the system. The second type reflects the page−referencing patterns of a particular process.

The **bfrpt** command uses the **global** filter to provide a comprehensive view of the system. The **global** filter calculates the overall footprint of each process found in the bf report.

The **bfrpt** command uses the **Address Correlation Technology** filter, or **act** filter, to analyze **bf** data with emphasis on a single process. The **act** filter translates a system, library, or process address into a symbolic routine name.

The **bfrpt** command formats output in tabular form as the default, but with the **−p** flag you can also record the output in a PostScript file.

Global Reports

The **bfrpt** command's **global** reports consist of header information and tables. The header information is a brief reminder to the user about the properties of segment registers. AIX maps 32−bit virtual memory addresses into 16 segments. Four of the 32 bits select one of the 16 bit segment registers. The remaining 28 bits give an offset within the segment. Each segment register contains a 24−bit segment ID, which, when prefixed to the 28−bit segment offset, forms a full 52−bit virtual address.

Header Information

The header information in a **global** report lists the abbreviations assigned to the 16 segment registers:

0: Kernel
1: Exec Program (Process text)
2: Private Read−Write (Process Data)
3−12: Addressable Files and Shared Memory Segments
13: Shared Library Text
14: Kernel
15: Shared Library Data

Note: Though the header information is accurate for most AIX processes, the system can use some segment registers for different purposes. Use the header information as a guideline only.

Non−kernel programs have read−only access to the two kernel segments, **0** and **14**, and the shared library segment, **13**. Segment **1** contains the executable of the current program. Other instances of the same program share the read−only contents of segment **1**.

Segment **2** contains the private read−write data space of the current program. Program variables, allocated memory, and the program stack come from the private segment of the process. In rare instances, library text resides in segment **2**.

Segment **13** is readable by all users. Therefore, shared libraries with file permissions that do not include read access for all other users on the system are not loaded in segment **13**. AIX loads libraries without global read access in the private data segment of each process that uses them.

Segment **15** contains the private shared library data for a process. The ten remaining segments, **3** through **12**, provide memory access to files and shared memory segments. The system maps files into virtual memory segments when a file is first opened.

Tables

Three tables follow the header information:

- **Process Name Table**
- **Process Name−Process ID Table**
- **Unique Pages Table**

The first table, the Process Name Table, contains the number of pages referenced by each process name. If the **bfrpt** command reports two or more processes with the same process name, it treats them logically as one process in the Process Name Table. The rows are sorted by the number of referenced pages.

```
************************************************************
*                                                          *
* Number of Pages Referenced by <Process Name>             *
*                                                          *
************************************************************
PNAME (20 chars)    TOTAL     0  1   2  3−12    13  14  15
             ksh      331   135 39  70     4    30   0  53
            mont      137    90  5  13     0    14   0  15
          rlogind       45    30  4   4     0     3   0   4
```

The second table, the Process Name−Process ID Table, lists each process in the **bfrpt** report on a separate row. Because AIX can reuse process IDs, they are combined with process names to distinguish all processes in the report.

```
************************************************************

*                                                          *

* Number of Pages Referenced by <Process Name, PID>        *

*                                                          *

************************************************************

PNAME (20 chars)    PID  TOTAL      0      1      2   3−12     13     14     15

           ksh    10600    247    119     26     35      1     29      0     37
```

```
          ksh    10598    206    101     21     30      1     24      0     29

          ksh    11336     99     47     33      5      2     11      0      1

         mont    10598    137     90      5     13      0     14      0     15

       rlogind     9354     45     30      4      4      0      3      0      4
```

The third table, the Unique Pages Table, contains the total *unique* pages referenced by each distinct process. A unique page is a page referenced by only one process. A page referenced by more than one process is a *shared* page. The system examines the full 52−bit virtual address, not just 32 bits, to determine the uniqueness of a page.

```
************************************************************
*                                                          *
* Number of Unique Pages Referenced by <Process Name, PID>  *
*                                                          *
************************************************************
NAME (20 chars)       PID     TOTAL

           ksh     11336        99
           ksh     10598        78
           ksh     10600        24
          mont     10598        20
       rlogind      9354        15
```

act Reports

The **act** reports provide information about the footprints of each routine in a process. Each **act** report comprises four logical sections:

- **Total Footprint**
- **Footprint Before Start**
- **Footprint After Stop**
- **Unique Data Footprint**

The **Total Footprint** section is divided into three tables. The first table contains the footprints for all routines other than the shared library and kernel routines. The second table contains footprints for the shared library routines that the main program uses. The third table contains the footprints for the kernel routines that operate on behalf of the process.

The **Footprint Before Start** and **Footprint After Stop** sections provide footprints for the kernel routines that operate either prior to the start or after the end of the main program.

The **Unique Data Footprint** section identifies footprints for unique data segments in the main program's routines. If two or more routines touch the same page, the page is shared, not unique.

Flags

−f*ReportFileName*	The name of the BigFoot page−trace file. Default is **__bf.rpt**.
−N*ProcessNum*	Number of processes to get from the page−trace file. Runs **act** on this number of processes. Default is top two processes.
−P*ProcessName*	Run **act** reports only on the specified process. The top two processes have **act** filtering done by default. See **−N** option.
−p	Creates reports in PostScript format. Files named **processname.ps** are **act** reports and files named **global.ps** are **global** reports.
−r*ReportType*	Specifies the type of report to create. The valid values, which are mutually exclusive, for *ReportType* are: **act**, **global**, and **all** (to create both **act** and **globalpp** reports). This is a

required flag.

Security

Access Control: You must have root authority to run this command.

Examples

1. To produce all reports in tabular form, enter:
   ```
   bfrpt -r all
   ```
2. To produce an **act** report on processes only named `example`, enter:
   ```
   bfrpt -r act -P example
   ```
3. To produce all reports but run **act** report on processes only named `example`, enter:
   ```
   bfrpt -r all -P example
   ```
4. To produce **act** reports on the 5 largest processes, enter:
   ```
   bfrpt -r act -N5
   ```

Files

/usr/bin/bfrpt
Contains the **bfrpt** command.

Related Information

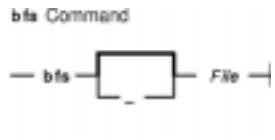
The **rmss** command, **svmon** command, and the **bf** command.

bfs Command

Purpose

Scans files.

Syntax



bfs [**−**] *File*

Description

The **bfs** command reads a file specified by the *File* parameter, but does not process the file. You can scan the file, but you cannot edit it.

The **bfs** command is basically a read−only version of the **ed** command with two exceptions: the **bfs** command can process much larger files and has additional subcommands.

Input files can be up to 32,767 lines long, with up to 255 characters per line. 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 in identifying sections of a large file that can be divided, using the **csplit** command, into more manageable pieces for editing.

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

The **bfs** command runs in both single− and multi−byte environments. The language environment is determined by the setting of the **LANG** environment variable (in the **/etc/environment** file) for the shell.

Forward and Backward Searches

The **bfs** command supports all of the address expressions described under the **ed** command. 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 continues searching from the beginning of the file after it reaches the end of the file. If you specify a backward search with wraparound, the command 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−matching routine of the **bfs** command differs somewhat from the one used by the **ed** command and includes additional features described in the **regcmp** subroutine. 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 file sizes. 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 in the **ed** command. However, the **bfs** command does not support a space between the address and the subcommand. Subcommands such as **−−**, **+++−**, **+++=**, **−12**, and **+4p** are accepted. **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 (the **xo**, **xt**, and **xc** subcommands, respectively). *Compressed Output* mode suppresses blank lines and replaces multiple spaces and tabs with a single space.

The following additional subcommands are available:

xf *File*

Reads the **bfs** subcommands from the specified 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 can 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 newline character. Spaces between the **:** (colon) and the start of the label are ignored. This subcommand can be used to insert comments into a subcommand file, since labels need not be referenced.

[*Address1*[**,***Address2*]]**xb**/*Pattern*/*Label*

Sets the current line to the line containing the specified pattern, and jumps to the specified label in the current command file if the pattern is matched within the designated range of lines. The jump fails under any of the following conditions:

- The value of either the *Address1* or *Address2* parameter is not between the first and last lines of the file.
- The *Address2* value is less than the *Address1* value.
- 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** subcommand and the null subcommands to the number of characters. The default value of the *Number* parameter is 192.

xv[*Digit*] [*Value*] Assigns the specified *Value* to the *Digit* parameter. The value of the *Digit* parameter can be 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 the *Value* parameter an !

(exclamation point), followed by the command name. For example:

```
xv5 !cat junk
```

stores the first line of the `junk` file 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 the specified label in the current command file if the value is 0.

xbn *Label* — Tests the last saved exit value from a shell command and jumps to the specified label in the current command file if the value is not 0.

xc [*Switch*] — Turns compressed output mode on or off. (Compressed output mode suppresses blank lines and replaces multiple spaces and tabs with a single space.)

If the *Switch* parameter has a value of 1, output from the **p** subcommand and the null subcommands is compressed. If the *Switch* parameter is 0, this output is not compressed. If you do not specify a value for the *Switch* parameter, the current value of the *Switch* parameter, initially set to 0, reverses.

Files

/usr/bin/bfs Contains the **bfs** command.

Related Information

The **csplit** command, **ed** or **red** command.

The **environment** file.

The **regcmp** or **regex** subroutine.

File and Directory Access Modes in *AIX Version 4.3 System User's Guide: Operating System and Devices* introduces file ownership and permissions to access files and directories.

Directory Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes files, file types, and how to name files.

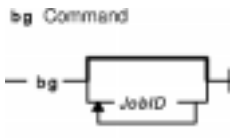
Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output.

bg Command

Purpose

Runs jobs in the background.

Syntax



bg [*JobID* ...]

Description

If job control is enabled (see "Job Control in the Korn Shell" in *AIX Version 4.3 System User's Guide: Operating System and Devices*), the **bg** command resumes suspended jobs in the current environment by running them as background jobs. If the specified job is already running in the background, the **bg** command has no effect and exits successfully. If no *JobID* parameter is supplied, the **bg** command uses the most recently suspended job.

The *JobID* parameter can be a process ID number, or you can use one of the following symbol combinations:

%*Number*	Refers to a job by the job number.
%*String*	Refers to a job whose name begins with the specified string.
%?*String*	Refers to a job whose name contains the specified string.
%+ OR **%%**	Refers to the current job.
%−	Refers to the previous job.

Using the **bg** command to place a job into the background causes the job's process ID to become known in the current shell environment. The **bg** command output displays the job number and the command associated with that job. The job number can be used with the **wait**, **fg**, and **kill** commands by prefixing the job number with a % (percent sign). For example, **kill %3**.

A job is suspended by using the **Ctrl−Z** key sequence. That job can be restarted in the background using the **bg** command. This is effective if the job expects no terminal input and if job output is redirected to non−terminal files. If a background job has terminal output, the job can be forced to stop by entering the following command:

```
stty tostop
```

A background job can be stopped by entering the following command:

```
kill −s stop JobID
```

The **/usr/bin/bg** command does not work when operating in its own command execution environment, because that environment does not have suspended jobs to manipulate. This would be the case in the following example:

```
Command | xargs bg
```

Each **/usr/bin/bg** command operates in a different environment and does not share the parent shell's understanding of jobs. For this reason, the **bg** command is implemented as a Korn shell or POSIX shell regular built−in.

Exit Status

The following exit values are returned:

0 Successful completion.
>0 An error occurred.

If job control is disabled, the **bg** command exits with an error, and no job is placed in the background.

Examples

If the output of the **job** command displays the following stopped job:

```
[2] +   Stopped (SIGSTOP)   sleep 100 &
```

use the job number to resume the `sleep 100 &` command by entering:

```
bg %2
```

The screen displays the revised status of job `2`:

```
[2] sleep 100 &
```

Files

/usr/bin/ksh Contains the Korn shell **bg** built−in command.
/usr/bin/bg Contains the **bg** command.

Related Information

The **csh** command, **fg** command, **jobs** command, **kill** command, **wait** command.

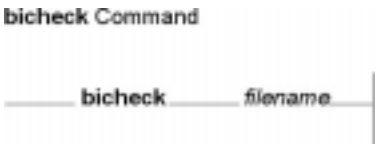
Job Control in the Korn Shell in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

bicheck Command

Purpose

Syntax checker for user−modified **bosinst.data** files.

Syntax



bicheck*Filename*

Description

The **bicheck** command checks for the existence of the control flow, target_disk_data, and locale stanzas in the **bosinst.data** file. The parameter *Filename* indicates the **bosinst.data** file you want to verify. The value−−if not blank−−for each field in a stanza is confirmed to match an allowable value, if possible, and checked for length limitations and/or other possible limitations.

If a non−prompted install is specified, the existence of values for required fields is confirmed.

If a dump stanza exists and if the value is not blank, the value is determined to match an allowable value, if possible. It is also checked for length limitations and/or other possible limitations.

The **bicheck** command does not stop after the first error, but continues to list all problems it finds with the given **bosinst.data** file. All error messages are sent to standard error.

Exit Status

This command returns the following exit values:

0 Successful completion.
1 An error occurred.

Files

/usr/lpp/bosinst/bicheck contains the **bicheck** command.

Related Information

The **mksysb** command.

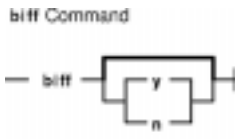
Chapter 4, Customizing the BOS Installation Guide, *AIX Version 4.3 Installation Guide*

biff Command

Purpose

Enables or disables mail notification during the current session.

Syntax



biff [**y** | **n**]

Description

The **biff** command informs the system whether you want to be notified when mail arrives. When mail notification is enabled, From and Subject header lines and the first 7 lines or 560 characters of a message are displayed on the screen when mail arrives. Notification, specified by the **biff y** command, is often included in the **$HOME/.login** or **$HOME/.profile** file to be executed each time the user logs in. The **biff n** command disables notification.

> **Note:** In addition to **y** and **n**, you can use **yes** and **no** to enable and disable mail notification.

The **biff** command operates asynchronously. To receive notification when mail arrives, ensure:

1. The message permission setting is on in your shell (`mesg y`).
2. **comsat** is running (started by the **inetd** daemon).
3. Notification is enabled (`biff y`).

For synchronous notification, use the **MAIL** variable of either the **ksh** command, **bsh** command, or the **csh** command.

Options

y Enables mail notification.

n Disables mail notification.

Examples

1. To display the current setting, enter:

   ```
   biff
   ```

2. To be notified during the current terminal session whenever mail arrives, enter the following statement in your **$HOME/.login** or **$HOME/.profile** file:

   ```
   biff y
   ```

The From and Subject header lines and the first seven lines or 560 characters of the message will be displayed on the screen when mail arrives.

Files

$HOME/.login	Read by **login** shell at login.
$HOME/.profile	Controls start−up processes and daemons.
/usr/bin/biff	Contains **biff** command.

Related Information

The **bsh** command, **csh** command, **ksh** command, **mail** command.

The **comsat** daemon.

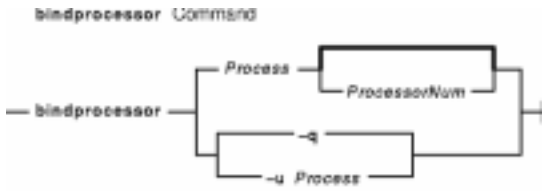
Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

bindprocessor Command

Purpose

Binds or unbinds the kernel threads of a process to a processor.

Syntax



bindprocessor *Process* [*ProcessorNum*] | **−q** | **−u***Process*

Description

The **bindprocessor** command binds or unbinds the kernel threads of a process, or lists available processors. The *Process* parameter is the process identifier of the process whose threads are to be bound or unbound, and the *ProcessorNum* parameter is the logicial processor number of the processor to be used. If the *ProcessorNum* parameter is omitted, the process is bound to a randomly selected processor.

It is important to understand that a process itself is not bound, but rather its kernel threads are bound. Once kernel threads are bound, they are always scheduled to run on the chosen processor, unless they are later unbound. When a new thread is created, it has the same bind properties as its creator. This applies to the initial thread in the new process created by the **fork** subroutine: the new thread inherits the bind properties of the thread which called **fork**. When the **exec** subroutine is called, thread properties are left unchanged.

The **−q** flag of the **bindprocessor** command lists the available logical processor numbers: you can use the logical numbers given as values for the *ProcessorNum* parameter. The **−u** flag unbinds the threads of a process, allowing them to run on any processor.

Notes:

1. The **bindprocessor** command is meant for multiprocessor systems. Although it will also work on uniprocessor systems, binding has no effect on such systems.
2. You need root authority to bind or unbind threads in processes you do not own.

Flags

−q Displays the processors which are available.

−u Unbinds the threads of the specified process.

Examples

1. To see which processors are available (possible *ProcessorNum* values*)*, enter:

   ```
   bindprocessor −q
   ```

   For a four processor system, the output is similar to:

```
The available processors are: 0 1 2 3
```

2. To bind the threads in process 19254 to processor 1, enter:

```
bindprocessor 19254 1
```

File

/usr/sbin/bindprocessor Contains the **bindprocessor** command.

Related Information

The **cpu_state** command, **smit** command.

The **bindprocessor** subroutine. **exec** subroutine, **fork** subroutine.

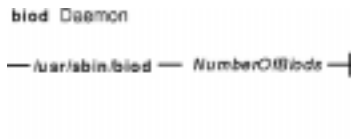
Controlling Processor Use in *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

biod Daemon

Purpose

Handles client requests for files.

Syntax



/usr/sbin/biod *NumberOfBiods*

Description

The **biod** daemon runs on all NFS client systems. When a user on a client wants to read or write to a file on a server, the **biod** daemon sends this request to the server. The **biod** daemon is activated during system startup and runs continuously.

The *NumberOfBiods* parameter allows the user to specify the number of block I/O daemons to start. The user assigns the number of daemons based on the load the client can handle. Six to eight daemons can handle an average load. You must run at least one daemon for NFS to work.

To change the number of daemons started when using the System Resource Controller (SRC) commands, use the **chnfs** command. In general, to change the parameters of an SRC−controlled daemon, use the **chssys** command. For more information on SRC commands, see "Controlling NFS" in *AIX Version 4.3 System Management Guide: Communications and Networks.*

The **biod** daemons should be started and stopped with the following SRC commands:

```
startsrc −s biod
```

```
stopsrc −s biod
```

Examples

1. To start **biod** daemons using an **src** command, enter:

   ```
   startsrc −s biod
   ```

   In this example, the `startsrc−sbiod` command starts the number of specified daemons.

2. To change the number of daemons running on your system, enter:

   ```
   chssys −s biod −a 6
   ```

In this example, the `chssys` command changes the number of **biod** daemons running on your system to six.

Files

/etc/rc.nfs Contains the startup script for the NFS and NIS daemons.

Related Information

The **chnfs** command, **chssys** command, **mount** command.

The **mountd** daemon, **nfsd** daemon.

How to Mount a File System Explicitly in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

List of NFS Commands.

bj Command

Purpose

Starts the blackjack game.

Syntax

bj

Description

The **bj** command invokes the blackjack game. Blackjack is a card game. The object of blackjack is to be dealt cards with a value of up to but not over 21 and to beat the dealer's hand. The computer plays the role of the dealer in blackjack.

You place bets with the dealer on the likelihood that your hand will come equal or closer to 21 than will the dealer's. The following rules apply to betting.

The bet is two dollars every hand. If you draw a natural blackjack, you win three dollars. If the dealer draws a natural blackjack, you lose two dollars. If you and the dealer both have natural blackjacks, 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 blackjack, winning two dollars if the dealer has a natural blackjack and losing one dollar 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 two dollars on each hand. If the value of your original hand is 10 or 11, you can double down, that is, double the bet to four dollars and receive exactly one more card in that hand.

Under normal play, you can draw a card (take a 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 draw another card), the dealer takes hits until a total of 17 or more is reached. If the dealer busts, you win. If both you and the dealer stand, the one with the higher total below or equal to 21 wins. A tie is a push.

The computer deals, keeps score, and asks the following questions at appropriate times: `Do you want a hit? Insurance? Double? Double down?` To answer yes, press Y; 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 the Interrupt (Ctrl−C) or End Of File (Ctrl−D) key sequence; the computer displays the final action and score and exits.

Files

/usr/games Location of the system's games.

Related Information

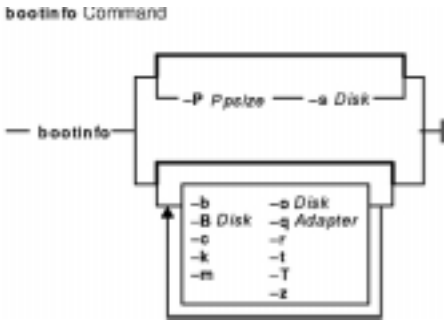
The **arithmetic** command, **back** command, **craps** command, **fish** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command, **turnon** command, **wump** command.

bootinfo Command

Purpose

Determines and displays various boot information, including boot device type and boot device name. This command is NOT a user−level command and is NOT supported in AIX Version 4.2 or later.

Syntax



bootinfo [−**P***Ppsize*−**s***Disk*] | [−**b**] [−**B** *Disk*] [−**c**] [−**k**] [−**m**] [−**q** *Adapter*] [−**r**] [−**t**] [−**T**] [−**o***Disk*] [−**z**]

Description

The **bootinfo** command is used during the boot and BOS install to gather and display information. During boot it is used to determine the boot device type and which device the machine has booted from. During a network boot, the **bootinfo** command displays the contents of the client's **bootpd** daemon REPLY packet. This information is used by the boot programs to contact the server to obtain the client's file systems.

The **bootinfo** command uses the device configuration databases in information searches. For some information, the **bootinfo** command uses NVRAM (nonvolatile random access memory).

When booting a system across a network, use the **−c** flag to display the following information:

- Client IP address
- Server IP address
- Gateway IP address
- Network type of boot, where:

  68 (ASCII "D") Ethernet

  79 (ASCII "O") Token Ring

  80 (ASCII "P") FDDI

- Slot number of the network adapter
- 802.3 indicator
- Boot file
- Vendor tag information
- Hardware attribute value

  For token−ring:

4 Specifies 4 megabit−per−second token ring

5 Specifies 16 megabit−per−second Token Ring

For MCA Ethernet:

```
7 Specifies thin cable
8 Specifies thick cable
```

All of the items except the hardware attribute value are part of the **bootp** daemon reply packet information. The **bootinfo** command determines the hardware attribute value by looking in NVRAM.

Flags

−b Returns the last boot device.

−B *Disk* Displays a 1 if the IPL code in the ROS on the machine running the command is capable of booting from the specified disk. Otherwise, the command displays a 0.

−c Displays **bootp** daemon reply packet information stored with IPL control block.

−k Specifies the key position. The return value indicates:

1
Key is in Secure position.

2
Key is in Service position.

3
Key is in Normal position.

−m Displays the machine model code.

−o*Disk* Displays either the disk device name or the location depending upon the value of *disk*.

−P*Ppsize* Size of disk physical partitions to be used for calculating defaults.

−q*Adapter* Displays a 1 if the IPL code in the ROS on the machine running the command is capable of booting via the specified adapter. Otherwise, this command displays a 0.

−r Displays amount of real memory in kilobytes.

−s*Disk* Displays disk size in megabytes.

−t Specifies the type of boot. The return values include:

1
Disk boot

3
CD−ROM boot

4
Tape boot

5
Network boot

−T Displays the hardware platform type of the running machine. Machines with fundamental differences, such as different types of busses, may have different hardware platform types. For more information see the **bosboot** command.

−z Specifies whether the machine hardware is MP−capable (capable of running the multi−processor kernel and supporting more than one processor). The return value indicates:

0
The machine is not MP−capable.

1
The machine is MP−capable.

Examples

1. To determine default physical−partition size for disk hdisk2, enter:

   ```
   bootinfo −P 0 −s hdisk2
   ```

2. To display amount of real memory, enter:

   ```
   bootinfo −r
   ```

Related Information

The **bosboot** command

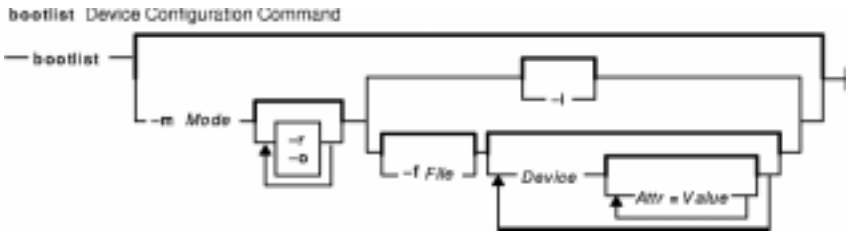
Understanding the Boot Process in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

bootlist Command

Purpose

Displays and alters the list of boot devices available to the system.

Syntax



bootlist [{ **−m***Mode* } [**−r**] [**−o**] [[**−i**] | [[**−f***File*] [*Device* [*Attr*=*Value* ...] ...]]]

Description

The **bootlist** command allows the user to display and alter the list of possible boot devices from which the system may be booted. When the system is booted, it will scan the devices in the list and attempt to boot from the first device it finds containing a boot image. This command supports the updating of the following:

- Normal boot list. The normal list designates possible boot devices for when the system is booted in normal mode.
- Service boot list. The service list designates possible boot devices for when the system is booted in service mode. How a system is booted in service mode is hardware−platform dependent. It may require a key switch to be turned to the Service position, a particular function key to be pressed during the boot process, or some other mechanism, as defined for the particular hardware platform.
- Previous boot device entry. This entry designates the last device from which the system booted. Some hardware platforms may attempt to boot from the previous boot device before looking for a boot device in one of the other lists.

Support of these boot lists may vary from platform to platform. A boot list can be displayed or altered only if the platform supports the specified boot list. It may even be the case that a particular hardware platform does not support any of the boot lists.

When searching for a boot device, the system selects the first device in the list and determines if it is bootable. If no boot file system is detected on the first device, the system moves on to the next device in the list. As a result, the ordering of devices in the device list is extremely important.

The **bootlist** command supports the specification of generic device types as well as specific devices for boot candidates. Possible device names are listed either on the command line or in a file. Devices in the boot device list occur in the same order as devices listed on the invocation of this command.

The devices to be entered into the boot list may be specified in a file. This makes an alterable record of the boot devices available for reference or future update. When the **−f** flag is used, the list of devices is taken from the file specified by the *file* variable. Devices from this list are then placed in the boot list in the order found in the file.

Attention: Care must be taken in specifying the possible boot devices. A future reboot may

fail if the devices specified in the device list become unbootable. The system must not be powered off or reset during the operation of the **bootlist** command. If the system is reset, or if power fails at a critical point in the execution of this command, the boot list may be corrupted or lost.

The selection of the boot list to display or alter is made with the **−m***mode* option, where the *mode* variable is one of the keywords: **service**, **normal**, **both**, or **prevboot**. If the **both** keyword is specified, then both the normal boot list and the service boot list will be displayed, or if being altered, will be set to the same list of devices. If the **prevboot** keyword is specified, the only alteration allowed is with the **−i** (invalidate) flag . The **−i** flag invalidates the boot list specified by the **−m** flag.

In versions 4.2 and later, the devices currently in the boot list may be displayed by using the **−o** flag. The list of devices that make up the specified boot list will be displayed, one device per line. If a device specified in the boot list is no longer present on the system, a `−' is displayed instead of a name. The output is in a form that can be captured in a file and used as input to the **bootlist** command with the **−f** flag. This may be useful for restoring a boot list after making a temporary change.

Device Choices

The device name specified on the command line (or in a file) can occur in one of two different forms:

- It can indicate a specific device by its device logical name.
- It can indicate a generic or special device type by keyword. The following generic device keywords are supported:

  fd Any standard I/O−attached diskette drive

  scdisk Any SCSI−attached disk (including serial−link disk drives)

  badisk Any direct bus−attached disk

  cd Any SCSI−attached CD−ROM

  rmt Any SCSI−attached tape device

  ent Any Ethernet adapter

  tok Any Token−Ring adapter

  fddi Any Fiber Distributed Data Interface adapter

  Note: Some hardware platforms do not support generic device keywords. If a generic device keyword is specified on such a platform, the update to the boot list is rejected and this command fails.

  When a specific device is to be included in the device list, the device's logical name (used with system management commands) must be specified. This logical name is made up of a prefix and a suffix. The suffix is generally a number and designates the specific device. The specified device must be in the Available state. If it is not, the update to the device list is rejected and this command fails. The following devices and their associated logical names are supported (where the bold type is the prefix and the *xx* variable is the device−specific suffix):

  fd*xx* Diskette−drive device logical names

  hdisk*xx* Physical−volume device logical names

  cd*xx* SCSI CD−ROM device logical names

  rmt*xx* Magnetic−tape device logical names

  ent*xx* Ethernet−adapter logical names

  tok*xx* Token−ring adapter logical names

  fddi*xx* Fiber Distributed Data Interface adapter logical names

Attribute Choices

Attributes are extra pieces of information about a device that the user supplies on the command line. Since this information is specific to a particular device, generic devices do not have attributes. Attributes apply to the device that immediately precedes them on the command line, which allows attributes to be interspersed among devices on the command line. Currently, only network devices have attributes. These are:

bserver	IP address of the BOOTP server
gateway	IP address of the gateway
client	IP address of the client
hardware	Hardware address

These attributes can be combined in the following ways:

- The **hardware** attribute cannot be specified alone; it must be specified with the **bserver** or **gateway** attribute. When specified with **bserver** or **gateway**, it applies to the server or gateway, respectively; when both **bserver** and **gateway** are specified, **hardware** will apply to **gateway**.
- The **bserver** attribute can be specified alone, with **hardware**, and/or **gateway**.
- If the **gateway** attribute is specified, **bserver** and **client** must also be specified.
- The **client** attribute can only be specified with **gateway** and **bserver**.

Some of these attributes may not be supported on some hardware platforms. Additional hardware platform restrictions may apply.

The syntax for specifying an attribute is *attr=value*, where *attr* is the attribute name, *value* is the value, and there are no spaces before or after the =.

File Format When Using the −f Flag

The file specified by the *file* variable should contain device names separated by white space:

```
hdisk0 hdisk1 cd1
```

or one device per line:

```
hdisk0
hdisk1
cd1
```

Error Handling

If this command returns with an error, the device lists are not altered. The following device list errors are possible:

- If the user attempts to display or alter a boot list that is not supported by the hardware platform, the command fails, indicating the mode is not supported.
- If the user attempts to add too many devices to the boot list, the command fails, indicating that too many devices were requested. The number of devices supported varies depending on the device selection and the hardware platform .
- If an invalid keyword, invalid flag, or unknown device is specified, the command fails with the appropriate error message.
- If a specified device is not in the Available state, the command fails with the appropriate error message.

Flags

Device	Provides the names of the specific or generic devices to include in the boot list.
−f*File*	Indicates that the device information is to be read from the specified file name.
−i	Indicates that the device list specified by the **−m** flag should be invalidated.
−m*Mode*	Specifies which boot list to display or alter. Possible values for the *mode* variable are **normal**, **service**, **both**, or **prevboot**.
−o	Indicates that the specified boot list is to be displayed after any specified alteration is performed. The output is a list of device names. This flag applies only to AIX Version 4.2 or later.
−r	Indicates that the specified boot list is to be displayed after any specified alteration is performed. The output is hardware−platform dependent. It may be a hexadecimal dump of the boot list or a list of device names. (This is normally used for problem determination.)

Security

Privilege Control: Only the root user and members of the security group should have execute (x) access to this command.

Auditing Events:

Event	Information
NVRAM_Config	File name

Examples

1. To invalidate the Service mode boot list, enter:

   ```
   bootlist −m service −i
   ```

2. To make a boot list for Normal mode with devices listed on the command line, enter:

   ```
   bootlist −m normal hdisk0 hdisk1 rmt0 fd
   ```

3. To make a boot list for Normal mode with a device list from a file, enter:

   ```
   bootlist −m normal −f /bootlist.norm
   ```

   where **bootlist.norm** is a file containing device names to be placed in the boot list for Normal mode. The device names in the **bootlist.norm** file must comply with the described format.

4. To invalidate the previous boot device entry, enter:

   ```
   bootlist −m prevboot
   ```

5. To boot from a Token−Ring device in slot 2, enter:

   ```
   bootlist −m normal tok0
   ```

6. To attempt to boot through a gateway using Ethernet, and then try other devices, enter:

   ```
   bootlist −m normal ent0 gateway=129.35.21.1 bserver=129.12.2.10
   \ client=129.35.9.23 hdisk0 rmt0 tok0 bserver=129.35.10.19
   hdisk1
   ```

Related Information

The **nvram** special file.

Device Configuration Subsystem Programming Introduction in *AIX Kernel Extensions and Device Support Programming Concepts*.

List of Device Configuration Commands in *AIX Kernel Extensions and Device Support Programming Concepts*.

Special Files Overview in *AIX Version 4.3 Files Reference*.

bootparamd Daemon

Purpose

Provides information for booting to diskless clients.

Syntax

/usr/sbin/rpc.bootparamd [**−d**]

Description

The **bootparamd** daemon is a server process that provides information necessary to diskless clients for booting. It consults either the **bootparams** database or the **/etc/bootparams** file if the NIS service is not running.

Flags

−d Displays debugging information.

Files

/etc/bootparams Contains the list of client entries that diskless clients use for booting.

Related Information

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

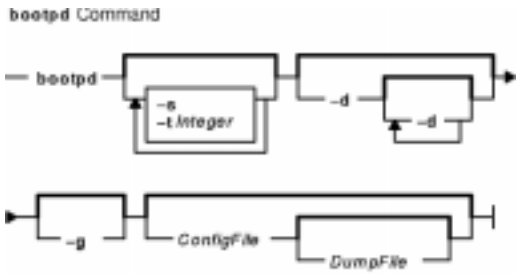
List of NFS Commands.

bootpd Daemon

Purpose

Sets up the Internet Boot Protocol server.

Syntax



bootpd [**−s**] [**−t** *Integer*] [**−d** [**−d** ...]] [**−g**] [C*onfigFile* [*DumpFile*]]

Description

The **bootpd** command implements an Internet Boot Protocol server.

The **bootpd** daemon is normally started by the **inetd** daemon. The default **/etc/inetd.conf** file contains the following line:

```
bootps dgram udp wait root /usr/sbin/bootpd bootpd
```

By default, this entry is commented out. One way to add the **bootpd** daemon to the **inetd** daemon's list of available subservers is to use the System Management Interface Tool (SMIT). Another way to make the **bootpd** daemon available is to edit the **/etc/inetd.conf** file, uncomment the `bootps` entry, and enter `refresh −sinetd` or `kill −1 InetdPid` to inform the **inetd** daemon of the changes to its configuration file. Now, when a bootp request arrives, **inetd** starts the **bootpd** daemon. Once the daemon is started, **bootpd** continues to listen for boot requests. However, if the server does not receive a boot request within 15 minutes of the previous one, it exits to conserve system resources. This time−out value of 15 minutes can be changed using the **−t** flag.

To start the **bootpd** daemon without **inetd**, use the `−s` flag. In this mode, the **bootpd** daemon continues to listen for bootp requests until the daemon is killed.

Upon startup, the **bootpd** daemon looks in the **/etc/services** file to find the port numbers to use, and extracts the following entries:

bootps The BOOTP server listening port.

bootpc The destination port used to reply to clients.

Then, the **bootpd** daemon reads its configuration file. If a configuration file is not specified, the default file is **/etc/bootptab**. Once the configuration file is read, the **bootpd** daemon begins listening for and processing bootp requests. The **bootpd** daemon rereads its configuration file when it receives a **SIGHUP** hang−up signal, or when it receives a bootp request packet and detects that the file has been updated. Hosts may be added, deleted, or modified when the configuration file is reread.

Flags

−d Increases the level of debugging output. This flag can be used many times. The following levels of debugging are available:

Debug Level	Syntax	Message
1	−d	Only error messages.
2	−d −d	Level 1 messages and messages indicating potential errors.
3	−d −d −d ...	Level 1 and level 2 and general information messages.

If the debug level is set to >0 and if the **syslogd** daemon is running, then all debug messages are printed in the **syslogd** log file.

−g Keeps the same gateway IP address that is in **bootp** request in **bootp** reply.

−s Runs the **bootpd** command in a stand−alone configuration. This mode is used for large network installations with many hosts. In this case, the **−t** flag has no effect since the **bootpd** command never exits.

−t Specifies a different time−out value in minutes, such as `−t20`. A time−out value of `0` means forever. The default time−out value is 15 minutes.

ConfigFile Specifies the configuration file. The default configuration file is **/etc/bootptab**.

DumpFile Specifies the file into which the **bootpd** daemon dumps a copy of the bootp server database. The default dump file is **/etc/bootpd.dump**.

Examples

1. To start the bootpd daemon in a stand−alone mode, enter the following:

   ```
   /usr/sbin/bootpd −s
   ```

2. To start the **bootpd** daemon in a stand−alone mode with a debug level of 3, with a configuration file of **/etc/newconfig**, and a dump file of **/etc/newdumpfile**, enter the following:

   ```
   /usr/sbin/bootpd −s −d −d −d /etc/newconfig /etc/newdumpfile
   ```

Files

/etc/bootpd.dump The default **bootpd** dumpfile

/etc/bootptab The default **bootpd** configuration file.

/etc/services Defines sockets and protocols used for Internet services.

/etc/inetd.conf Contains the configuration information for the **inetd** daemon.

Related Information

The **x_add_nfs_fpe** command, **x_add_trm_120** command, **x_add_trm_130** command, **x_add_trm_140** command, **x_add_trm_150** command, **x_add_trm_160** command, **x_add_xst_fpe** command, **x_chg_net**, **x_chg_trm_120** command, **x_chg_trm_130** command, **x_chg_trm_140** command, **x_chg_trm_150** command, **x_chg_trm_160** command, **x_def_net** command, **x_ls_net** command, **x_ls_net_fp** command, **x_ls_trm** command, **x_rm_fpe** command, **x_rm_net** command, and **x_rm_trm** command.

How to Start the System in *AIX Version 4.3 Quick Beginnings*.

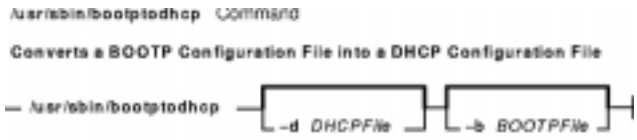
bootptodhcp Command

Purpose

To convert a BOOTP configuration file into a DHCP configuration file or to remove BOOTP configuration information for a particular host from the DHCP configuration file.

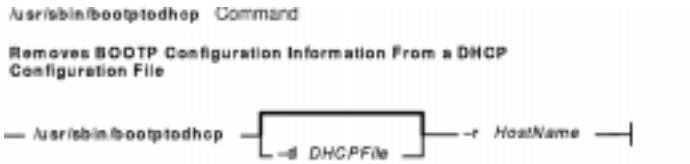
Syntax

To Convert a BOOTP Configuration File into a DHCP Configuration File



/usr/sbin/bootptodhcp [**−d***DHCPFile*] [**−b***BOOTPFile*]

To Remove a BOOTP Configuration Information From a DHCP Configuration File



/usr/sbin/bootptodhcp [**−d***DHCPFile*] **−r***HostName*]

Description

The **bootptodhcp** command has two functions. The first is to translate a BOOTP configuration file into a DHCP configuration. The default command with no arguments translates the **/etc/bootptab** file. The filenames may be changed by using the **−b** or **−d** flags to specify a different file names.

The second function of the **bootptodhcp** command is the removal of a BOOTP client's information from a DHCP configuration file. The **−r** flag specifies which client to remove from the file. If the **−d** flag is not used.

Flags

−b*BOOTPFile* Specifies the BOOTP configuration file. The default is **/etc/bootptab**.

−d*DHCPFile* Specifies the DHCP configuration file.

−r*HostName* Specifies the hostname of a BOOTP section to delete from the DHCP configuration file.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Security

Access Control: Any User

Files Accessed: Need appropriate access permissions for files

Files

/usr/sbin/bootptodhcp	Contains the **bootptodhcp** command.
/etc/bootptab	Contains the default configuration file for bootpd.

Related Information

The **dhcpsconf** command

DHCP Client Configuration File

DHCP Server Configuration File

bootp Configuration File

TCP/IP Address and Parameter Assignment − Dynamic Host Configuration Protocol (DHCP)

The SMIT Interface for TCP/IP in *AIX Version 4.3 System Management Guide: Communications and Networks*.

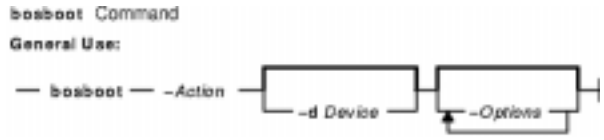
bosboot Command

Purpose

Creates boot image.

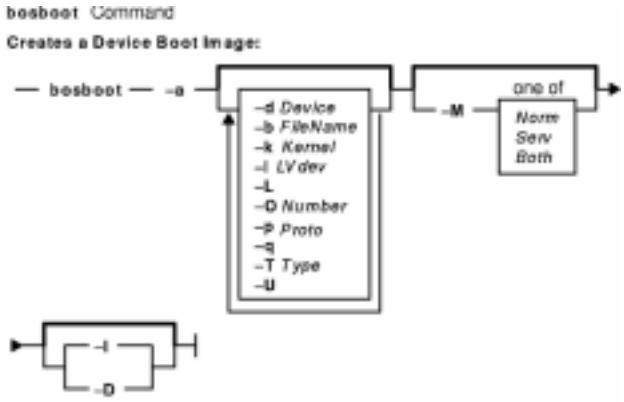
Syntax

For General Use:



bosboot −*Action* [−**d** *Device*] [−*Options ...*]

To Create a Device Boot Image:

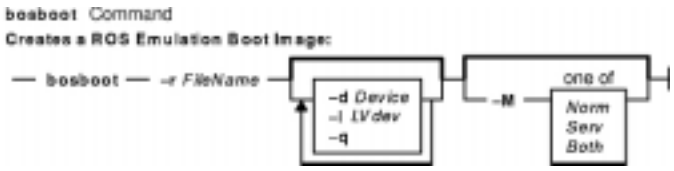


bosboot −**a** [−**d** *Device*] [−**p** *Proto*] [−**k** *Kernel*] [−**U**] [−**I** | −**D**] [−**l** *LVdev*] [−**L**] [−**M** { *Norm* | *Serv* | *Both* }] [−**O** *Number*][−**T***Type*][−**b** *FileName*] [−**q**]

To Copy a Device Boot Image:

bosboot−**w***FileName* [−**d** *Device*] [−**q**]

To Create a ROS Emulation Boot Image:



bosboot −**r***FileName* [−**d** *Device*] [−**l** *LVdev*] [−**M** { *Norm* | *Serv* | *Both* }] [−**q**]

Description

The **bosboot** command creates the boot image that interfaces with the machine boot ROS (Read−Only Storage) EPROM (Erasable Programmable Read−Only Memory).

The **bosboot** command creates a boot file (boot image) from a RAM (Random Access Memory) disk file system and a kernel. This boot image is transferred to a particular media that the ROS boot code recognizes. When the machine is powered on or rebooted, the ROS boot code loads the boot image from the media into memory. ROS then transfers control to the loaded image's kernel.

The associated RAM disk file system contains device configuration routines that make the machine's devices and file systems available. The RAM disk file system contains differing configuration files depending upon the boot device. A **mkfs** prototype file is supplied for each type of device. (See note 6 below.) Currently supported devices are:

- CD−ROM
- Disk
- Tape
- Network

A network device may be a token ring, Ethernet, or Fiber−Distributed Data Interface (FDDI) used to boot from a network boot server over a local area network (LAN).

The boot image varies for each type of device booted and is compressed by default to fit on certain media and to lessen real memory requirements. The boot image can be left uncompressed by specifying the **−U** flag. The logical volume must be large enough for the uncompressed boot image.

In addition to creating a boot image, the **bosboot** command always saves device configuration data for disk. It does not update the list of boot devices in the NVRAM (nonvolatile random access memory). You can modify the list with the **bootlist** command.

The **bosboot** command is normally called during the Base Operating System installation and by the **updatep** command when the operating system is upgraded.

Notes:

1. You must have root user authority to use the **bosboot** command.
2. Do not reboot the machine if the **bosboot** command is unsuccessful with a message not to do so while creating a boot disk. The problem should be resolved and the **bosboot** command run to successful completion.
3. The **bosboot** command requires some space in the **/tmp** file system and the file system where the target image is to reside, if there is such an image.
4. The **bosboot** command requires that the specified physical disk contain the boot logical volume. To determine which disk device to specify, issue the following command:

   ```
   lsvg    -M    rootvg
   ```

   This command displays a map of all logical volumes. The default boot logical volume is **hd5**. Use the disk device that contains the boot logical volume.

5. When the device is not specified with the **−d** flag, the **bosboot** command assumes the default device is the disk the system is booted from. However, if the prototype file is specified with a **−p** flag, the device must also be specified with a **−d** flag.
6. The prototype file used by the **bosboot** command to build the RAM disk file system depends on the boot device and the hardware platform (**sys0**) type of the machine the

boot image will run on.

The hardware platform type is an abstraction which allows machines to be grouped according to fundamental configuration characteristics such as number of processors and/or I/O bus structure. Machines with different hardware platform types will have basic differences in the way their devices are dynamically configured at boot time. The hardware platform type **rs6k** applies to all Micro Channel−based uni−processor models. The type **rs6ksmp** applies to all Micro Channel−based symmetric multi−processor models. The type **rspc** applies to all ISA−bus models. As new models are developed, their hardware platform types will either be one of the aforementioned types or, if fundamental configuration differences exist, new types will be defined. Boot images for a given boot device type will generally be different for machines with different hardware platform types.

The prototype file used by **bosboot** is constructed by starting with a copy of the base prototype file for the platform type and boot device (e.g. **/usr/lib/boot/rs6k.disk.proto**). Next the **bosboot** command looks at the **pcfg** file for the platform type being used (e.g **/usr/lib/boot/rs6k.pcfg**). The **pcfg** file contains entries which **bosboot** uses in a template to search for proto extension files. These files, located in the directory **/usr/lib/boot/protoext**, provide extensions to the prototype file under construction. As an example, if the platform type is **rs6k** and the boot device is disk, and the file **/usr/lib/boot/protoext/rs6k.pcfg** contains the following:

mca.

scsi.

.

.

.

The **bosboot** command will start with the base prototype file **/usr/lib/boot/rs6k.disk.proto** and search the directory **/usr/lib/boot/protoext** for any files that match the template **disk.proto.ext.mca.***. The contents of these files are added to the prototype file under construction. Next, the contents of files matching the template **/usr/lib/boot/protoext/disk.proto.ext.scsi.*** are added to the prototype file under construction. This continues until all lines in the **pcfg** file have been processed. At this point the prototype file under construction is complete. The **bosboot** command passes this prototype file to the **mkfs** command which builds the RAM disk file system.

7. In AIX Version 3.2.5, the prototype files used by the BOSBOOT command to build boot images were dependent on the boot device. This is still true in AIX Version 4.1.3, but in addition, the prototype files are dependent on the system device type (sys0) of the machine for which the boot image is built.

   This is reflected in the names of the prototype files:

   /usr/lib/boot/rs6k.disk.proto

   /usr/lib/boot/rs6ksmp.disk.proto

   /usr/lib/boot/rs6k.tape.proto

/usr/lib/boot/rs6ksmp.tape.proto

/usr/lib/boot/rs6k.cd.proto

/usr/lib/boot/rs6ksmp.cd.proto

/usr/lib/boot/network/rs6k.tok.proto

/usr/lib/boot/network/rs6ksmp.tok.proto

/usr/lib/boot/network/rs6k.ent.proto

/usr/lib/boot/network/rs6ksmp.ent.proto

/usr/lib/boot/network/rs6k.fddi.proto

/usr/lib/boot/rspc.disk.proto

/usr/lib/boot/rspc.cd.proto

/usr/lib/boot/network/rspc.tok.proto

/usr/lib/boot/network/rspc.ent.proto

The system device type is an abstraction that allows machines to be grouped according to fundamental configuration characteristics, such as number of processors and I/O bus structure. The system device is the highest−level device in the system node, which consists of all physical devices in the system.

Machines with different system device types have basic differences in the way their devices are dynamically configured at boot time. The system device type RS6K applies to all of the RS/6000(*) models supported by AIX versions prior to 4.1. The type RS6KSMP applies to symmetric multiprocessor models.

The **bosboot** command, by default, uses the prototype file that matches the system device type of the machine executing the command. The −T option allows you to specify the system device type of the prototype file. To determine the system device type of a machine, enter:

```
bootinfo −T
```

Note: The **bootinfo** command does not apply to AIX Version 4.2 or later.

Flags

−d*device* Specifies the boot device. This flag is optional for hard disk.

The following flags are action flags. One and only one flag must be specified.

−a Creates complete boot image and device.

−r*FileName* Creates a ROS emulation boot image. A ROS emulation boot image is created from a file containing the ROS emulation code. This image is used to provide network boot capability to machines whose ROS does not provide it. A ROS emulation boot image is transferred to a diskette, tape, or boot logical volume. ROS will load the image from the media into memory, whereupon ROS will transfer control to the ROS emulation code. This code emulates the later

version ROS code that supports network boot.

−w*FileName* Copies the specified device boot image to device.

The following flags are option flags:

−b*FileName*	Uses specified file name as the boot image name. This flag is optional.
−D	Loads the low level debugger. This flag is optional.
−I (upper case i)	Loads and invokes the low−level debugger. This flag is optional.
−k	Allows to specify an alternate kernel file. If this flag is not specified, /**unix** is the default.
−k*Kernel*	Uses the specified kernel file for the boot image. This flag is optional.
−L	Enables lock instrumentation for MP systems. This flag has no effect on systems that are not using the MP kernel.
−l (lower case L) *LVDev*	Uses target boot logical volume for boot image. This flag is optional.
−M*Norm*|*Serv*|*Both*	Specifies the boot mode. The options are: **norm** Indicates normal mode. **serv** Indicates service mode. **both** Indicates both modes.
−O*Number*	Uses specified number as the offset (in 512−byte blocks) for CD−ROM boot image from the beginning of the CD−ROM device. This flag is optional.
−p*Proto*	Uses the specified prototype file for the RAM disk file system. This flag is optional.
−q	Determines how much disk space is required in which file system to create the boot image. Boot image is not created. This flag is optional.
−T *Type*	Specifies the hardware platform type (see note 6). This causes the **bosboot** command to create a boot image for the hardware platform type specified. If the type is not specified, the **bosboot** command creates a boot image whose hardware platform type matches that of the currently running machine. This flag is optional.
−U	Creates an uncompressed boot image. This flag is optional.

Security

Access Control: Only the root user can read and execute this command.

Examples

1. To create a boot image on the default boot logical volume on the fixed disk from which the system is booted, enter:

   ```
   bosboot −a
   ```

2. To create a bootable image called **/tmp/tape.bootimage** for a tape device, enter:

   ```
   bosboot −ad /dev/rmt0 −b /tmp/tape.bootimage
   ```

3. To copy a given tape boot image to a tape device, enter:

   ```
   bosboot −w /tmp/tape.bootimage −d rmt0
   ```

4. To create a boot image file for an Ethernet boot, enter:

   ```
   bosboot −ad /dev/ent0 −M both
   ```

5. To create an uncompressed boot image for a hard disk **/dev/hdisk1**, enter:

   ```
   bosboot −ad /dev/hdisk1 −U
   ```

6. To create a token ring boot image for a machine whose hardware platform type is **rspc** while you are running on a machine whose hardware platform type is **rs6k**, enter:

   ```
   bosboot −ad /dev/tok −T rspc
   ```

Files

/usr/sbin/mkboot Specifies boot creation routine.

/usr/lib/boot/rs6k.disk.proto

/usr/lib/boot/rs6ksmp.disk.proto

/usr/lib/boot/rspc.disk.proto Specifies disk RAM file system template.

/usr/lib/boot/rs6k.tape.proto

/usr/lib/boot/rs6ksmp.tape.proto Specifies tape RAM file system template.

/usr/lib/boot/rs6k.cd.proto

/usr/lib/boot/rs6ksmp.cd.proto

/usr/lib/boot/rspc.cd.proto Specifies CD−ROM RAM file system template.

/usr/lib/boot/network/rs6k.tok.proto

/usr/lib/boot/network/rs6ksmp.tok.proto

/usr/lib/boot/network/rspc.tok.proto Specifies token−ring RAM file system template

/usr/lib/boot/network/rs6k.ent.proto

/usr/lib/boot/network/rs6ksmp.ent.proto

/usr/lib/boot/network/rspc.ent.proto Specifies Ethernet RAM file system template.
/usr/lib/boot/network/rs6k.fddi.proto Specifies FDDI RAM file system template.

Related Information

The **bootlist** command, **mkboot** command, and **lockstat** command.

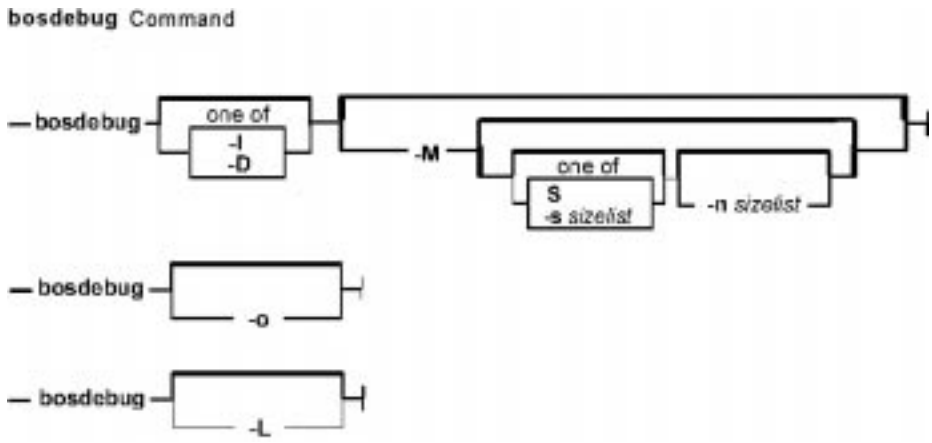
Understanding the Boot Process in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

bosdebug Command

Purpose

Enables, disables and/or displays the status information of the system

Syntax



bosdebug [**−I** | **−D**] [**−M** [**−S** | **−s***sizelist*] [**−n***sizelist*]]

bosdebug [**−o**]

bosdebug [**−L**]

Description

The **bosdebug** command enables, disables, and/or displays the status of debugging features of the system.

−D Causes the kernel debug program to be loaded on each subsequent reboot.

−I Causes the kernel debug program to be loaded and invoked on each subsequent reboot.

−L Displays the current settings for the kernel debug program and the memory overlay detection system. Note that the settings shown will not take effect until after the next time that the **bosboot −a** and **shutdown −r** commands are run. This is the default.

−M Causes the memory overlay detection system to be enabled. Memory overlays in kernel extensions and device drivers will cause a system crash.

−n*sizelist* Has the same effect as the **−s** option, but works instead for network memory. Each size must be in the range from 32 to 2048, and must be a power of 2. This causes the net_malloc_frag_mask variable of the **no** command to be turned on during boot.

−o Turns off all debugging features of the system.

−s*sizelist* Causes the memory overlay detection system to promote each of the specified allocation sizes to a full page, and allocate and hide the next subsequent page after each allocation. This causes references beyond the end of the allocated memory to cause a system crash. sizelist is a list of memory sizes separated by commas. Each size must be in the range from 16 to 2048, and must be a power of 2.

−S Causes the memory overlay detection system to promote all allocation sizes to the next higher multiple of page size (4096), but does not hide subsequent pages. This improves the chances that references to freed memory will result in a crash, but it does not detect reads or writes beyond the

end of allocated memory until that memory is freed.

Any changes made by this command will not take effect until the **bosboot −a** and **shutdown −r** commands have been run.

Related Infomation

The **bosboot** command and **shutdown** command.

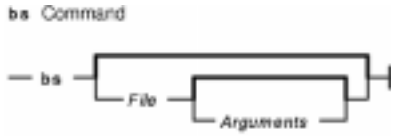
Memory Overlay Detection System (MODS) in the *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*.

bs Command

Purpose

Compiles and interprets modest−sized programs.

Syntax



bs [*File* [*Arguments*]]

Description

The **bs** command is a compiler and interpreter for 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 parameter *File* specifies a file of program statements that you create and that the compiler reads before it reads from standard input. Statements entered from standard input are normally executed immediately (see **compile** and **execute** statement syntax). By default, statements read from *File* are compiled for later execution.

Unless the final operator is assignment to a variable, the result of an immediate expression statement is displayed.

Additional command line *Arguments* can be passed to the program using the built−in functions **arg** and **narg**.

Program lines must conform to one of the following formats:

```
statement
label statement
```

The 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.

> **Note:** To avoid unpredictable results when using a range expression in the international environment, use a character class expression rather than a standard range expression.

Statement Syntax

break	Exits the innermost **for** or **while** loop.
clear	Clears the symbol table and removes compiled statements from memory. A **clear** is always executed immediately.
compile [*Expression*]	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. **compile** 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, **dump** displays the number of the last statement and (possibly) the user−function trace.

exit [*Expression*] Returns to the system level. The *Expression* is returned as process status.

execute Changes to immediate execution mode (pressing the INTERRUPT key has the same effect). This statement does not cause stored statements to execute (see **run**).

for Performs repeatedly, under the control of a named variable, a statement or a group of statements using one of the following syntaxes:

```
for name=Expression Expression statement
next
```

OR

```
for name=Expression Expression
statement . . .
next
```

OR

```
for Expression, Expression, Expression statement
next
```

OR

```
for Expression, Expression, Expression
statement . . .
next
```

The first format specifes a single statement where the variable takes on the value of the first expression and then is increased by one on each loop until it exceeds the value of the second expression. You can use the second format to do the same thing , but you can specify a group of statements.

The third format requires an initialization expression followed by a test expression (such as true to continue) and a loop−continuation action expression. You can use the fourth format to do the same thing, but you can specify a group of statements. Use commas to separate the expressions in the third and fourth formats.

fun Defines a user−written function using the following syntax:

```
fun f ([a, . . . ]) [v, . . . ]
statement . . .
nuf
```

f specifies the function name, *a* specifies any parameters, and *v* identifies any local variables for the user−written function. You can specify up to 10 parameters and local variables; however, they cannot be arrays or

associated with I/O functions. You cannot nest function definitions.

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

goto *Name* Passes control to the compiled statement with the matching label of *Name*.

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 entered as alphabetic characters a−f**.** A leading digit is required when a hexadecimal number begins with an alphabetic character (for example, f0a must be entered as 0f0a). **ibase** is always executed immediately.

if Performs a statement in one of the following syntaxes:

```
if Expression statement
[else
statement . . . ]
fi
```

OR

```
if Expression
statement . . .
 [else
statement . . . ]
fi
```

The first format specifies a single statement and the second format specifies a group of statements to continue using if the expression evaluates to nonzero. The strings 0 and "" (null) evaluate as zero.

In the second format, an optional **else** 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** is an **if**. You can put **fis** only on the same line as another **fi**. You can combine **else** and **if** into **elif**. You can close an **if** . . . **elif** . . . [**else** . . .] sequence with a single **fi**.

include *Expression* Evaluates an *Expression* to the name of a file containing program statements. Such statements become part of the program being compiled. The **include** statements are always executed immediately. Do not nest **include** statements.

obase *n* Sets the output base to *n*. The only supported values for *n* are 8, 10 (the default), and 16**.** Hexadecimal values 10 through15 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 **ibase**, **obase** is always executed immediately.

onintr Provides program control of interrupts using one of the following syntaxes:

```
onintr Label
```

OR

```
onintr
```

In the first format, control passes to the *Label* given, just as if a **goto** had

	been performed when **onintr** was executed. The effect of the **onintr** statement is cleared after each interrupt. In the second format, pressing INTERRUPT ends the **bs** program.
return [*Expression*]	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; **run** is always executed immediately.
stop	Stops execution of compiled statements and returns to immediate mode.
trace [*Expression*]	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 will be written. Each return decreases by one the **trace** expression value.

while	Performs repeatedly, under the control of a named variable, a statement or a group of statements using one of the following syntaxes:
	`while Expression statement`
	`next`
	OR
	`while Expression`
	`statement . . .`
	`next`
	The **while** statement is similar to the **for** statement except that only the conditional expression for loop continuation is given.

!*cmd*	Runs a command and then returns control to the **bs** program.
# *Comment*	Inserts a comment line.

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**).
*Name***(**[*Expression*[, *Expression*] . . .]**)**	Calls function *Name* and passes to it the parameters in parentheses. Except for built−in functions, *Name* must be defined in a **fun** statement. Function parameters are passed by value.
*Name***[***Expression*[, *Expression*] . . . **]**	References either arrays or tables (see built−in function table). 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 32,767.
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	Represents a character string delimited by " " (double quotation marks). Within the string, you can use the \ (backslash) as 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.
(*Expression*)	Alters the normal order of evaluation.
(*Expression***,** *Expression*[, *Expression*] . . . **) [***Expression***]**	Specifies to use the bracketed expression outside the parentheses as a subscript to the list of expressions within the parentheses. List elements are numbered from the left, starting at zero. The following expression has the value of True if the comparison is true: `(False, True) [a == b]`
Expression Operator Expression	Converts the operands to numeric form before the operator is applied unless the operator is an assignment, concatenation, or relational operator.

Unary Operators

? *Expression*	Tests for the success of *Expression* rather than its value. This interrogation operator is useful for testing: • The end of file • Result of the **eval** built−in function • Return from user−written functions (see **freturn**) 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.
− *Expression*	Negates *Expression*.
++ *Name*	Increases by one the value of the variable (or array reference).
−− *Name*	Decreases by one the value of the variable.
! *Expression*	Specifies the logical negation of *Expression*.

Note: Unary operators treat a null string as a zero.

Binary Operators (in increasing precedence)

=	Specifies the assignment operator. The left operand must be a name or array element. It acquires the value of the right operand. Assignment binds right to left; all other operators bind left to right.
_	Specifies the concatenation operator. (It is the underline character).

& | Specifies logical AND, logical OR.

The result of:

Expression & Expression

is 1 (true) only if both of its parameters are non−zero (true); it is 0 (false) if one or both of its parameters are 0 (false).

The result of:

Expression | *Expression*

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

< <= > >= == != Specifies the relational operators:

- < for less than
- <= for less than or equal to
- > for greater than
- >= for greater than or equal to
- == for equal to
- != for not equal to

The relational operators return 1 if the specified relation is True; otherwise they return 0 (false). 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 variable **LC_COLLATE**. The National Language Support Overview contains more information on this environment variable.

+ − Specifies addition and subtraction.

*** / %** Specifies multiplication, division, and remainder.

^ Specifies exponentiation.

Note: Binary operators treat a null string as a zero.

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.

Mathematical Functions

abs(*x*) Returns the absolute value of *x*.

atan(*x*) Returns the arc tangent 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*.

String Functions

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

bsize(*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. 0 return if no match is found. For 2−byte extended characters, the location of the first byte is returned.

trans(*s*, *f*, *t*) Translates characters in the source string *s* which match characters in *f* into 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.

substr(*s*, *Start*, *Length*) Returns the substring of *s* defined by *Start* position in characters and *Length* in characters.

match(*String*, *Pattern*)
mstring(*n*) Returns the number of characters in *string* that match *pattern*. The characters **.**, *****, **$**, **[**, **]**, ^ (when inside square brackets), \ (and \) have the following special meanings:

Note: See **ed** 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 any remaining characters in a 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 it (after an initial ^ , if any).

\(. . . \)
Marks a substring and matches it exactly.The pattern must match from the beginning of the string and the longest possible string. Consider, for example:

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

In this instance, .* matches `a 123a` (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 the "International Character Support Overview" for more information on collating sequences and equivalence classes.

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

File−Handling Functions

open*(Name, File, Mode)*

close*(Name*) Specifies the name, file type and file mode. *Name* must be a legal variable name (passed as a string). After a close, the name becomes an ordinary variable. For open, the *File* can be one of the following:

- 0 for standard input
- 1 for standard output
- 2 for error output
- A string representing a file name
- A string beginning with an !, which represents a command to be run (using "sh −c")

Mode must be specified with an r for read, w for write, W for write without the new line character, or a for append. The initial associations are:

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

access*(p, m)* Performs the access subroutine. 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 system request is permitted, −1 if it is denied.

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.

Table Functions

table*(Name, Size)* Specifies an associatively accessed, one−dimensional array. "Subscripts" (called keys) are strings (numbers are converted). *Name* must be a **bs** variable name (passed as a string). *Size* sets the minimum number of elements to be allocated. On table overflow, **bs** writes an error message.

item*(Name, i)*

key( ) Accesses table elements sequentially instead of in an orderly progression of key values. Where the item function accesses values, the key function accesses the "subscript" of the previous item call. Do not quote *Name*.

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 "parity", 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 one for true, zero for false.

Miscellaneous Functions

eval*(string)* Specifies to evaluate the string parameter as an expression. The function is handy for converting numeric strings to numbers. **eval** can also be used as a crude form of indirection, as in:

```
name = "x,y,z"
eval("++"_name)
```

which increments the variable `"x,y,z"`. In addition, when **eval** is preceded by ? (interrogation operator), you can control **bs** error conditions. For example:

```
?eval ("open(\"X\",\"XXX\", \"r\")")
```

returns the value zero 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)* 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 **tplot**.

The call requests are as follows:

***plot(0**, term)*
: Causes further plot output to be piped into **tplot** with a flag of **−T***term.*

plot(1)
: Erases the plotter.

***plot(2**, string)*
: Labels the current point with *string*

***plot(3**, x1, y1, x2, y2)*
: Draws the 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 end points *(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.
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 ( ) Returns, in immediate mode, the most recently computed value.

Example

To execute the bs command and direct the result to a file called output, enter:

```
bs < input.n > output
```

OR

```
bs input.n > output
```

Related Information

The **ksh** command.

The **access** subroutine, **printf** subroutine.

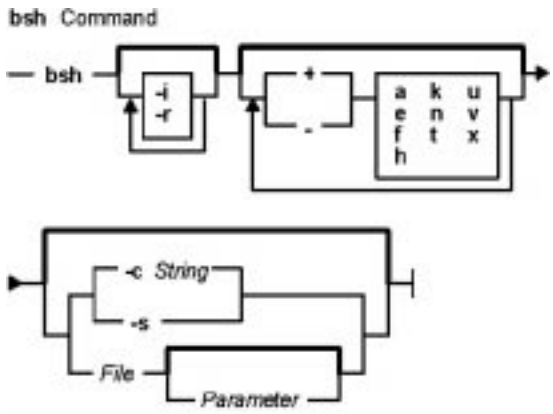
The National Language Support Overview for Programming in *AIX General Programming Concepts: Writing and Debugging Programs*.

bsh Command

Purpose

The **bsh** command invokes the Bourne shell.

Syntax



bsh [**−i**] [**−r**] [{ **+** | **−** } { [**a**] [**e**] [**f**] [**h**] [**k**] [**n**] [**t**] [**u**] [**v**] [**x**] }] [**−c** *String* | **−s** | *File* [*Parameter*]]

> **Note:** Preceding a flag with a **+** (plus sign) rather than a **−** (minus sign) turns it off.

Description

The **bsh** command invokes the Bourne shell, an interactive command interpreter and command−programming language. The shell carries out commands either interactively from a terminal keyboard or from a file.

For more information about the Bourne shell, see "Bourne Shell" in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

Flags

The Bourne shell interprets the following flags only when the shell is invoked at the command line.

> **Note:** Unless you specify either the **−c** or **−s** flag, the shell assumes that the next parameter is a command file (shell script). It passes anything else on the command line to that command file. See the discussion of positional parameters in "Variable Substitution in the Bourne Shell" in *AIX Version 4.3 System User's Guide: Operating System and Devices* for more information.

−a Marks for export all variables to which an assignment is performed. If the assignment precedes a command name, the export attribute is effective only for that command's execution environment, except when the assignment precedes one of the special built−in commands. In this case, the export attribute persists after the built−in command has completed. If the assignment does not precede a command name, or if the assignment is a result of the operation of the **getopts** or **read** command, the export attribute persists until the variable is unset.

−c *String* Runs commands read from the *String* variable. Sets the value of special parameter `0` from the value of the *String* variable and the positional parameters (`$1`, `$2`, and so on) in sequence from the remaining *Parameter* operands. The shell does not read additional commands from standard input when you specify this flag.

−e Exits immediately if all of the following conditions exist for a command:
- It exits with a return value greater than 0.
- It is not part of the compound list of a **while**, **until**, or **if** command.
- It is not being tested using AND or OR lists.
- It is not a pipeline preceded by the ! (exclamation point) reserved word.

−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.)

−i Makes the shell interactive, even if input and output are not from a workstation. In this case the shell ignores the **TERMINATE** signal, so that the **kill 0** command does not stop an interactive shell, and traps an **INTERRUPT** signal, so you can interrupt the function of the **wait** command. In all cases, the shell ignores the **QUIT** signal.

−k Places all keyword parameters in the environment for a command, not just those preceding the command name.

−n Reads commands but does not execute them. The **−n** flag can be used to check for shell−script syntax errors. An interactive shell may ignore this option.

−r Invokes the restricted shell. Using this flag is the same as issuing the **Rsh** command, except the shell enforces restrictions when reading the **.profile** files.

−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.

−t Exits after reading and executing one command.

−u Treats an unset variable as an error and immediately exits when performing variable substitution. An interactive shell does not exit.

−v Displays shell input lines as they are read.

−x Displays commands and their arguments before they are executed.

Note: Using a + (plus sign) rather than a − (minus sign) unsets flags. The `$−` special variable contains the current set of flags.

Files

/usr/bin/sh Specifies a default path name to the shell. This file is a link to the Bourne shell.

/usr/bin/bsh Specifies the path name to the Bourne shell.

/usr/bin/Rsh Specifies the path name to the restricted Bourne shell, a subset of the Bourne shell.

/tmp/sh* Contains temporary files that are created when a shell is opened.

Related Information

The **env** command, **sh** command, **Rsh** command.

The **/etc/passwd** file, **null** special file, **environment** file.

The **profile** file format.

Bourne Shell in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

Bourne Shell Special Commands in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

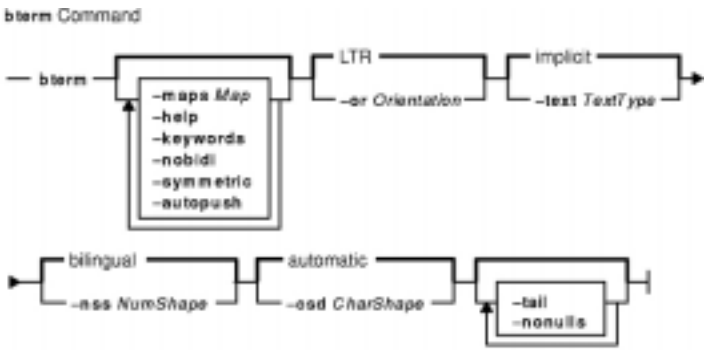
Variable Substitution in the Bourne Shell in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

bterm command

Purpose

Emulates terminals in bidirectional (BIDI) mode.

Syntax



bterm [**−maps***Map*] [**−help**] [**−keywords**] [**−nobidi**] [**−symmetric**] [**−autopush**] [**−or***Orientation*] [**−text***TextType*] [**−nss***NumShape*] [**−csd***CharShape*] [**−tail**] [**−nonulls**]

Description

The **bterm** command emulates the IBM 3151, VT220, HFT and other terminals. It operates in BIDI mode on ASCII terminals. This command creates a BIDI shell that can run any BIDI application. You cannot initiate the **bterm** command recursively from within itself.

The maps that determine the keyboard mapping and the symmetric swapping of characters are specified by the **−maps** flag. You can specify other BIDI behaviors using the flags available to the **bterm** command or by setting them in the defaults files. Such behaviors include the default text mode, the default screen orientation, the default mode of Arabic character shaping, the default shape of numerals, whether the Symmetric Swapping mode is enabled and whether the Autopush mode is enabled or not. The behaviors specified with flags take precedence over the behaviors set in the defaults files.

The default files are searched in the following order:

1. The **.Bidi−defaults** file is searched for in your home directory.
2. If the file is not found, the **bterm** command searches for the **BTerm** resource file in the **/usr/lib/nls/bidi/$LANG/app−defaults** file.

Flags

−autopush
Enables the Autopush mode in visual text mode.

−csd *CharShape*
Specifies the shape of Arabic characters. The *CharShape* variable can be one of the following options:

- **automatic**
- **isolated** (visual text mode only)
- **initial** (visual text mode only)
- **middle** (visual text mode only)
- **final** (visual text mode only)

- **passthru**

The default is automatic shaping.

−help Lists the available parameters and their syntax.

−keywords Lists the keywords available in defaults file.

−maps *Map* Specifies the map used for keyboard mapping and symmetric swapping of characters. Each language has a different map, and the available options for the *Map* variable are in the **/usr/lib/nls/bidi/maps** directory. You must specify the environment variable **BIDIPATH** as follows:

```
export BIDIPATH=/usr/lib/nls/bidi
```

−nobidi Disables the BIDI mode.

−nonulls Initializes the screen with spaces instead of nulls.

−nss *NumShape* Specifies the shape of the numerals. Specify one of the following options for the *NumShape* variable:

- **bilingual**
- **hindi**
- **arabic**
- **passthru**

The default is **bilingual**.

−or *Orientation* Specifies screen orientation. The *Orientation* variable can be either **LTR** or **RTL**. The default is **LTR**.

−symmetric Enables the Symmetric Swapping mode.

−tail Writes the "seen," "sheen," "sad," and "dad" characters of the Arabic language in two cells instead of one cell.

−text *TextType* Specifies the type of data stream. The *TextType* variable can be either **implicit** or **visual**. The default is **implicit**.

Key Combinations

To change the BIDI settings using key combinations, press the Ctrl+X key sequence to enter a BIDI command mode. Any key you type after this key sequence is interpreted as a BIDI command. Invalid keys sound a beep and exit the BIDI command mode. The following keys are valid BIDI commands:

Key	Purpose
r	Reverses the screen orientation.
n	Sets the language keyboard layer to National.
l	Sets the language keyboard layer to Latin.
a	Toggles the automatic shaping variable option of the Arabic characters (valid also for Implicit mode).
t	Displays the status.
space	Enters a required space (RSP).

For implicit mode only :

Key	Purpose
c	Toggles the column heading mode.

For visual mode only :

Key	Purpose
s	Initiates the Push mode.
e	Terminates the End Push mode.
p	Toggles the Autopush mode.
f	Shapes Arabic characters in their final forms.
i	Shapes Arabic characters in their initial forms.
b	Shapes Arabic characters in the Passthru mode.
o	Shapes Arabic characters in their isolated forms.
m	Shapes Arabic characters in their middle forms.

.Bidi−defaults Keywords

Use the following keywords to set the defaults for the **bterm** command.

.Bidi−defaults Keywords		
Keywords	**Value**	**Effect**
fScrRev	on	Screen reverse function key is enabled.
	off	Screen reverse function key is disabled.
fRTL	on	RTL keyboard layer function key is enabled.
	off	RTL keyboard layer function key is disabled.
fLTR	on	LTR keyboard layer function key is enabled.
	off	LTR keyboard layer function key is disabled.
fPush	on	Push function key is enabled.
	off	Push function key is disabled.
fEndPush	on	End Push function key is enabled.
	off	End Push function key is disabled.
fAutoPush	on	AutoPush function key is enabled.
	off	AutoPush function key is disabled.
fASD	on	Automatic Shape Determination function key is enabled.
	off	Automatic Shape Determination function key is disabled.
fShapeIS	on	Isolated Shape function key is enabled.
	off	Isolated Shape function key is disabled.
fShapeIN	on	Initial Shape function key is enabled.
	off	Initial Shape function key is disabled.
fShapeM	on	Middle Shape function key is enabled.
	off	Middle Shape function key is disabled.
fShapeF	on	Final Shape function key is enabled.

	off	Final Shape function key is disabled.
textType	implicit	Type of data stream is set to Implicit.
	visual	Type of data stream is set to Visual.
orientation	LTR	Left−to−right default screen orientation.
	RTL	Right−to−left default screen orientation.
symmetric	on	Symmetric Swapping enabled.
	off	Symmetric Swapping disabled.
numShape	bilingual	Numeral shaping is set to bilingual.
	hindi	Numerals are represented in Hindi.
	arabic	Numerals are represented in Arabic/Hebrew.
	passthru	Numerals are represented in passthru.
charShape	automatic	Arabic characters are shaped automatically.
	passthru	Arabic characters are displayed in passthru mode.
	isolated	Arabic characters are displayed in isolated mode.
	initial	Arabic characters are displayed in initial mode.
	final	Arabic characters are displayed in final mode.
	middle	Arabic characters are displayed in middle mode.
maps		Specifies the page code directory to be used for Keyboard. layering, input, output and symmetric character swapping.
expandTail	on	Writes "seen"−like characters and their tails on two cells.
	off	Writes "seen"−like characters and their tails on one cell.
nobidi	on	Activates BIDI mode.
	off	Turn off BIDI mode.
noNulls	on	Replaces nulls by spaces.
	off	Leaves nulls as null, no replacement of spaces.

Related Information

The **aixterm** command, the **telnet, tn, or tn3270** command.

Bidirectionality and Arabic Character Shaping Overview*AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

bugfiler Command

Purpose

Automatically stores bug reports in specified mail directories.

Syntax

bugfiler [**−d**] [**−m** *MessageMode*] [**−b** *BugUserName*] [*MailDirectory*]

Description

The **bugfiler** command automatically intercepts bug reports, summarizes them, and stores them in the appropriate folders in the directory specified by the *MailDirectory* variable.

The mail delivery program starts the **bugfiler** command through a line in the **/etc/aliases** file. The line has the following format:

```
bugs:"|/usr/lib/bugfiler $HOME/bugstuff"
```

In the example, the bug reports are placed in the **$HOME/bugstuff** directory. If no directory is specified, the **bugfiler** command places the bug reports in the **$HOME/mail** default directory.

> **Note:** The **$HOME/mail** directory must be created for the **bugfiler** command to use as a default directory.

If the *BugUserName* is other than `bugs`, the entry in the **/etc/aliases** file should contain a **−b** *BugUserName* flag, as in the following example:

```
hadley:"|/usr/lib/bugfiler -b hadley"
```

In this example, `hadley` is declared the *BugUserName* and all bug reports are placed in the **/home/hadley/mail** default directory. All directories used by the **bugfiler** command must be owned by `hadley`.

The **bugfiler** command reads bug reports from standard input, checks the format of each report, then either sends a message acknowledging receipt ($HOME/*MailDirectory*/**.ack** file) or indicates improper format ($HOME/*MailDirectory*/**.format** file).

Improperly formatted bug reports are filed in the **errors** directory, which the **bugfiler** command creates as a subdirectory of the *MailDirectory* variable. Bug reports must be in the format found in the **/usr/lib/bugformat** file. Use the **sendbug** command to start the **/usr/lib/bugformat** file. The **bugfiler** command summarizes valid bug reports and files them in the folder specified in the `Index:` line of the report. The source directory name in the `Index:` line must match one of the directory names in the mail directory. The **bugfiler** command appends a line in the following format to the *MailDirectory*/**summary** file:

```
DirectoryName/MessageNumber IndexInformation SubjectInformation
```

Note: The **bugfiler** command does not recognize forwarded mail. It notifies the forwarder, not the sender, unless a `Reply-To:` line is included in the header of the report.

Format of Bug Reports

Bug reports must be submitted in ARPA RFC 822 format. The **sendbug** command contains information to compose and mail bug reports in the correct format.

The reports require the following header lines for proper indexing:

`Date:`	Followed by the date the **bugfiler** command receives the report.
`From:`	Followed by the valid return address of the sender.
`Subject:`	Followed by a short summary of the problem.
`Index:`	Followed by the path of the source directory and source file, the version number, and optionally, the **Fix** keyword.

The body of the bug report requires the following lines:

`Description:`	Followed by a detailed description of the problem, suggestion, or complaint.
`Repeat-By:`	Followed by a procedure to repeat the problem.
`Fix:`	Followed by a **diff** command comparing the old and new source files or a description of how to solve the problem. Include the `Fix:` line only if the **Fix** keyword is specified in the `Index:` line.

Redistribution of Bug Reports

Bug reports can be redistributed according to index information in the *MailDirectory*/**.redist** file. The *MailDirectory*/**.redist** file is examined for a line beginning with an index name followed by a tab. Following the index name and tab is a comma−separated list of mail addresses to receive copies of bug reports. If the list continues on multiple lines, each line but the last must end with a \ (backslash). The following is an example of index information in the **.redist** file:

```
myindex   joe@hal,mary@mercutio,martha@banquo,sarah@mephisto,\
dee@hamlet,dewayne@ceasar
```

Flags

−b*BugUserName*	Specifies a new user ID. If the **−b***BugUserName* flag is not specified, the **bugfiler** command defaults to the login name.
−d	Sets debugging on. When the **−d** flag is specified, the **bugfiler** command sends error messages to standard output.
−m*MessageMode*	Sets message protection. The **−m***MessageMode* flag specifies file permissions, using hexadecimal format, for all files that the **bugfiler** command creates.

Examples

1. The syntax of the **bugfiler** command when used with all three flags and a specified *MailDirectory* variable is as follows:

   ```
   hadley:"|/usr/lib/bugfiler −d −m 755 −b hadley
   /home/hadley/bugdir"
   ```

When placed in the **/etc/aliases** file, this line starts debugging, sets file permissions to `rwxr-xr-x`, declares `hadley` as the *BugUserName*, and specifies the **/home/hadley/bugdir** directory.

2. The following is an example of a bug report:

```
Date: Mon, 27 Nov 89 11:26:15 -600
From: a@B
Subject: Read not setting errno correctly
Index: LFS/rdwr.c workstation 3.1

Description: Read not setting errno correctly

Repeat-By: Start an NFS daemon and it receives errors. Errno is
zero.
```

Files

/etc/aliases	Contains system-wide aliases for the mail transport system.
usr/sbin/sendmail	Contains the mail delivery program.
MailDirectory/**summary**	Contains the bug report summaries.
BugUserName/*MailDirectory*/**.ack**	Contains the message sent in acknowledgment.
BugUserName/*MailDirectory*/**.format**	Contains the message sent when format errors are detected.
MailDirectory/**.redist**	Contains the redistribution list for bug reports.

Related Information

The **sendbug** command.

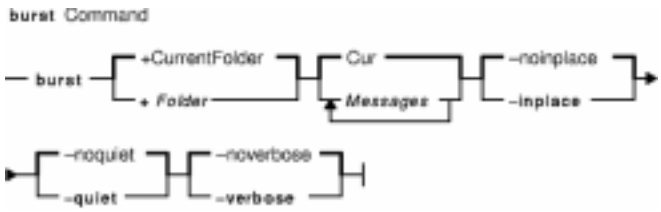
Mail Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

burst Command

Purpose

Divides a message into separate, new messages.

Syntax



burst [*+Folder*] [*Messages*] [**−inplace**] [**−noinplace**] [**−quiet**] [**−noquiet**] [**−verbose**] [**−noverbose**]

Description

The **burst** command allows you to divide a message into multiple, new messages. The **burst** command operates on digests, messages forwarded by the **forw** command, and blind carbon copies sent by the **forw** and **send** commands. Messages created using the **burst** command are numbered consecutively, beginning with the next highest number in the specified folder.

The **burst** command can create about 1000 messages from a single message. However, the **burst** command 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.

By default, the first message extracted from the first digest becomes the current message. If the **−inplace** flag is specified, the first new message becomes the current message.

Flags

+Folder Specifies the folder containing the message to divide. By default, the system uses the current folder.

−help Lists the command syntax, available switches (toggles), and version information.

> **Note:** For Message Handler (MH), the name of this flag must be fully spelled out.

−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 from the divided digest.

> **Attention:** The **burst** command does not place text displayed after the last encapsulated message in a separate message. When you specify the **−inplace** flag, the **burst** command 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, the **burst** command loses these remarks.

Messages Specifies the messages that you want to divide. This parameter may specify several messages, a range of messages, or a single message. Use the following references to specify messages:

Number
Number of the message. When specifying several messages, separate each number with a comma. When specifying a range, separate the first and last number in the range with a hyphen.

Sequence
A group of messages specified by the user. Recognized values include:

all
All messages in the folder.

cur *or* **.** *(period)*
Current message. This is the default.

first
First message in a folder.

last
Last message in a folder.

next
Message following the current message.

prev
Message preceding the current message.

−noinplace Preserves each digest. This is the default.

−noquiet Reports information about messages not in digest format. This flag is the default.

−noverbose Prevents reporting of the actions the **burst** command performs while dividing the digests. This flag is the default.

−quiet Prevents reporting of information about messages not in digest format.

−verbose Reports the actions the **burst** command performs while dividing a digest.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Current−Folder:` Sets the default current folder.

`Msg−Protect:` Sets the protection level for your new message files.

`Path:` Specifies a user's MH directory.

Examples

1. The user receives message `5` from `mickey@mouse` containing several messages in digest form:

```
5+ 03/02 mickey@mouse
6+ 03/02 disney@world
```

To burst message `5` into several, separate messages, enter:

```
burst 5
5+ 03/02 mickey@mouse
6 03/02 disney@world
7 first message in digest
8 second message in digest
9 third message in digest
```

The resulting new messages are appended to the end of the folder. Message `5` remains intact and still contains all four messages.

2. To burst message 5 using the −**inplace** flag, enter:

```
burst 5 -inplace
5+ 03/02 mickey@mouse
6 first message in digest
7 second message in digest
8 third message in digest
9 03/02 disney@world
```

The resulting new messages are placed immediately after the digest, and the **burst** command renumbers all the messages that follow. Message 5 now contains only the header and text of the forwarded message.

Files

$HOME/.mh_profile	Contains the MH user profile.
/usr/bin/burst	Contains the executable form of the **burst** command.

Related Information

The **forw** command, **inc** command, **msh** command, **packf** command, **send** command, **show** command.

The **mh_alias** file format, **mh_profile** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

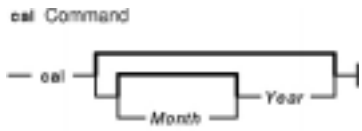
Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

cal Command

Purpose

Displays a calendar.

Syntax



cal [[*Month*] *Year*]

Description

The **cal** command displays a calendar of the specified year or month.

The *Year* parameter names the year for which you want a calendar. Since the **cal** command can display a calendar for any year from 1 through 9999, you must enter the full year rather than just the last two digits. The *Month* parameter identifies the month for which you want the calendar. It can be a number from 1 (indicating January) to 12 (indicating December). If you specify neither the *Year* nor the *Month* parameter, the **cal** command displays the current month. If you specify only one parameter, the **cal** command assumes the parameter is the *Year* parameter and displays the calendar for the indicated year.

> **Note:** The **cal** command does not accept standard input.

The **cal** command uses the appropriate month and day names according to the locale settings. The "National Language Support Overview for Programming" in *AIX General Programming Concepts: Writing and Debugging Programs* contains more information on the **LANG**, **LC_TIME**, **LC_ALL**, and **TZ** environment variables.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. To display a calendar for February, 1994, at your workstation, enter:

   ```
   cal 2 1994
   ```

2. To print a calendar for 1994, enter:

   ```
   cal 1994 | qprt
   ```

3. To display a calendar for the year 84, enter:

   ```
   cal 84
   ```

Files

/usr/bin/cal Contains the **cal** command.

Related Information

The **calendar** command.

National Language Support Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

National Language Support Overview for Programming in *AIX General Programming Concepts: Writing and Debugging Programs*.

calendar Command

Purpose

Writes reminder messages to standard output.

Syntax

calendar [−]

Description

The **calendar** command reads the **calendar** file and displays any line in the file that contains today's or tomorrow's date. The **calendar** file is user−created and must be in the same directory from which you run the **calendar** command. Typically, the **calendar** file resides in your home directory.

If you run the **calendar** command on a Friday, the **calendar** command displays all lines containing the dates for that Friday as well as the subsequent Saturday, Sunday, and Monday. The command does not recognize holidays.

The **calendar** command recognizes date formats such as *MonthDay*, *AbbreviationDate*, and *MonthNumeral*/*Date*. Examples of these formats include December 7, Dec. 7 and 12/7. The **calendar** command also recognizes the special character * (asterisk) when followed by a / (slash). It interprets */7, for example, as signifying the seventh day of every month. The **calendar** command does not recognize formats such as 7/*, 7 December, 7/12, * 7 or DEC. 7.

If the system administrator has created a **calendar** file for all users, you can access this file by placing the following line at the beginning of your local **calendar** file:

```
#include <FileName>
```

The actual value of the *FileName* variable is determined by the system administrator. The name of this file does not have to be **calendar**. When you run the **calendar** command, it displays reminders that were stored in your local **calendar** file as well as those stored in the file specified by the *FileName* variable.

> **Note:** When the **calendar** file contains include statements, the **calendar** command runs the **calendar** file through the C preprocessor. To use include statements with the **calendar** file, the C preprocessor, which is contained in the **/usr/ccs/lib/cpp** file, must be installed on the operating system.

For you to get reminder service, your **calendar** file must have read permission for others. See the **chmod** command for information on setting permissions.

Flag

− Calls the **calendar** command for everyone having a **calendar** file in the home directory. The

calendar command sends reminders using the **mail** command instead of writing the results to standard output.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. A typical **calendar** file might look like the following:

```
*/25 − Prepare monthly report
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
```

   To run the **calendar** command, enter:

```
calendar
```

   If today is Friday, August 24, then the **calendar** command displays the following:

```
*/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
```

2. A **calendar** file that contains an include statement might look like the following:

```
#include </tmp/out>
1/21 −Annual review
1/21 −Weekly project meeting
1/22 *Meet with Harrison in Dallas*
Doctor's appointment − 1/23
1/23 −Vinh's wedding
```

   To run the **calendar** command, enter:

```
calendar
```

   If today is Wednesday, January 21, then the **calendar** command displays the following:

```
Jan.21 Goodbye party for David
Jan.22 Stockholder meeting in New York
1/21 −Annual review
1/21 −Weekly project meeting
1/22 *Meet with Harrison in Dallas*
```

   The results of the **calendar** command indicate the `/tmp/out` file contained the following lines:

```
Jan.21 Goodbye party for David
Jan.22 Stockholder meeting in New York
```

Files

$HOME/calendar	Contains the **calendar** command.
/usr/lib/calprog	Contains the program that determines dates.
/usr/ccs/lib/cpp	Contains the C preprocessor.
/etc/passwd	Contains basic user attributes.

Related Information

The **cal** command, **chmod** command, **mail** or **Mail** command.

File and Directory Access Modes in *AIX Version 4.3 System User's Guide: Operating System and Devices* introduces file ownership and permissions to access files and directories.

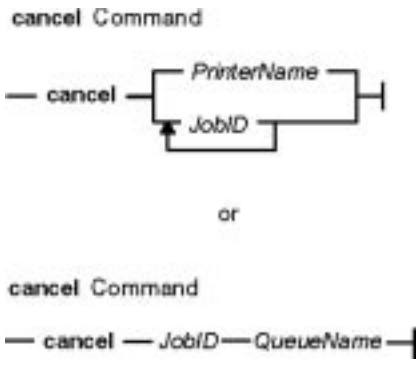
The Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

cancel Command

Purpose

Cancels requests to a line printer.

Syntax



cancel { *JobID ...* | *PrinterName* }

or

cancel*JobID QueueName*

Description

The **cancel** command cancels line printer requests that were made by the **lp** command.

Specifying the following cancels the local print jobs:

- *JobID* cancels the print request, even if it is currently printing.
- *PrinterName* cancels the printing of your jobs on the specified queue. (If you have root user authority, all jobs on the queue are deleted.)

In AIX 4.3.2 and above, **qstatus** was enhanced to improve the administration of local queues showing duplicate 3−digit job numbers. You can use the **−W** flag with the **enq**, **qchk**, **lpstat**, and **lpq** status commands to display more job number digits.

If your queue display shows duplicate 3−digit job numbers, use **qchk −W** to list job numbers with greater precision. You can then cancel a specific job.

For example, `qchk` might display job number `123` twice while, `qchk −W` would display job number `1123` and `2123`. If you want to cancel job number 2123, specifying `cancel 123`, causes the **qdaemon** to cancel the first matching job number it finds in its internal list, which may be `1123`. By having the additional information that the **−W** flag provides, you can cancel a specific job number.

And for remote print jobs, both the *JobID* and remote *QueueName* must be specified in order to explicitly cancel a job on a remote queue.

Notes:

1. You must have root−user authority, or be a member of the **print** group, to cancel print requests that were not submitted by your current ID.
2. The *JobID* must be a number.
3. If you enter `cancel −?`, the system displays the following error message:

```
enq: (FATAL ERROR): 0781−048: Bad queue or device name: −?
```

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Files

/var/spool/qdaemon/*	Contains temporary copies of enqueued files.
/var/spool/lpd/qdir/*	Contains job description files for print jobs.
/usr/bin/cancel	Contains the command file.

Related Information

The **enable** command, **enq** command, **lp** command, **lpstat** command, **qcan** command.

Canceling a Print Job (qcan Command) in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

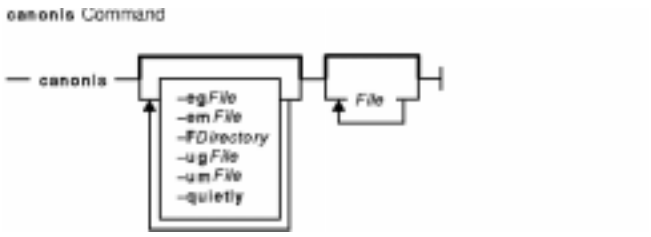
Printers, Print Jobs, and Queues Overview for Users in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

canonls Command

Purpose

Processes **troff** command output for the Canon LASER SHOT in LIPS III mode.

Syntax



canonls [−**eg***File*] [−**em***File*] [−**F***Directory*] [−**quietly**] [−**ug***File*] [−**um***File*] [*File* ...]

Description

The **canonls** command processes **troff** command output for the Canon LASER SHOT in LIPS III mode. This command is provided exclusively for Japanese language support.

The **canonls** command processes one or more files specified by the *File* parameter. If no file is specified, the **canonls** command reads from standard input.

The **canonls** command uses font files in the **/usr/lib/font/devcanonls** directory that have command names ending with **.out**. The **canonls** command does not produce correct output unless these files are provided.

Flags

−**eg***File*	Specifies the Gothic font for the IBM Japanese extended character set. By default, the **canonls** command uses the Gothic font found in the **/usr/lib/X11/fonts/JP/IBM_JPN23G.snf** file.
−**em***File*	Specifies the Mincho font for the IBM Japanese extended character set. By default, the **canonls** command uses the Mincho font found in the **/usr/lib/X11/fonts/JP/IBM_JPN23.snf** file.
−**F***Directory*	Specifies a directory name as the place to find font files. By default, the **canonls** command looks for font files in the **/usr/lib/font/devvcanonls** directory.
−**quietly**	Suppresses all nonfatal error messages.
−**ug***File*	Specifies the Gothic font for the user−defined characters of Japanese. By default, the **canonls** command uses the Gothic font found in the **/usr/lib/X11/fonts/JP/IBM_JPN23G.snf** file.
−**um***File*	Specifies the Mincho font for the user−defined characters of Japanese. By default, the **canonls** command uses the Gothic font found in the **/usr/lib/X11/fonts/JP/IBM_JPN23.snf** file.

Example

To process the `reports` file for the Canon LASER SHOT printer, enter:

```
troff reports |canonls | qprt −dp
```

The **canonls** command first processes the output of the **troff** command, then sends the file to a print queue.

File

/usr/lib/font/devcanonls/*.out Contains font files.

Related Information

The **troff** command.

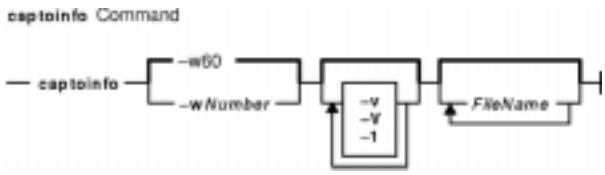
The **troff** font file format.

captoinfo Command

Purpose

Converts a **termcap** file to a **terminfo** descriptor file.

Syntax



captoinfo [**−w***Number*] [**−v**] [**−V**] [**−1**] [*FileName*...]

Description

The **captoinfo** command converts a **termcap** source file to a **terminfo** source file and displays it on the screen. The **termcap** file format is an older format. The **termcap** and **terminfo** files differ mainly in the capability names and the entry syntax. Therefore, the **captoinfo** command only makes the syntactical transformations and vocabulary substitutions. The command also strips obsolete **termcap** capabilities such as nc, and 2−character **termcap** names like D3.

By default, the **captoinfo** command converts the **termcap** description for the terminal specified by the **TERM** environment variable. The command reads the description of the terminal from the **/etc/termcap** file and outputs a **terminfo**−style description. If you specify the *Filename* parameter, the command converts all the descriptions in the file to **terminfo** format.

You can redirect the output of the **captoinfo** command to a file.

Flags

−v Turns on the verbose mode.

−V Displays the version number.

−w*Number* Defines the line width of the **terminfo** entry. The **captoinfo** command fits as many **terminfo** fields in this width as is possible on the output line. A **terminfo** field consists of a capability name and a corresponding value. If you specify the **−w** flag, you must specify a *Number* parameter. By default, the line width is 60.

Notes:

1. If the width you specify is too small to contain even one field, the command displays one field per line.
2. If the width you specify is zero or negative, the line width will be set to 60.

−1 Displays one **terminfo** field per line.

Examples

1. To convert the **termcap** file **Wyse50.tc** to a **terminfo** file and see the results on the display, enter:

```
captoinfo   Wyse50.tc
```

2. To convert the **termcap** file **Wyse50.tc** to a **terminfo** file and save the results, enter:

   ```
   captoinfo   Wyse50.tc   >   Wyse50.ti
   ```

3. To display one **terminfo** field per line and see more information, enter:

   ```
   captoinfo   -1   -v   Wyse50.tc
   ```

4. To produce a **terminfo** description of an ibm3101 terminal defined by the **TERM** environment variable, enter:

   ```
   captoinfo -w 40
   ```

   The **captoinfo** command converts the ibm3101 description in the **/etc/termcap** file into a **terminfo** description and produces a description with a 40 character width. The output of the command is similar to the following:

   ```
   ibm|ibm3101|3101|i3101|IBM 3101-10,
            am, xon,
            cols#80, lines#24,
            bel=^G, clear=\EK, cr=\r, cub1=\b,
            cud1=\n, cuf1=\EC,
            cup=\EY%p1%'\s'%+%c%p2%'\s'%+%c,
            cuu1=\EA, ed=\EJ, el=\EI,
            home=\EH, ht=\t, ind=\n,
            kcub1=\ED, kcud1=\EB, kcuf1=\EC,
            kcuu1=\EA,
   ```

Related Information

The **terminfo** file format.

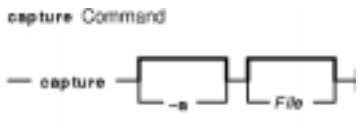
The Curses Overview for Programming in *AIX General Programming Concepts: Writing and Debugging Programs*.

capture Command

Purpose

Allows terminal screens to be dumped to a file.

Syntax



capture [**−a**] [*File*]

Description

The **capture** command allows a user to dump everything printed on the user's terminal to a file. The screen is printed to the file specified by the *File* parameter or to the **screen.out** file if no file is specified. If the **−a** flag is specified, the **capture** command appends the contents of the screen to the file.

In order to dump the screen to a file, the **capture** command creates a shell that emulates a VT100 terminal and maintains a record of what is being displayed on the screen. The **SHELL** environment variable determines the shell created. If the **SHELL** environment variable is not set, the **/usr/bin/bsh** shell is the default. The **TERM** environment variable is set to `TERM=vt100`. If, while running the **capture** command, the program asks for the terminal type in use, the user must enter `vt100`.

The Ctrl−P key sequence is the default keystroke to cause a screen dump to be performed. This can be changed by setting the **SCREENDUMP** environment variable to the 3−digit octal value of the desired screen dump key. For example, setting:

```
SCREENDUMP=014
```

changes the screen dump keystroke to Ctrl−L. Trying to set the **SCREENDUMP** environment variable by entering `^L` or `'\014'` results in an error message.

To stop the screen capture process, use the Ctrl−D key sequence or type `exit`. The system displays the message, `You are NO LONGER emulating a vt100 terminal`.

Flags

−a Appends the screen contents to the specified file or, if no file is specified, to the **screen.out** file.

Files

/usr/bin/capture Contains the **capture** command.

Related Information

The **bsh** command, **csh** command, **ksh** command, **script** command.

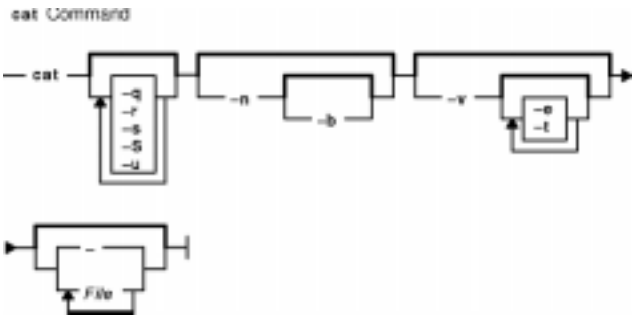
The Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output and how to use redirect and pipe symbols.

cat Command

Purpose

Concatenates or displays files.

Syntax



cat [**−q**] [**−r**] [**−s**] [**−S**] [**−u**] [**−n** [**−b**]] [**−v** [**−e**] [**−t**]] [**−** | *File* ...]

Description

The **cat** command reads each *File* parameter in sequence and writes it to standard output. If you do not specify a file name, the **cat** command reads from standard input. You can also specify a file name of **−** (dash) for standard input.

> **Attention:** Do not redirect output to one of the input files using the redirection symbol, > (caret). If you do this, you lose the original data in the input file because the shell truncates the file before the **cat** command can read it. See "Redirecting Input and Output in the Korn Shell" in *AIX Version 4.3 System User's Guide: Operating System and Devices* for more information.

Flags

−b Omits line numbers from blank lines, when specified with the **−n** flag.

−e Displays a $ (dollar sign) at the end of each line, when specified with the **−v** flag.

−n Displays output lines preceded by line numbers, numbered sequentially from 1.

−q Does not display a message if the **cat** command cannot find an input file. This flag is identical to the **−s** flag.

−r Replaces multiple consecutive empty lines with one empty line. This flag is identical to the **−S** flag.

−s Does not display a message if the **cat** command cannot find an input file. This flag is identical to the **−q** flag.

> **Note:** Previously, the **−s** flag handled tasks now assigned to the **−S** flag.

−S Replaces multiple consecutive empty lines with one empty line. This flag is identical to the **−r** flag.

−t Displays tab characters as `^I` if specified with the **−v** flag.

−u Does not buffer output.

−v Displays nonprinting characters as visible characters.

− Allows standard input to the **cat** command.

Exit Status

This command returns the following exit values:

0 All input files were output successfully.
>0 An error occurred.

Examples

Attention: Do not redirect output to one of the input files using the redirection symbol, > (caret).

1. To display a file at the workstation, enter:

   ```
   cat notes
   ```

   This command displays the data in the `notes` file. If the file is more than one less than the number of available display lines, some of the file scrolls off the screen. To list a file one page at a time, use the **pg** command.

2. To concatenate several files, enter:

   ```
   cat section1.1 section1.2 section1.3 >section1
   ```

   This command creates a file named `section1` that is a copy of `section1.1` followed by `section1.2` and `section1.3`.

3. To suppress error messages about files that do not exist, enter:

   ```
   cat −q section2.1 section2.2 section2.3 >section2
   ```

   If `section2.1` does not exist, this command concatenates `section2.2` and `section2.3`. The result is the same if you do not use the **−q** flag, except that the **cat** command displays the error message:

   ```
   cat: cannot open section2.1
   ```

   You may want to suppress this message with the **−q** flag when you use the **cat** command in shell procedures.

4. To append one file to the end of another, enter:

   ```
   cat section1.4 > > section1
   ```

   The >> (two carets) appends a copy of `section1.4` to the end of `section1`. If you want to replace the file, use the > (caret).

5. To add text to the end of a file, enter:

   ```
   cat >>notes
   Get milk on the way home
   Ctrl−D
   ```

   This command adds `Get milk on the way home` to the end of the file called `notes`. The **cat** command does not prompt; it waits for you to enter text. Press the Ctrl−D key sequence to indicate you are finished.

6. To concatenate several files with text entered from the keyboard, enter:

```
cat section3.1 - section3.3 >section3
```

This command concatenates `section3.1`, text from the keyboard (indicated by the minus sign), and `section3.3`.

7. To concatenate several files with output from another command, enter:

```
li | cat section4.1 - >section4
```

This command copies `section4.1`, and then the output of the **li** command to the `section4` file.

Files

/usr/bin/cat Contains the **cat** command.

Related Information

The **cp** command, **ksh** command, **pcat** command, **pg** command, **pr** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

File Systems and Directories Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

The Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

Redirecting Input and Output in the Korn Shell in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

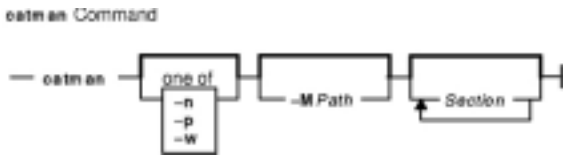
File and Directory Access Modes in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

catman Command

Purpose

Creates the cat files for the manual.

Syntax



catman [**−n**| **−p** | **−w**] [**−M***Path*][*Section ...*]

Description

The **catman** command creates the preformatted versions of the online manual from the **nroff** command input files. The **catman** command examines each manual page and recreates those pages whose preformatted versions are missing or out of date. If any changes are made, the **catman** command recreates the **whatis** command database.

Flags

−M *Path*

Updates manual pages located in the set of directories specified by the *Path* variable (the **/usr/share/man** directory by default).The *Path* variable has the form of a colon (:) separated by a list of directory names. For example:

```
'/usr/local/man:/usr/share/man'
```

If the environment variable **MANPATH** is set, its value is used for the default path. If the **nroff** command source file contains a line such as:

```
' .so manx/yyy.x'
```

a symbolic link is made in the **catx** directory to the appropriate preformatted manual page. This allows easy distribution of the preformatted manual pages among a group of associated machines using the **rdist** command.

The **nroff** command sources need not be distributed to all machines, thus saving the associated disk space.

For example, a local network of five machines (called mach1 through mach5) has mach3 with the manual page **nroff** command sources. Every night, mach3 runs the **catman** command by using the **cron** daemon and later runs the **rdist** command with a **distfile** file that looks like the following:

```
MANSLAVES = (mach1 mach2 mach4 mach5)
MANUALS = (/usr/share/man/cat[1−8no] /usr/share/man/whatis)
${MANUALS}    ->    ${MANSLAVES}
install −R;
```

```
notify root;
```

−n Prevents creation of the **whatis** command database.

−p Prints the names of the manual pages that need to be recreated or updated without recreating or updating them.

−w Reads the BSD−style manual pages in the **/usr/share/man/cat?/*.*** and **/usr/share/man/man?/*.*** files; then reads the hypertext information bases installed under the **/usr/share/man/info** directory, and creates the **/usr/share/man/whatis** database. If the **MANPATH** environment variable is set, a different database will be created in each directory that appears in **MANPATH**. Each of these will be built from the **cat?/*.*** and **man?/*.*** files for that particular directory. The HTML Library information will be built into the default **/usr/share/man/whatis** database if it is included in **MANPATH** or if the **MANPATH** environment variable is not set.

Examples

To update manual sections 1, 2, and 3 only, enter:

```
catman 123
```

Files

/usr/lbin/getNAME	Contains the command to create the **whatis** database.
/usr/share/man	Specifies the default manual directory location.
/usr/share/man/man?/*.*	Contains the raw (the **nroff** command input) manual sections.
/usr/share/man/cat?/*.*	Contains preformatted manual pages.
/usr/share/man/whatis	Contains the **whatis** command database.
/usr/lbin/mkwhatis	Contains the command script to make the **whatis** command database.

Related Information

The **man** command, **nroff** command and **rdist** command.

The **cron** daemon.

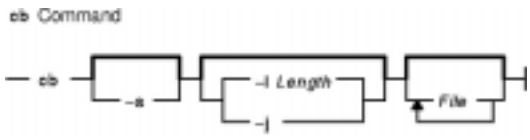
The **distfile** file.

cb Command

Purpose

Puts C source code into a form that is easily read.

Syntax



cb [**−s**] [**−l** *Length* | **−j**] [*File* ...]

Description

The **cb** command reads C programs from standard input or from specified files 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. Note that punctuation in preprocessor statements can cause indentation errors.

For best results, use this command on source code that is syntactically correct.

Flags

−j Joins lines that are split. Ignored if **−l** flag is given.

−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, enter:

```
cb pgm.c > pgm.pretty.c
```

Files

/usr/ccs/bin/cb Contains the **cb** command.

/usr/bin/cb Symbolic link to the **cb** command.

Related Information

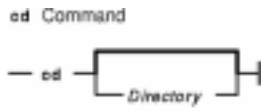
The **indent** command.

cd Command

Purpose

Changes the current directory.

Syntax



cd [*Directory*]

Description

The **cd** command moves you from your present directory to another directory. You must have execute (search) permission in the specified directory.

If you do not specify a *Directory* parameter, the **cd** command moves you to your login directory (**$HOME** in the **ksh** and **bsh** environments, or **$home** in the **csh** environment). If the specified directory name is a full path name, it becomes the current directory. A full path name begins with a / (slash) indicating root directory, a **.** (dot) indicating current directory, or a **..** (dot−dot) indicating parent directory. If the directory name is not a full path name, the **cd** command searches for it relative to one of the paths specified by the **$CDPATH** shell variable (or **$cdpathcsh** variable). This variable has the same syntax as, and similar semantics to, the **$PATH** shell variable (or **$path csh** variable).

> **NOTE:** Running **/usr/bin/cd** from a shell does not change the shell's working directory.

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Examples

1. To change to your home directory, enter:

   ```
   cd
   ```

2. To change to an arbitrary directory, enter:

   ```
   cd /usr/include
   ```

   This changes the current directory to `/usr/include`.

3. To go down one level of the directory tree, enter:

   ```
   cd sys
   ```

If the current directory is `/usr/include` and it contains a subdirectory named `sys`, then `/usr/include/sys` becomes the current directory.

4. To go up one level of the directory tree, enter:

   ```
   cd ..
   ```

The special file name, `..` (dot−dot), refers to the directory immediately above the current directory.

Related Information

The **bsh** command, **csh** command, **ksh** command, **pwd** command.

The **chdir** subroutine.

Directory Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes the structure and characteristics of directories in the file system.

File Systems and Directories Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes shells, the different types, and how they affect the way commands are interpreted.

cdc Command

Purpose

Changes the comments in a SCCS delta.

Syntax

cdc −**r***SID* [−**m** [*ModificationRequestList*]] [−**y** [*Comment*]] *File ...*

Description

The **cdc** command changes the Modification Requests (MRs) and comments for the specified SCCS delta (the *SID* variable) 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.

You can change the comments and MRs for an SID only if you made the SID or you own the file and the directory.

Flags

−**m**[*ModificationRequestList*] Supplies a list of MR numbers for the **cdc** program to add or delete in the SID specified by the −**r** flag. You can only use this flag if the specified file has the **v** header flag set. A null MR list has no effect.

In the actual *ModificationRequestList* parameter, 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 them with a comment line indicating that the MRs were 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 workstation, 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−line character or a blank line. MRs are always read before comments (see the −**y** flag).

If the **v** header flag has a value, the **cdc** command interprets the value as the name of a program that validates MR numbers. If the MR number validation program returns a nonzero exit value, the **cdc** command stops and does not

change the MRs.

−r*SID* Specifies the SCCS identification number of the delta for which the **cdc** command will change the comments or MRs.

−y[*Comment*] Specifies comment text to replace an existing comment for the delta specified by the **−r** flag. The **cdc** command keeps the existing comments but precedes them by a comment line stating that they were changed. A null *Comment* value 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 workstation, 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.

Example

To change the comment for SID 1.3 of SCCS file `s.text.c` to `"new comment"`, enter:

```
cdc −r1.3 −y"new comment" s.test.c
```

Files

/usr/bin/cdc Contains the path to SCCS **cdc** command**.**

Related Information

The **admin** command, **delta** command, **get** command, **prs** command, **sccshelp** command.

The **sccsfile** file format.

Source Code Control System (SCCS) Overview in *AIX General Programming Concepts: Writing and Debugging Programs*.

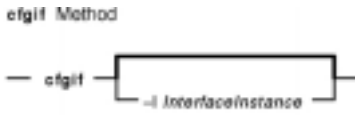
List of SCCS Commands in *AIX General Programming Concepts: Writing and Debugging Programs*.

cfgif Method

Purpose

Configures or activates one or all network interface (IF) instance(s) defined in the system configuration database.

Syntax



cfgif [**−l** *InterfaceInstance*]

Description

The **cfgif** method configures or activates one or all IF instance(s) of TCP/IP defined in the system configuration database. The **cfgif** method performs the following steps:

1. Retrieves the attributes associated with the Interface Program from the customized database. The attributes may include network address, network mask, security level and other related information.
2. Invokes the **ifconfig** command to load the IF instance using the customized attributes. The **ifconfig** command will load the appropriate interface program if it has not already been loaded.
3. Calls the **ifconfig** command to attach a routine to establish a path between the interface instance and the adapter.
4. Sets the status of a particular IF instance to "AVAILABLE" in the customized database. All the IF instances are set to "DEFINED" at system reboot. When the **cfgif** method is invoked during boot time or from the command line, the IF instance(s) are then made available.

Flags

−l *InterfaceInstance*	Specifies the interface instance to configure. If the instance name is specified, only that Interface instance is configured. If this flag is not used, all Interface instances in the defined state are configured.
−2	Indicates that **ifconfig** will be invoked from the second phase of IPL so that a hex value will be shown on the front panel display. This flag should not be used during runtime.

Examples

1. To configure a particular token−ring IF instance, enter the following command. Note that `tr0` is the logical name for the token−ring IF instance. It should be defined using the **defif** method.

   ```
   cfgif −l tr0
   ```

2. To configure all IF instances, use the following command:

   ```
   cfgif
   ```

Related Information

The **chdev** command, **defif** method, **definet** method, **ifconfig** command, **mkdev** command.

The **cfginet** method.

The **odm_run_method** subroutine.

Object Data Manager (ODM) Overview for Programmers in *AIX General Programming Concepts: Writing and Debugging Programs*.

Writing a Device Method in *AIX Kernel Extensions and Device Support Programming Concepts*.

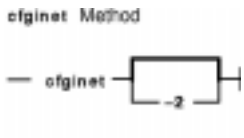
TCP/IP Network Interfaces, TCP/IP Addressing in *AIX Version 4.3 System Management Guide: Communications and Networks*.

cfginet Method

Purpose

Loads and configures an Internet instance and its associated IF instances.

Syntax



cfginet [**−2**]

Description

The **cfginet** method loads and configures an instance of TCP/IP (an Internet instance) by performing the following steps:

1. Loads the protocol code.
2. Initializes entries in the Address Family Domain switch table and in the Network Input switch table.
3. Sets the status flag of the Internet instance to AVAILABLE.
4. Invokes the **hostname** command and the **route** command to set the hostname and static routes. The hostname and routing data are retrieved from the configuration database.

   Note: The **cfginet** method is a programming tool and should not be executed from the command line.

Flag

−2 Specifies the second phase of IPL device configuration. A predetermined hex value will be displayed on the front panel. This option should not be used during regular run−time operation.

Example

To configure an Internet instance on a host, enter the method in the following format:

```
cfginet
```

Related Information

The **mkdev** command.

The **odm_run_method** subroutine.

TCP/IP Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Object Data Manager (ODM) Overview for Programmers in *AIX General Programming Concepts: Writing and Debugging Programs*.

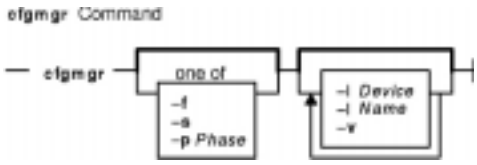
Writing a Device Method in *AIX Kernel Extensions and Device Support Programming Concepts*.

cfgmgr Command

Purpose

Configures devices and optionally installs device software by running the programs specified in the Configuration Rules object class.

Syntax



cfgmgr [**−f** | **−s** | **−p***Phase*] [**−i***Device*] [**−l***Name*] [**−v**]

Description

The **cfgmgr** command configures devices and optionally installs device software into the system. The devices to be configured are controlled by the Configuration Rules object class, which is part of the Device Configuration database. Each configuration rule specifies three items:

- The full path name of an executable program to run
- When to run the program (in relation to the other rules)
- In which phase to run the program

During system boot, the **cfgmgr** command configures all the devices that are necessary to use the system. System boot is a two−step process. The first step is called phase 1, and it begins when the kernel is brought into the system and the boot file system is initialized. During this phase, the **cfgmgr** command is invoked, specifying this as phase 1 by using the **−f** flag. The **cfgmgr** command executes all of the phase 1 configuration rules, which results in the base devices being configured. After this, phase 2 execution begins, and the **cfgmgr** command is called with the **−s** flag.

The **cfgmgr** command recognizes three phases of configuration rules:

- Phase 1
- Phase 2 (second boot phase for normal boot)
- Phase 3 (second boot phase for service boot)

Normally, the **cfgmgr** command executes all the rules for the phase specified during invocation (for example, phase 1 rules for the **−f** flag). However, if the **−l** flag is used, the **cfgmgr** command configures only the named device and its children.

If the **cfgmgr** command is invoked without a phase option (for example, without the **−f**,**−s**, or **−p** flags), then the command executes the phase 2 rules. The only way to run the phase 3 rules is with the **−p** flag.

The configuration rules for each phase are ordered based on the values specified in the `seq` field. This field is an integer that specifies the priority in which to execute this rule, relative to the other rules for this phase. The higher the number specified by the `seq` field, the lower the priority; for example, a value of 1 specified in the `seq` field is executed before a rule with a value of 10. There is one exception: a `seq` field value of 0 implies a "don't care" condition, and any `seq` field value of 0 is executed last. Therefore, a `seq` field value of 1 is the

highest priority (first to execute).

If there are any devices detected that have no device software installed when configuring devices, the **cfgmgr** command returns a warning message with the name or a list of possible names for the device package that must be installed. If the specific name of the device package is determined, it is displayed as the only package name, on a line below the warning message. If the specific name cannot be determined, a colon−separated list of possible package names is displayed on a single line. A package name or list of possible package names is displayed for each of the devices, if more than one device is detected without its device software.

The system displays the following warning message when devices without their device software are detected:

```
cfgmgr: 0514−621 WARNING: The following  device packages are
        required for device support but are not currently
        installed.
devices.pci.22100020
devices.pci.14101800
devices.pci.scsi:devices.pci.00100300:devices.pci.NCR.53C825
```

In this example, two devices were found whose software is missing, and the **cfgmgr** command displayed the names of the device packages that must be installed. A third device whose software is missing was also found, but in this case, the **cfgmr** command displays several possible device package names.

When more than one possible package name is identified for a device, typically only one of the names will actually correspond to a device package on the install medium. This is the package to install. However, in some cases, more than one of the names will correspond to actual device packages on the install medium. In this case, the first package name in the list for which there is an actual device package on the install medium is the package that must be installed. If the **cfgmgr** command is used with the **−i** flag, then the correct packages will be installed.

If you invoke the **cfgmgr** command with the **−i** flag, the command attempts to install device software automatically for each new detected device. The *Device* variable of the **−i** flag specifies where to find the installation medium. The installation medium can be a hardware device (such as a tape or diskette drive), a directory that contains installation images, or the installation image file itself.

> **Attention:** To protect the Configuration database, the **cfgmgr** command is not interruptible. Stopping this command before execution is complete could result in a corrupted database.

Flags

−f Specifies that the **cfgmgr** command executes the phase 1 configuration rules. This flag is not valid at run time (after system start).

−i *Device* Specifies the location of the installation medium.

−l *Name* Specifies the named device to configure along with its children.

−p*Phase* Specifies that the **cfgmgr** command executes the specified phase.

−s Specifies that the **cfgmgr** command executes the phase 2 configuration rules.

−v Specifies verbose output. The **cfgmgr** command writes information about what it is doing to standard output.

Configuration Rules

phase Specifies whether this rule belongs to phase 1 , phase 2 , or phase 3 (second boot phase for service mode).

seq S pecifies as an integer, the relative priority of this rule .

rule A string containing the full path name of a program to execute (can also contain any flags, but they must follow the program name, as this whole string is executed as if it were typed in on the command line).

Security

Access Control: Only the root user and members of the system group should have execute (x) access to this command.

Auditing Event:

Event	Information
DEV_Configure	Device name

Examples

The following examples are based on the configuration rules containing the following information:

```
phase   seq        rule


  1     1        /usr/lib/methods/defsys
  1     10       /usr/lib/methods/deflvm
  2     1        /usr/lib/methods/defsys
  2     5        /usr/lib/methods/ptynode
  2     10       /usr/lib/methods/startlft
  2     15       /usr/lib/methods/starttty
  3     1        /usr/lib/methods/defsys
  3     5        /usr/lib/methods/ptynode
  3     10       /usr/lib/methods/startlft
  3     15       /usr/lib/methods/starttty
```

1. When the **cfgmgr** command is invoked with the **−f** flag, the command gets all of the configuration rules with `phase = 1` and executes them in the following order:
   - **/usr/lib/methods/defsys**
   - **/usr/lib/methods/deflvm**

   Note: The **−f** flag cannot be used during run time.
2. When the **cfgmgr** command is run with the **−s** flag, the command gets all of the configuration rules with `phase = 2` and executes them in the following order:
   - **/usr/lib/methods/defsys**
   - **/usr/lib/methods/ptynode**
   - **/usr/lib/methods/startlft**
   - **/usr/lib/methods/starttty**
3. When the **cfgmgr** command is run with the **−p** 3 flag, the command gets all of the configuration rules with `phase = 3` and executes them in the following order:
   - **/usr/lib/methods/defsys**
   - **/usr/lib/methods/ptynode**
   - **/usr/lib/methods/startlft**
   - **/usr/lib/methods/starttty**
4. If the **cfgmgr** command is run without a flag, the command functions the same as when used with the **−s** flag.
5. To configure detected devices attached to the SCSI0 adapter, enter:

   ```
   cfgmgr −l scsi0
   ```

6. To install device software automatically during configuration (with the software contained in a directory), enter:

```
cfgmgr -i /usr/sys/inst.images
```

Files

/usr/sbin/cfgmgr	Specifies the command file.
/usr/include/sys/cfgdb.h	Contains numeric representations for fields in the Configuration Rules object class.

Related Information

The **chdev** command, **lsattr** command, **lsdev** command, **mkdev** command, **rmdev** command.

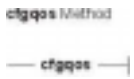
Device Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* describes adding, changing, moving, and removing devices.

cfgqos Method

Purpose

Loads, configures, and activates the Quality of Service (QoS) instance.

Syntax



cfgqos

Description

The **cfgqos** method enables Quality of Service (QoS) for the TCP/IP protocol suite on an AIX host by performing the following steps:

1. Loads the QoS kernel exension
2. Initializes the QoS instance
3. Attaches to the TCP/IP instance
   Note: The **cfgqos** method is a programming tool and is not intended to be invoked from the command line.

Example

To configure QoS on a host, use the following format:

```
cfgqos
```

Related Information

The **cfginet** command, and **ucfgqos** method.

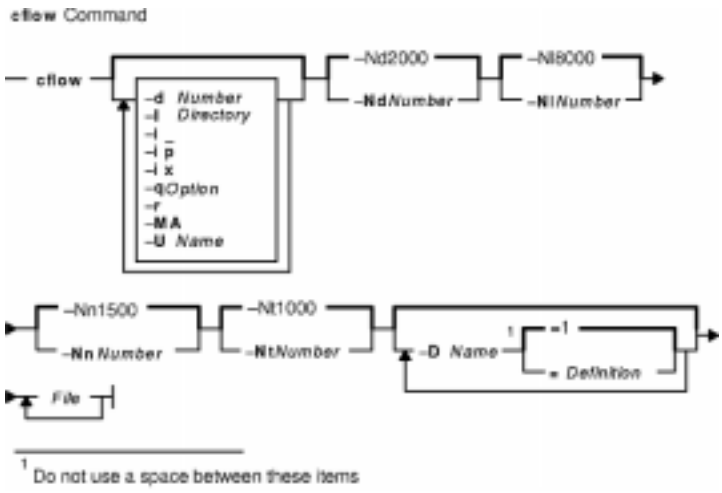
TCP/IP Quality of Service (QoS) in the *AIX Version 4.3 System Management Guide: Communications and Networks*.

cflow Command

Purpose

Generates a C and C++ flow graph of external references.

Syntax



cflow [**−d** *Number*] [**−I** *Directory*] [**−i _**] [**−i p**] [**−i x**] [**−q***Option*] [**−r**] [**−MA**] [**−U** *Name*] [**−Nd***Number*] [**−Nl***Number*] [**−Nn***Number*] [**−Nt***Number*] [**−D** *Name*[**=***Definition*]] *File ...*

Description

The **cflow** command analyzes the C, C++, **yacc**, **lex,** assembler, and object files and writes a chart of their external references to standard output.

> **Note:** Processing of C++ language files by the **cflow** command requires the presence of the IBM C Set++ Compiler/6000 package.

The **cflow** command sends files with the **.y**, **.l**, and **.c** suffixes to the **yacc** command, **lex** command, and **cpp** command for processing. A modified first pass of the **lint** command then processes the **yacc**, **lex**, and **cpp** output, or any **.i** files. The **cflow** command sends files with a **.C** suffix to the C Set++ compiler.

The **cflow** command assembles files with the **.s** suffix, extracting information from the symbol table (as it does with **.o** files). From this output, the **cflow** command produces a graph of external references and writes it to standard output.

Each line of output provides the following information (in order from left to right):

- A line number followed by sufficient tabs to indicate the level of nesting
- The name of the global, a colon, and its definition.

The name is normally a function not defined as external and not beginning with an underline character (see the **−i_** and **−i** inclusion flags).

For information extracted from C and 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, such as **.text** or **.data**. The **cflow** command deletes leading underline characters in C−style external names.

Once the **cflow** command displays a name, later references to the name contain only the **cflow** line number where the definition can be found. For undefined references, **cflow** displays only < > (angled brackets).

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 less than eight spaces per tab stop.

> **Note:** To ensure that the line numbers produced by the **cflow** command match your **lex** and **yacc** files, you must send the **.l** or **.y** file to the **cflow** command.

Flags

−d *Number*	Sets to a decimal integer the depth at which the flow graph is cut off. By default this is a large number. Do not set the cutoff depth to a nonpositive integer.
−i _	Includes names that begin with an underline character. The default excludes these functions (and corresponding data if the **−ix** flag is used).
−i p	Disables ANSI function prototypes. The default option is to fill in undefined function information with available prototype declarations.
−i x	Includes external and static data symbols. The default includes only functions.
−r	Produces an inverted listing that shows the callers of each function, sorted by called function.
−MA	Specifies ANSI mode. The **cflow** command expects ANSI C code in this mode. The default mode of operation is extended mode.
−Nd*Number*	Changes the dimension table size to the *Number* parameter. The default value of *Number* is 2000.
−Nl*Number*	Changes the number of type nodes to the *Number* parameter. The default value of *Number* is 8000.
−Nn*Number*	Changes the symbol table size to the *Number* parameter. The default value of *Number* is 1500.
−Nt*Number*	Changes the number of tree nodes to the *Number* parameter. The default value of *Number* is 1000.

In addition, the **cflow** command recognizes the following flags of the **cpp** command (macro preprocessor):

−D *Name*[=*Definition*]	Defines the *Name* parameter, as if by the **#define** statement. The default *Definition* is 1.
−q*Option*	Passes the **−q***Option* to the preprocessor. For example, `−qmbcs` sets multibyte mode specified by the current locale and `−qidirfirst` modifies the search order for files included with the **#includefile_name** directive.
−**I** *Directory*	Adds the specified *Directory* to the list of directories in which the **cflow** program searches for **#include** files.
−U *Name*	Removes any initial definition of the *Name* parameter, where *Name* is a reserved symbol that is predefined by the particular preprocessor.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To generate a default flow graph of these C files that compose a program, enter:

   ```
   cflow timeout.c kill.c error.c
   ```

2. To produce a **cflow** graph with a single level of nesting of functions, enter:

   ```
   cflow −d1 resam.c pptp.c ptpt.c rrr.c whn.c
   ```

3. To generate a **cflow** graph of a **lex** program, enter:

   ```
   cflow scan.l
   ```

4. To generate a **cflow** graph of the **yacc** program, enter:

   ```
   cflow yaccfile.y
   ```

5. To generate an inverted listing showing the callers of each of the functions in the C files used in example 2, enter:

   ```
   cflow −r resam.c pptp.c ptpt.c rrr.c whn.c
   ```

Files

/usr/ccs/bin/cflow	Driver for the **cflow** command
/usr/ccs/lib/cflow1	Executable for the **cflow** command
/usr/ccs/lib/dag	Executable for the **cflow** command
/usr/ccs/lib/flip	Executable for the **cflow** command
/usr/ccs/lib/lpfx	Executable for the **cflow** command
/usr/ccs/lib/nmf	Executable for the **cflow** command
/var/tmp/cf.*	Temporary files created by the **cflow** command

Related Information

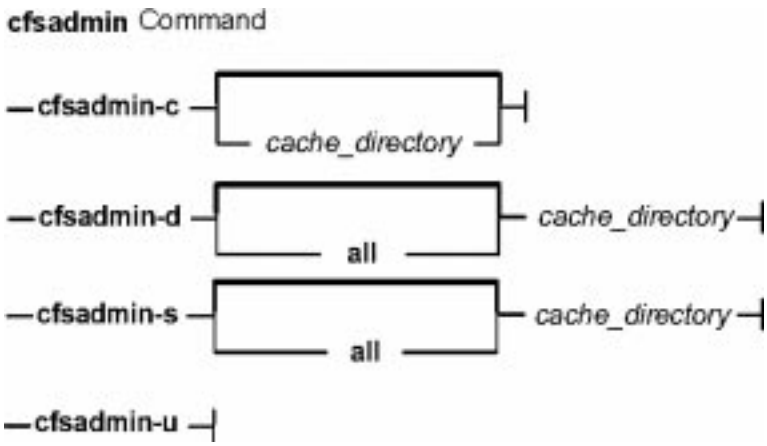
The **as** command, **cpp** command, **lex** command, **lint** command, **nm** command, **pr** command, **yacc** command.

cfsadmin Command

Purpose

Administers disk space used for caching file systems with the Cache File−System (CacheFS).

Syntax



cfsadmin−c*cache_directory*

cfsadmin−d [**all**] *cache_directory*

cfsadmin−s [**all**] *cache_directory*

cfsadmin−u

Description

The **cfsadmin** command provides the following functions:

- cache creation
- deletion of cached file systems
- Listing of cache contents and statistics
- resource parameter adjustment when the file system is unmounted.

For each form of the command, unless the **−u** flag is specified, you must specify a cache directory, that is, the directory under which the cache is actually stored. A path name in the front file system identifies the cache directory. For the of the command, you must specify a mount point.

You can specify a cache ID when you mount a file system with CacheFS, or you can let the system generate one for you. The **−l** flag includes the cache ID in its listing of information. You must know the cache ID to delete a cached file system.

Flags

−c Creates a cache under the directory specified by *cache_directory*. This directory must not exist prior to cache creation.

−d Removes the file system whose cache ID you specify and release its resources, or remove all file systems

in the cache by specifying *cache_directory*. After deleting a file system from the cache, you must run the command to correct the resource counts for the cache.

−l Lists file systems stored in the specified cache, as well as statistics about them. Each cached file system is listed by cache ID. The statistics document resource utilization and cache resource parameters.

−s Requests a consistency check on the specified file system (or all cachefs mounted file systems). The **−s** flag only works if the cache file system was mounted with demandconst enabled. Each file in the specified cache file system is checked for consistency with its corresponding file in the back file system. The consistency check is performed file by file as files are accessed. If no files are accessed, no checks are performed. Using this flag does not result in a sudden storm of consistency checks. The −s option is not currently supported in AIX CacheFS.

−u Updates resource parameters of the specified cache directory. Parameter values can only be increased. To decrease the values, you must remove the cache and recreate it. All file systems in the cache directory must be unmounted when you use this flag. Changes will take effect the next time you mount any file system in the specified cache directory. The **−u** flag with no **−o** flag sets all parameters to their default values.

CacheFS Resource Parameters

You can specify the following cacheFS resource parameters as arguments to the **−o** flag. Separate multiple parameters with commas.

maxblocks=*n* Maximum amount of storage space that CacheFS can use, expressed as a percentage of the total number of blocks in the front file system. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the space the **maxblocks** parameter allows will be available. The default is 90.

minblocks=*n* The minimum amount of storage space, expressed as a percentage of the total number of blocks in the front file system, that CacheFS is always allowed to use without limitation by its internal control mechanisms. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the space the **minblocks** parameter attempts to reserve will be available. The default is 0.

threshblocks=*n* A percentage of the total blocks in the front file system beyond which CacheFS cannot claim resources once its block usage has reached the level specified by **minblocks**.The default is 85.

maxfiles=*n* Maximum number of files that CacheFS can use, expressed as a percentage of the total number of inodes in the front file system. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the inodes the **maxfiles** parameter allows will be available. The default is 90.

minfiles=*n* Minimum number of files, expressed as a percentage of the total number of inodes in the front file system, that CacheFS is always allowed to use without limitation by its internal control mechanisms. If CacheFS does not have exclusive use of the front file system, there is no guarantee that all the inodes the **minfiles** parameter attempts to reserve will be available. The default is 0.

threshfiles=*n* A percentage of the total inodes in the front file system beyond which CacheFS cannot claim inodes once its usage has reached the level specified by **minfiles**. The default is 85.

maxfilesize==*n* Largest file size, expressed in megabytes, that CacheFS is allowed to cache. The default is 30.

Note: You cannot decrease the block or inode allotment for a cache. To decrease the size of a cache, you must remove it and create it again with different parameters.

Examples

1. To create a cache directory named cache, enter:
   ```
   cfsadmin −c /cache
   ```

2. To create named /cache1 that can claim a maximum of 60 percent of the blocks in the front file system, can use 40 percent of the front file system blocks without interference by CacheFS internal control mechanisms, and has a threshold value of 50 percent. The threshold value indicates that after CacheFS reaches its guaranteed minimum, it cannot claim more space if 50 percent of the blocks in the front file system are already used.
   ```
   cfsadmin −c −o maxblocks=60,minblocks=40,threshblocks=50 /cache1
   ```

3. To change the **maxfilesize** parameter for the cache directory /cache2 to 2 megabytes, enter:
   ```
   cfsadmin −u −o maxfilesize=2 /cache2
   ```

4. To list the contents of a cache directory named /cache3 and provides statistics about resource utilization, enter:
   ```
   cfsadmin −l /cache3
   ```

5. To remove the cached file system with cache ID 23 from the cache directory /cache3 and free its resources (the cache ID is part of the information returned), enter:
   ```
   cfsadmin −d 23 /cache3
   ```

6. To remove all cached file systems from the cache directory, enter:
   ```
   cfsadmin −d all /cache3
   ```

7. To check all filesystems mounted with demandconst enabled for consistency. No errors will be reported if no demandconst filesystems were found. Enter:
   ```
   cfsadmin
   ```

Related Information

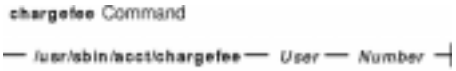
The **mount** command and **fsck_cachefs** command.

chargefee Command

Purpose

Charges end users for the computer resources they use.

Syntax



/usr/sbin/acct/chargefee*User Number*

Description

The **chargefee** command is used by someone with administrative authority to charge the individual specified by the *User* parameter for the number of work units specified by the *Number* parameter. The *Number* value can be an integer or a decimal value.

The **chargefee** command writes a record to the **/var/adm/fee** file. This information is merged with other accounting records by the **acctmerg** command to create the daily report.

> **Note:** You should not share accounting files among nodes in a distributed environment. Each node should have its own copy of the various accounting files.

Security

Access Control: This command should grant execute (x) access only to members of the adm group.

Examples

To charge `smith` for `10` units of work on a financial report, enter:

```
/usr/sbin/acct/chargefee smith 10
```

A record is created in the **/var/adm/fee** file, which the **acctmerg** command will merge with records in other accounting files to produce the daily report.

Files

/usr/sbin/acct The path to the accounting commands.
/var/adm/fee Accumulates the fees charged to each login name.

Related Information

The **acctmerg** command.

For more information about the Accounting System, the preparation of daily and monthly reports, and the

accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

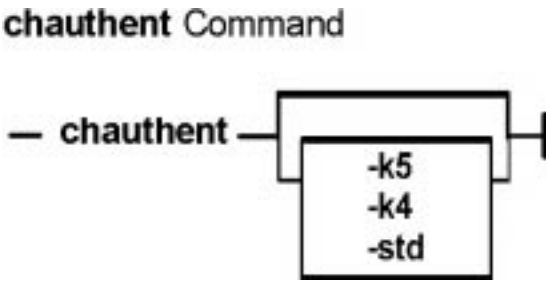
Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains the steps you must take to establish an accounting system.

chauthent Command

Purpose

Changes the configured authentication methods for the system.

Syntax



chauthent [**−k5**] [**−k4**] [**−std**]

Description

The **chauthent** command sets the desired configuration based on the flags the user sets. The authentication methods are set in the order in which the flags are given to the command. If none of the flags are set, then the **rcmds** will be disabled from functioning. If the **−std** flag is set, it must be the last flag set or the command will fail.

> **Note:** The complete order of authentication methods must be specified each time. The command does not modify the current order when replacing it with the new one.

The user must have root authority to execute the command.

The **chauthent** command takes the flags set and calls the **set_auth_method** routine in **libauthm.a** to cause the change.

The **chauthent** command writes an error message to **stderr** and returns a −1 if **set_auth_method** fails.

Flags

−k5 Sets the Kerberos 5 authentication method.

−k4 Sets the Kerberos 4 authentication method.

−std Sets the Standard AIX authentication method.

Examples

1. Set all of the methods in descending order:
   ```
   chauthent  −k5   −k4  −std
   ```
2. Set all of the methods with Kerberos 4 attempted first:
   ```
   chauthent −k4 −k5 −std
   ```
3. Clear all of the methods:
   ```
   chauthent
   ```

Related Information

The **ftp** command, **lsauthent** command, **rcp** command, **rlogin** command, **rsh** command, **telnet**, **tn**, or **tn3270** command.

The **get_auth_method** and **set_auth_method** routines.

Network Overview in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Secure Rcmds in *AIX Version 4.3 System User's Guide: Communications and Networks*.

chclass Command

Purpose

Change a Workload Management class and its limits.

Syntax

chclass −a *Attribute*=*Value* {[−a *Attribute*=*Value*]...} [**−c**|**−m** <keywd>=<value>] [**−d** *Config_dir*] *Name*

Description

The **chclass** command changes attributes for the class identified by the *Name* parameter. The class must already exist. To change an attribute, specify the attribute name and the new value with the Attribute=Value parameter. To change a limit or shares value, use option −c for cpu and −m for memory, with the keyword value in **min**, **max** or **shares**.

Attributes

The following attributes can be changed:

Class properties:

tier The tier value for a class is the position of the class in the hierarchy of resource limitation desirability for all classes. A class with a lower tier value will be more favored. The tier value is a number from 0 to 9 (default is 0).

Class limits and shares, for CPU or memory resource:

min Minimum percentage of the resource (CPU time or memory pages) that must be made available when requested, expressed as a percentage of the total resource available in the system. Possible values range from 0 to 100 (default is 0).

shares Maximum ratio of the resource that can be made available if there is contention. This parameter is expressed in *shares* of the total resource available in the system. The actual ratio of the resource is dynamically computed, proportionally to the shares of all active classes. If a class has no running process, its shares are excluded from the computation. The shares are arbitrary numbers from 1 to 65535 (default is 1).

max Maximum percentage of the resource that can be made available, even if there is no contention. Possible values range from 1 to 100 (default is 100).
Specifying a value different from the default 100 for memory can result in some memory pages remaining unused, while some processes in the class could use more.

Class description:

description The class description text can be composed of any ASCII character, except colon (":").

Flags

−d*Config_dir* Use */etc/wlm/Config_dir* as alternate directory for the properties files. If this flag is not present, the current configuration files in the directory pointed to by **/etc/wlm/current** are

used.

Files

/etc/wlm/current/classes	Contains the names and definitions of the classes for the current workload management configuration.
/etc/wlm/current/limits	Contains the resource limits enforced on the classes for the current workload management configuration.
/etc/wlm/current/shares	Contains the resource shares attributes for each class of the current workload management configuration.
/etc/wlm/current/description	Contains the class description text for each class of the current workload management configuration.

Related Information

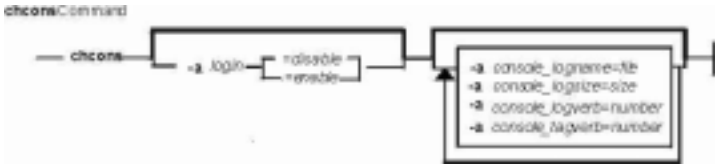
The **wlmcntrl**, **lsclass**, **mkclass**, and **rmclass** commands.

chcons Command

Purpose

Redirects the system console to a specified device or file to be effective on the next startup of the system.

Syntax



chcons [**−a login** { **=disable** | **=enable**}] [**−a console_logname=***file*] [**−a console_logsize=***size*]
[**−a console_logverb=***number*] [**−a console_tagverb=***number*] *PathName*

Description

The **chcons** command changes the system console effective on the next system startup. The current operation of the system console is not affected.

The *PathName* parameter must be a fully qualified path name to a device or file that is to become the system console.

If the *PathName* parameter specifies a file that does not exist, the **chcons** command creates the file at the next system startup. If the file does exist, the **chcons** command sends any console message output to the file. For a regular file, the system does not start the login program.

If the console path name is a character device, the system starts the login program on the device. Login is enabled on the console at all run levels. If no login is desired, use the **−a login=disable** flag.

> **CAUTION:** If the console is the only login terminal on the system, you cannot log in at the next start of the system using the **−a login=disable** flag.

Additional Information

The **chcons** command saves the specified information into the database to be used on the next start−up of the system with the console configuration method. This method checks the specified device path name to determine if it is a character special file. If it is not, or does not exist, the device path name is assumed to be a file, and the console is set accordingly. If the device path name is a character special file, the console configuration method uses the base name as a logical name and attempts to look up the device name in the device database. If the device is found and available, the console is set to the device.

If the device is not found or is found but not available, a console finder routine is run that displays a prompt requesting that a new system console device be selected. By default, the tty on the S1 port and all graphics displays will display the prompt. The **/etc/consdef** file must be modified to display the prompt on S2 or other ports.

For a device, an entry in the **inittab** file with the console identifier is set to the respawn action to allow a login on the console if the console login was specified as the **enable** parameter. This causes a login to be available at all run levels. If the console login was specified with the **disable** parameter or if a file is

designated as the console, the console entry in the **inittab** file is set to the OFF action, and login is disabled on the console for all run levels.

Flags

−alogin= [**disable** | **enable**]	Enables or disables the login on the console for all run levels at the next start−up of the system.
−a console_logname=*file*	Specifies the full path name to use for the console output log file.
−a console_logsize=*size*	Specifies the size, in bytes, of the console output log file.
−a console_logverb=*number*	Specifies the verbosity level for console output logging. Zero disables logging; 1 through 9 enable logging.
−a console_tagverb=*number*	Specifies the verbosity level for console output tagging. Zero disables tagging, 1 through 9 enable tagging.

Examples

1. To change the system console to a file called **console.out** in the **/tmp** directory, enter:

   ```
   chcons /tmp/console.out
   ```

2. To change the system console to a terminal with the `tty3` logical name, enter:

   ```
   chcons /dev/tty3
   ```

3. To change the system console to the terminal associated with the `/dev/tty3` device and ensure a login at the console, enter:

   ```
   chcons −a login=enable /dev/tty3
   ```

4. To change the system console to a terminal with the `tty0` logical name and disable login at the console, enter:

   ```
   chcons −a login=disable /dev/tty0
   ```

5. To change the console to the default physical LFT display, enter:

   ```
   chcons /dev/lft0
   ```

Files

/dev/console	Specifies the special file for system console access.
/etc/consdef	Enables non−default terminal to be selected as the console device.
/usr/sbin/chcons	Specifies the command file.

Related Information

The **init** command, **lscons** command, **swcons** command.

The **inittab** file, **consdef** file.

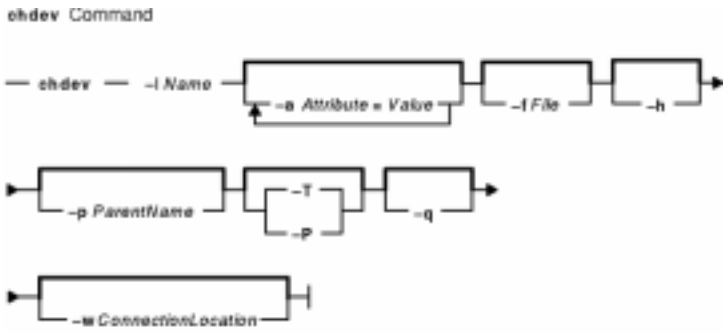
The **console** special file.

chdev Command

Purpose

Changes the characteristics of a device.

Syntax



chdev **−l***Name* [**−a***Attribute*=*Value* ...] [**−f** *File*] [**−h**] [**−p** *ParentName*] [**−P** | **−T**] [**−q**] [**−w***ConnectionLocation*]

Description

The **chdev** command changes the characteristics of the device specified with the given device logical name (the **−l** *Name* flag).The device can be in the Defined, Stopped, or Available state. Some changes may not be allowed when the device is in the Available state. When changing the device characteristics, you can supply the flags either on the command line or from a specified *File* parameter.

When neither the **−P** nor the **−T** flags are specified, the **chdev** command applies the changes to the device and updates the database to reflect the changes. If the **−P** flag is specified, only the database is updated to reflect the changes, and the device itself is left unchanged. This is useful in cases where a device cannot be changed because it is in use; in which case, the changes can be made to the database with the **−P** flag, and the changes will be applied to the device when the system is restarted. The **−T** flag is used to make a temporary change in the device without the change being reflected in the database. It is temporary in that the device will revert to the characteristics described in the database when the system is restarted. Not all devices support the **−P** and **−T** flags. A device that is in the Defined state can only have changes applied to the database.

> **Attention:** To protect the Configuration database, the **chdev** command is not interruptible. To stop this command before execution is complete could result in a corrupted database.

You can use the Web−based System Manager Devices application (**wsm devices** fast path) to change device characteristics. You could also use the System Management Interface Tool (SMIT) **smit chdev** fast path to run this command for certain devices.

Flags

−a *Attribute*=*Value*
: Specifies the device attribute value pairs used for changing specific attribute values. The *Attribute*=*Value* parameter can use one attribute value pair or multiple attribute value pairs for one **−a** flag. If you use an **−a** flag with multiple attribute value pairs, the list of pairs must be enclosed in quotes with spaces between the pairs. For example, entering `−a Attribute=Value` lists one attribute value pair per flag,

	while entering `-a 'Attribute1=Value1 Attribute2=Value2'` lists more than one attribute value pair.
-f *File*	Reads the needed flags from the named *File* parameter.
-h	Displays the command usage message.
-l *Name*	Specifies the device logical name, specified by the *Name* parameter, in the Customized Devices object class whose characteristics are to be changed.
-P	Changes the device's characteristics permanently in the Customized Devices object class without actually changing the device. This is useful for devices that cannot be made unavailable and cannot be changed while in the available state. The change can be made to the database with the **-P** flag. By restarting the system, the changes will be applied to the device. This flag cannot be used with the **-T** flag. Not all devices support the **-P** flag.
-p *ParentName*	Specifies the new device logical name of the parent device, specified by the *ParentName* parameter, in the Customized Devices object class. This flag is used only when changing the parent of the device. Not all devices support the **-p** flag.
-q	Suppresses the command output messages from standard output and standard error.
-T	Changes the characteristics of the device temporarily without changing the Customized Devices object class for the current start of the system. This flag cannot be used with the **-P** flag. Not all devices support the **-T** flag.
-w *ConnectionLocation*	Specifies the new connection location on the parent. This flag is used only when changing the connection location of the device. Not all devices support the **-w** flag.

Security

Access Control: Only the root user and members of the security group should have execute (x) access to this command.

Auditing Event	Information
DEV_Change	Parameters to the method the **cfgmgr** command calls.

Examples

1. To change the retention instructions of a 150M-byte, .25-inch tape drive `to rmt0` (so that the drive does not move the tape to the beginning, then to the end, and then back to the beginning each time a tape is inserted or the drive is powered on), enter:

   ```
   chdev -l rmt0 -a ret=no
   ```

   The system displays a message similar to the following:

   ```
   rmt0 changed
   ```

2. To change one or more attributes of the token-ring adapter `tok0` to preset values as described in the **changattr** file, enter:
   ```
   chdev -l tok0 -f changattr
   ```

   The system displays a message similar to the following:

   ```
   tok0 changed
   ```

3. To change the SCSI ID of the available SCSI adapter `scsi0` that cannot be made unavailable or changed due to available disk drives connected to it, enter:

```
chdev −l scsi0 −a id=6 −P
```

The system displays a message similar to the following:

```
scsi0 changed
```

To apply the change to the adapter, shutdown and restart the system.

4. To change the attribute describing which external connector is to be used by adapter `ent0`, enter:

   ```
   chdev −l ent0 −a bnc_select=dix
   ```

   The system displays a message similar to the following:

   ```
   ent0 changed
   ```

   If an error message is displayed because the device is in use, reissue the **chdev** command using the **−P** flag. Shutdown and restart the system to apply the change to the adapter.

5. To move a defined tty device `tty11` to port `0` on another serial adapter `sa5`, enter:

   ```
   chdev −l tty11 −p sa5 −w 0
   ```

   The system displays a message similar to the following:

   ```
   tty11 changed
   ```

   ```
   chdev −l sys0 −a maxuproc=100
   ```

Files

/usr/sbin/chdev Specifies the command file.

Related Information

The **lsattr** command, **lsconn** command, **lsdev** command, **lsparent** command, **mkdev** command, and **rmdev** command.

The Devices in *AIX Version 4.3 System Management Guide: Operating System and Devices* provides information about adding, changing, moving, and removing devices.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

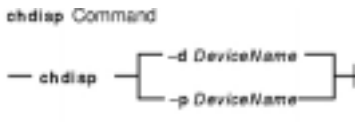
The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* tells you about the SMIT application.

chdisp Command

Purpose

The **chdisp** command changes the default display being used by the Low Function Terminal Subsystem.

Syntax



chdisp { **−d***DeviceName* | **−p***DeviceName* }

Description

The **chdisp** command changes the display used by the low function terminal (LFT) subsystem.

To generate a list of available displays and their respective display identifiers and descriptions, use the **lsdisp** command. For an example of the listing displayed, see the **lsdisp** command example listing.

> **Note:** The **chdisp** command can be used only on an LFT.

Alternatively, you can use the Web−based System Manager Devices application (**wsm devices** fast path) to change device characteristics. You could also use the System Management Interface Tool (SMIT) **smit chdisp** fast path to run this command for certain devices.

Flags

−d*DeviceName* Changes the default display currently being used by the LFT. This change is temporary resulting in the default display reverting back to the original display when the system is rebooted.

−p*DeviceName* Changes the default display to the specified display at the next reboot. This stays in effect until the user changes the default display again. The user must have superuser access to use this option.

Examples

1. To temporarily change the default display to a display with a device name `ppr0`, enter:

   ```
   chdisp −d ppr0
   ```

2. To permanently change the default display beginning with the next reboot to a display with the device name `gda1`, enter:

   ```
   chdisp −p gda1
   ```

Files

/bin/chdisp Contains the **chdisp** command.

Related Information

The **lsdisp** command.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

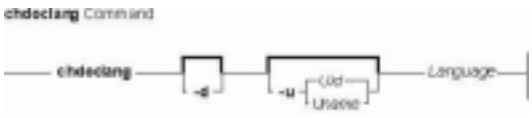
LFT Subsystem Component Structure Overview in *AIX Kernel Extensions and Device Support Programming Concepts*.

chdoclang Command

Purpose

Sets the default documentation language for the system or a user.

Syntax



chdoclang [**−d**] [**−u***UID* | *Uname*] *Language*

Description

The **chdoclang** command sets the default documentation language for the system or a user by adding an environment variable definition to either the **/etc/environment** or a user's **.profile** file. When the user opens the documentation library, the user's default documentation language will be used instead of the system wide setting.

This is the language in which the documentation library application will appear if it is launched using the **docsearch** command or the **Documentation Library** icon in the CDE desktop.

Flags

−d Removes a previous default documentation language setting.

−u Makes modification for specified user.

Examples

1. To change the default documentation language to English, enter:
   ```
   chdoclang en_US
   ```
2. To change the default documentation language to Japanese for the user fred, enter:
   ```
   chdoclang −u fred Ja_JP
   ```
3. To change the default documentation language to German for the user whose user id is 201, enter:
   ```
   chdoclang −u 201 de_DE
   ```

Files

/usr/bin/chdoclang	Change documentation language command
/etc/environment	Specifies basic environment for all processes
$HOME/.profile	Specifies environment for specific user needs

Related Information

The environment file, profile file format.

National Language Support Overview for System Management in the *AIX Version 4.3 System Management*

Concepts: Operating System and Devices.

Documentation Search Service in the *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

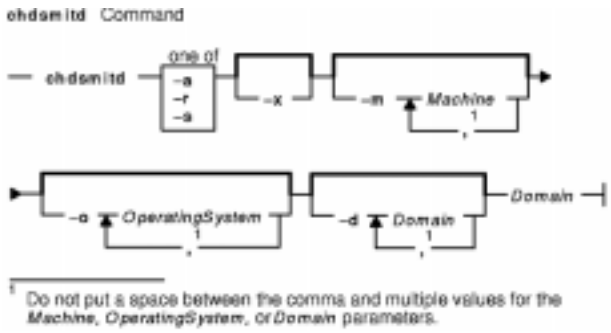
Documentation Search Service in the *AIX Version 4.3 General Programming Concepts: Writing and Debugging Programs*.

chdsmitd Command

Purpose

Changes a domain in the member list for the Distributed System Management Interface Tool (DSMIT).

Syntax



chdsmitd [**−a** | **−r** | **−s**] [**−x**] [**−m** *Machine* [**,***Machine*] ...] [**−d** *Domain* [**,***Domain*] ...] [**−o** *OperatingSystem* [**,***OperatingSystem*] ...] *Domain*

> **Note:** Do not put a space between the comma and multiple values for the *Machine*, *OperatingSystem*, or *Domain* parameters.

Description

The **chdsmitd** command changes a domain member list in DSMIT. The *Domain* parameter specifies the name of the domain to change. Domains must be homogeneous, consisting only of clients with operating systems of the same type. Use the **−d**, **−m**, and **−o** criteria flags and the **−x** intersect flag to add and delete domain members and to redefine the member list.

Flags

−?	Displays the usage statement.
−a	Adds a machine to a domain.
−d *Domain*	Specifies one or more domain names.
−m *Machine*	Specifies one or more machine names.
−o*OperatingSystem*	Specifies an operating system.
−r	Redefines a domain's member list.
−s	Deletes a machine from the domain.
−x	Uses the intersection of the subsets of machines defined by the **−d**, **−m**, and **−o** criteria flags.

Examples

1. To add a machine to a domain, enter:

   ```
   chdsmitd −a −m NewMachine Domain1
   ```

2. To delete a machine from a domain, enter:

```
chdsmitd −s −m OldMachine Domain1
```

Files

/usr/share/DSMIT/domains	Contains the list of domains used by DSMIT.
/usr/share/DSMIT/dsmitos	Contains the list of operating systems of DSMIT clients.
/usr/share/DSMIT/hosts	Contains the list of machines with DSMIT installed that can run commands built by the DSMIT server.

Related Information

The **mkdsmitd** command and **rmdsmitd** command.

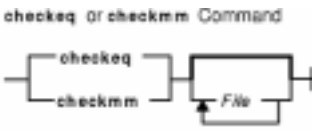
Distributed System Management Interface Tool (DSMIT) Overview in the *Distributed SMIT 2.2 for AIX: Guide and Reference*.

checkeq or checkmm Command

Purpose

Checks documents formatted with memorandum macros.

Syntax



{ **checkeq** | **checkmm** } [*File*...]

Description

The **checkeq** command is used to check for syntax errors in the specified files (*File*) that have been prepared for the **neqn** or **eqn** command. The **checkeq** command reports missing or unbalanced delimiters and the **.EQ** and **.EN** macro pair.

The **checkeq** command is functionally equivalent to the **checkmm** command.

The **checkmm** (check memorandum macros) command is used to check for syntax errors in files that have been prepared for the **mm** command or **mmt** command. For example, the **checkmm** command checks that you have a **.DE** (display end) macro corresponding to every **.DS** (display start) macro. *File* specifies files to be checked by the **checkeq** or **checkmm** command.

The output for the **checkmm** command is the number of lines checked and a list of macros that are unfinished because of missing macros.

Related Information

The **eqn** command, **mm** command, **mmt** command, **mvt** command, **neqn** command, **tbl** command.

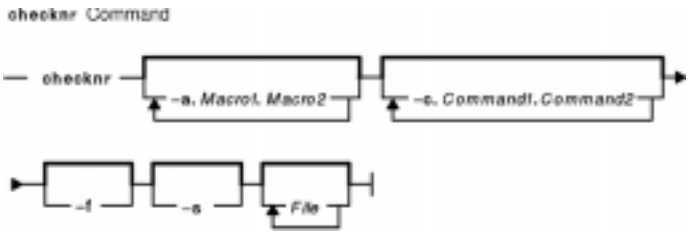
The **.DE** and **.DS** macros, **.EN** and **.EQ** macros, **mm** macro package.

checknr Command

Purpose

Checks **nroff** and **troff** files.

Syntax



checknr [**−a.***Macro1***.***Macro2* ...] [**−c.***Command1***.***Command2* ...] [**−f**] [**−s**] [*File* ...]

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, the **checknr** command checks standard input.

Delimiters checked are:

- Font changes using the **\f***Newfont* ... **\f**P.
- Size changes using the **\s***Newsize* ... **\s**0.
- Macros that come in open and close forms (such as the **.TS** and **.TE** macros) that must always come in pairs.

The **checknr** command can handle both the **ms** and **me** macro packages.

The **checknr** command is intended to be used on documents that are prepared with the **checknr** command in mind, much the same as the **lint** command. The **checknr** command requires a certain document writing style for the **\f** and **\s** commands, in that each **\f***Newfont* must be terminated with **\fP** and each **\s***Newsize* must be terminated with **\s0**. While it works to go directly into the next font or to explicitly specify the original font or point size, such a practice produces error messages from the **checknr** command.

File specifies **nroff** or **troff** input files for errors involving mismatched opening and closing delimiters and unknown commands. The default is standard input.

Flags

−a.*Macro1***.***Macro2*
: Adds pairs of macros to the list. This flag must be followed by groups of six characters, each group defining a pair of macros. The six characters are a period, *Macro1*, another period, and *Macro2*. For example, to define the pair, **.BS** and **.ES**, use **−a.BS.ES**.

  Note: There is no way to define a 1−character macro name using the **−a** flag.

−c.*Command1***.***Command2*

	Defines otherwise undefined commands that would get error messages from the **checknr** command.
−f	Causes the **checknr** command to ignore **\f** font changes.
−s	Causes the **checknr** command to ignore **\s** size changes.

Note: The **checknr** command does not correctly recognize certain reasonable constructs, such as conditionals.

Related Information

The **checkeq** command, **lint** command, **nroff** command, **troff** command.

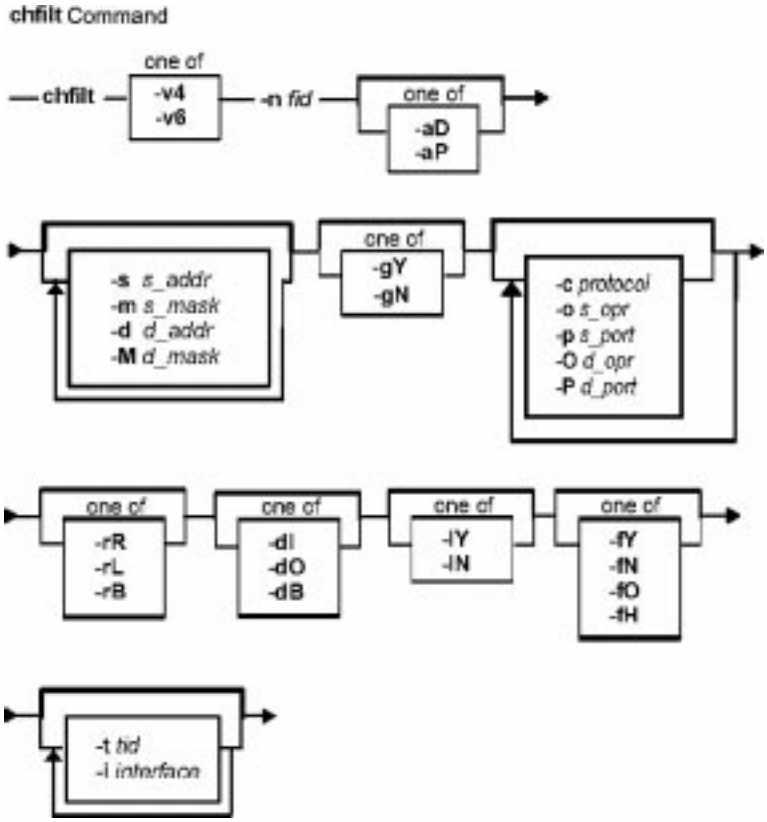
The **me** macro package, **ms** macro package.

chfilt Command

Purpose

Changes a filter rule.

Syntax



chfilt−**v4**|**6**−**n***fid* [−**aD**|**P**] [−**s***s_addr*] [−**m***s_mask*] [−**d***d_addr*] [−**M***d_mask*] [−**gY**|**N**] [−**c***protocol*] [−**o***s_opr*] [−**p***s_port*] [−**O***d_opr*] [−**P***d_port*] [−**rR**|**L**|**B**] [−**dI**|**O**|**B**] [−**lY**|**N**] [−**fY**|**N**|**O**|**H**] [−**t***tid*] [−**i***interface*]

Description

Use the **chfilt** command to change the definition of a filter rule in the filter rule table. Auto−generated filter rules and manual filter rules can be changed by this command. If an auto−generated filter rule is modified by **chfilt**, it will then become a manual filter rule.

Flags

−**a**	Action. The value of Deny (**D**) will block traffic, and the value of Permit (**P**) will allow traffic.
−**c***protocol*	Protocol. The valid values are: **udp**, **icmp**, **icmpv6**, **tcp**, **tcp/ack**, **ospf**, **ipip**, **esp**, **ah**, and **all**. Value **all** indicates that the filter rule will apply to all the protocols. The protocol can also be specified numerically (between 1 and 252).
−**d***d_addr*	Destination address. It can be an IP address or a host name. If a host name is specified, the first IP address returned by the name server for that host will be used. This value along with the destination subnet mask will be compared against the destination address of the IP packets.

−f Fragmentation control. This flag specifies that this rule will apply to either all packets (**Y**), fragment headers and unfragmented packets only (**H**), fragments and fragment headers only (**O**), or unfragmented packets only (**N**).

−g Apply to source routing? Must be specified as **Y** (yes) or **N** (No). If **Y** is specified, this filter rule can apply to IP packets that use source routing.

−i*interface* The name of IP interface(s) to which the filter rule applies. Examples are: **all**, **tr0**, **en0**, **lo0**, and **pp0**.

−l Log control. Must be specified as **Y** (yes) or **N** (No). If specified as **Y**, packets that match this filter rule will be included in the filter log.

−M*d_mask* Destination subnet mask. This will be applied to the Destination address(**−d** flag) when compared with the destination address of the IP packets.

−m*s_mask* Source subnet mask. This will be applied to the Source address (**−s** flag) when compared with the source address of the IP packet.

−n*fid* The ID of the filter rule you want to change. It must exist in the filter rule table and for IP version 4, it cannot be 1 (rule 1 is a system reserved rule and is unchangeable).

−O*d_opr* Destination port or ICMP code operation. This is the operation that will be used in the comparison between the destination port/ICMP code of the packet with the destination port or ICMP code (**−P** flag). The valid values are: **lt**, **le**, **g**t, **ge**, **eq**, **neq**, and **any**. This value must be **any** when the **−c** flag is **ospf**.

−o*s_opr* Source port or ICMP type operation. This is the operation that will be used in the comparison of the source port/ICMP type of the packet with the source port or ICMP type (**−p** flag) specified in this filter rule. The valid values are: **lt**, **le**, **gt**, **ge**, **eq**, **neq**, and **any**. The value must be **any** when the **−c** flag is **ospf**.

−P*d_port* Destination port/ICMP code. This is the value/code that will be compared to the destination port (or ICMP code) of the IP packet.

−p*s_port* Source port or ICMP type. This is the value/type that will be compared to the source port (or ICMP type) of the IP packet.

−r Routing. This specifies whether the rule will apply to forwarded packets (**R**), packets destined or originated from the local host (**L**), or both (**B**).

−s*s_addr* Source address. It can be an IP address or a host name. If a host name is specified, the first IP address returned by the name server for that host will be used. This value along with the source subnet mask will be compared against the source address of the IP packets.

−t*tid* ID of the tunnel related to this filter rule. All the packets that match this filter rule must go through the specified tunnel.

−v IP version of the target filter rule.

−w Direction. This specifies whether the rule will apply to incoming packets (**I**), outgoing packets (**O**), or both (**B**).

chfn Command

Purpose

Changes a user's gecos information.

Syntax

chfn [*Name*]

Description

The **chfn** command changes a user's gecos information. Gecos information is general information stored in the **/etc/passwd** file. This information is not used by the system. The type of information you store in this field is up to you. Some system administrators store information such as the user's full name, phone number, and office number.

The **chfn** command is interactive. After you enter the command, the system displays the current gecos information and prompts you to change it. To exit the **chfn** command without changing any information, press Enter.

You can use any printable characters in the gecos information string except a **:** (colon), which is an attribute delimiter.

By default, the **chfn** command changes the gecos information of the user who runs the command. You can also use this command to change the gecos information of other users. However, you must have execute permission for the **chuser** command to change the gecos information for another user.

Security

Access Control: All users should have execute (x) access to this command since the program enforces its own access policy. This command should be installed as a program in the trusted computing base (TCB). The command should be owned by the security group with the **setgid** (SGID) bit set.

Files Accessed:

Mode	File
x	**/usr/bin/chuser**
rw	**/etc/passwd**

Examples

1. If you are John Smith and want to change your gecos information, enter:

   ```
   chfn
   ```

   The current gecos string appears, followed by a prompt that asks if a change should be made:

```
current gecos:
     "John Smith;555-1746;room 74"
change (y/n)? >
```

To change the room number from `74` to `36,` enter `y` to request a change and enter the revised information when the `to? >` prompt appears:

```
current gecos:
     "John Smith;555-1746;room 74"
change (y/n)? > y
to? > John Smith;555-1746;room 36
```

2. If you are John Smith and want to view your gecos information but not change it, enter:

   ```
   chfn
   ```

   The current gecos string appears, followed by a prompt that asks if a change should be made:

   ```
   current gecos:
        "John Smith;555-1746;room 74"
   change (y/n)? >
   ```

   If you decide not to change the information, enter `n` after the `change (y/n)?` prompt or press the Enter key:

   ```
   current gecos:
        "John Smith;555-1746;room 74"
   change (y/n)? > n
   ```

   This is your opportunity to indicate that the information should remain unchanged. If you enter `y`, you are committed to enter an information string or use the Enter key to set the string to null. Note that the function of the Enter key differs before and after a `y` character is entered.

3. If you have execute (x) permission for the **chuser** command and want to change the gecos information for the `johns` user, enter:

   ```
   chfn johns
   ```

The current gecos string and prompts appear as in Example 1.

Files

/usr/bin/chfn	Specifies the path to the **chfn** command.
/usr/bin/chuser	Changes user information.
/etc/passwd	Contains basic user attributes.

Related Information

The **chgroup** command, **chgrpmem** command, **chuser** command, **lsgroup** command, **lsuser** command, **mkgroup** command, **mkuser** command, **passwd** command, **pwdadm** command, **rmgroup** command, **rmuser** command, **setgroups** command, **setsenv** command.

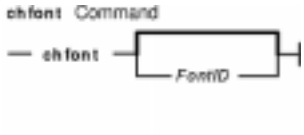
Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices* describes the identification and authentication of users, discretionary access control, the trusted computing base, and auditing.

chfont Command

Purpose

Changes the default font selected at boot time.

Syntax



chfont [*FontID*]

Description

The chfont command changes the font used by a display at system restart.

To see a list of available fonts with their respective font ids, font names, the glyph size and the font encoding, see the **lsfont** command. For an example of the listing displayed, see the **lsfont** command example listing.

You must have root authority to run this command.

> **Note:** This command can be used only on an LFT (Low Function Terminal).

You can use the Web−based System Manager Devices application (**wsm devices** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chfont** fast path to run this command.

Parameter

FontID The font id of the new font.

Examples

To change the font used by this display to the third font in the font palette, enter:

```
chfont 2
```

Files

/bin/chfont Contains the **chfont** command.
/usr/lpp/fonts Contains the font directory.

Related Information

The **lsfont** command, **mkfont** command.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide:*

Operating System and Devices.

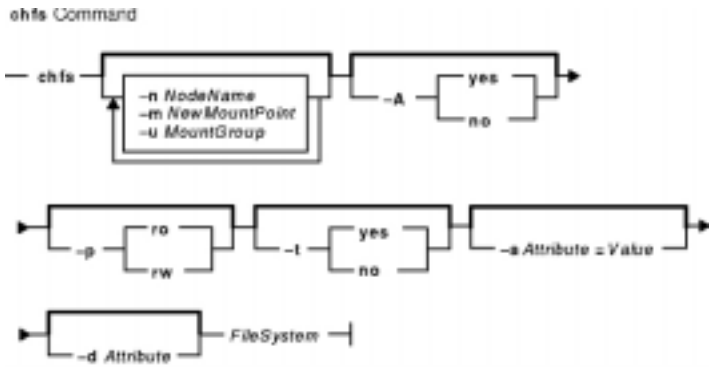
LFT Subsystem Component Structure Overview in *AIX Kernel Extensions and Device Support Programming Concepts*.

chfs Command

Purpose

Changes attributes of a file system.

Syntax



chfs [**−n***NodeName*] [**−m***NewMountPoint*] [**−u***MountGroup*] [**−A** { **yes** | **no** }] [**−p** { **ro** | **rw** }] [**−t** { **yes** | **no** }] [**−a***Attribute*=*Value*] [**−d***Attribute*] *FileSystem*

Description

The **chfs** command changes the attributes of a file system. The new mount point, automatic mounts, permissions, and file system size can be set or changed. The *FileSystem* parameter specifies the name of the file system, expressed as a mount point.

Some file system attributes are set at the time the file system is created and cannot be changed. For the Journaled File System (JFS), such attributes include the fragment size, block size, number of bytes per i−node, compression, and the minimum file system size.

You can use the Web−based System Manager File Systems application (**wsm fs** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chhfs** fast path to run this command.

Flags

−a *Attribute=Value*
: Specifies the *Attribute=Value* pairs; dependent on virtual file system type. To specify more than one *Attribute=Value* pair, provide multiple **−a** *Attribute=Value* parameters.

  The following attribute/value pairs are specific to the Journaled File System (JFS):

  −a *Size=NewSize*
  : Specifies the size of the Journaled File System in 512−byte blocks. If Value begins with a + it is interpreted as a rquest to increase the file system size by the specified amount. If the specified size is not evenly divisible by the physical partition size, it is rounded up to the closest number that is evenly divisible.

    The maximum size of a JFS file system is a function of its fragment size and the nbpi value.

These values yield the following size restrictions:

nbpi	Fragment size in bytes	Maximum size in 512−byte blocks
512	512, 1024, 2048, 4096	16777216
1024	512, 1024, 2048, 4096	33554432
2048	512, 1024, 2048, 4096	67108864
4096	512, 1024, 2048, 4096	134217728
8192	512, 1024, 2048, 4096	268435456
16384	512	268435456
16384	1024, 2048, 4096	536870912
32768	512	268435456
32768	1024	536870912
32768	2048, 4096	1073741824
65536, 131072	512	268435456
65536, 131072	1024	536870912
65536, 131072	2048	1073741824
65536, 131072	4096	2147483648
AIX 4.1 is limited to NBPI values from 512 to 16384. In AIX4.3, you can have NBPI values from 512 to 128K, with corresponding maximum file system sizes.		

The volume group in which the file system resides defines a maximum logical volume size and also limits the file system size.

−a *log=LVName*
Specifies the full path name of the filesystem logging logical volume name of the existing log to be used. The log device for this filesystem must reside on the same volume group as the filesystem

−a*splitcopy=NewMountPointName*
Splits off a mirrored copy of the file system and mounts it read−only at the new mount point. This provides a copy of the file system with consistent JFS meta−data that can be used for backup purposes. User data integrity is not guaranteed, so it is recommended that file system activity be minimal while this action is taking place.

−a*copy=Copy#*
Specifies which mirror copy to split off when used in conjunction with the splitcopy attribute. The default copy is the second copy. Valid values are 1, 2, or 3.

−A
Specifies the attributes for auto−mount.

yes
File system is automatically mounted at system restart.

no
File system is not mounted at system restart.

−d*Attribute*
Deletes the specified attribute from the **/etc/filesystems** file for the specified file system.

−m*NewMountPoint*

Specifies the new mount point.

−*n* *NodeName*
Specifies a node name for the specified file system. The node name attribute in the **/etc/filesystems** file is updated with the new name. The node name attribute is specific to certain remote virtual file system types, such as the NFS (Network File System) virtual file system type.

−*p*
Sets the permissions for the file system.

ro
Specifies read−only permissions.

rw
Specifies read−write permissions.

−*t*
Sets the accounting attribute for the specified file system:

yes
File system accounting is to be processed by the accounting subsystem.

no
File system accounting is not to be processed by the accounting subsystem; this is the default.

−*u* *MountGroup*
Specifies the mount group. Mount groups are used to group related mounts, so that they can be mounted as one instead of mounting each individually. For example, if several scratch file systems always need to be mounted together when performing certain tests, they can each be placed in the test mount group. They can then all be mounted with a single command, such as the **mount −t***test* command.

Examples

1. To change the file system size of the `/test` Journaled File System, enter:

   ```
   chfs −a size=24576 /test
   ```

   This command changes the size of the `/test` Journaled File System to 24576 512−byte blocks, or 12MB (provided it was previously no larger than this).

2. To increase the size of the `/test` Journaled File System, enter:

   ```
   chfs −a size=+8192 /test
   ```

   This command increases the size of the `/test` Journaled File System by 8192 512−byte blocks, or 4MB.

3. To change the mount point of a file system, enter:

   ```
   chfs −m /test2 /test
   ```

   This command changes the mount point of a file system from `/test` to `/test2`.

4. To delete the accounting attribute from a file system, enter:

   ```
   chfs −d account /home
   ```

   This command removes the accounting attribute from the `/home` file system. The accounting attribute is deleted from the `/home:` stanza of the **/etc/filesystems** file.

5. To split off a copy of a mirrored file system and mount it read−only for use as an online backup, enter:
   ```
   chfs −a splitcopy=/backup −a copy=2 /testfs
   ```

This mount a read−only copy of /testfs at /backup.

File

/etc/filesystems Lists the known file systems and defines their characteristics.

Related Information

The **crfs** command, **mkfs** command, **mklv** command.

The Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the file system accounting subsystem.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains file system types, management, structure, and maintenance.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains SMIT structure, main menus, and tasks.

chgif Method

Purpose

Reconfigures an instance of a network interface.

Syntax

chgif [**−d**] **−l** *InterfaceInstance***−a** "*Attribute*=*Value* ..."

Description

The **chgif** method first modifies the database and then reconfigures the specified network interface instance (*InterfaceInstance*) by issuing a call to the **ifconfig** command. Only one interface can be changed per command invocation, and at least one attribute must be specified. This method is not normally used on the command line. Rather, it is called by high−level commands.

> **Note:** The **chgif** method is a programming tool and should not be executed from the command line.

Flags

−a"*Attribute*=*Value* ..."

Specifies pairs of attributes and values that configure the Interface instance. The *AttributeValue* pairs must be surrounded by quotes.

Valid attribute values are as follows:

netaddr
Specifies the Internet address of the network interface.

state (up/down)
Marks the interface as up or down.

trailers (on/off)
Turns the trailer link−level encapsulation on or off.

arp (on/off)
Enables or disables the use of the Address Resolution Protocol.

allcast (on/off)
Specifies whether to broadcast packets to all token−ring networks or just the local token−ring network. This attribute applies only to token−ring networks.

hwloop (on/off)
Enables or disables hardware loopback mode.

netmask
Specifies the network mask in dotted−decimal format.

security*SecurityLevelKeyword*
(**inet** only) Specifies the security level associated with the

interface. The value of the *SecurityLevelKeyword* variable can be one of the following:

- **none**
- **unclassified**
- **confidential**
- **secret**
- **top_secret**

When the level of security is defined as **none** or **unclassified**, no IP Option header is added to the IP header.

authority*AuthorityLevelKeyword*
(**inet** only) Specifies the security authority level associated with the interface. The value of the *AuthorityLevelKeyword* variable can be one or more of the following:

genser
Defense Communications Agency

siop
Department of Defense Organization of the Joint Chiefs of Staff

dsccs−spintcom
Defense Intelligence Agency

dsccs−criticom
National Security Agency

When more than one level of authority is specified, the values are separated by commas without embedded spaces.

mtu
Maximum IP packet size for this system.

broadcast
Specifies the address to use for representing broadcasts to networks.

dest
Specifies the destination address on a point−to−point link.

−d Specifies that changes are made only in the configuration database. Changes take effect at the next system restart.

−l*InterfaceInstance* Specifies the instance of the network interface to be reconfigured.

Related Information

The **chdev** command, **ifconfig** command.

The **chginet** method.

The **odm_run_method** subroutine.

TCP/IP Protocols, TCP/IP Addressing, TCP/IP Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Object Data Manager (ODM) Overview for Programmers in *AIX General Programming Concepts: Writing and Debugging Programs*.

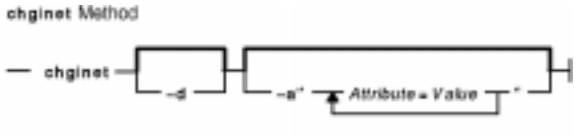
Writing a Device Method in *AIX Kernel Extensions and Device Support Programming Concepts*.

chginet Method

Purpose

Reconfigures the Internet instance.

Syntax



chginet [**−d**] [**−a**"*Attribute*=*Value* ..."]

Description

The **chginet** method reconfigures the Internet instance, and can also change the *HostName* variable and any static routes that are defined. The **chginet** method calls the **hostname** command to change the host name. The **chginet** method also calls the **route** command to change any static routes. The **chdev** command calls method.

> **Note:** The **chginet** method is a programming tool and should not be entered from the command line.

Flags

−a"*Attribute=Value ...*"
: Specifies the customized attributes of the Internet instance. The following are valid attributes:

hostname
: Specifies the name of the host.

gateway
: Specifies the default gateway.

route
: Specifies the route. The format of the *Value* variable of the *Route* attribute is: *route=destination, gateway,* [*metric*].

delroute
: Specifies the route to delete. The format of the value is: *route=destination, gateway, [metric]*.

−d
: Specifies that changes are made only in the configuration database. Changes take effect with the next IPL.

Examples

To change an Internet instance and specify a route, enter a method in the following format:

```
chginet −a"route=192.9.200.0,bcroom"
```

This example specifies a new route. The new route is being set to network `192.9.200.0`, the `bcroom` gateway.

Related Information

The **chdev** command, **hostname** command, **mkdev** command, **route** command.

The **odm_run_method** subroutine.

TCP/IP Network Interfaces in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Object Data Manager (ODM) Overview for Programmers in *AIX General Programming Concepts: Writing and Debugging Programs*.

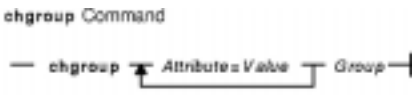
Writing a Device Method in *AIX Kernel Extensions and Device Support Programming Concepts*.

chgroup Command

Purpose

Changes attributes for groups.

Syntax



chgroup*Attribute=Value ... Group*

Description

> **Attention:** Do not use the **chgroup** command if you have a Network Information Service (NIS) database installed on your system, as this could cause serious system database inconsistencies.

The **chgroup** command changes attributes for the group specified by the *Group* parameter. The group name must already exist as a string of 8 bytes or less. To change an attribute, specify the attribute name and the value you want to change it to in the *Attribute=Value* parameter.

You can use the Web−based System Manager Users application (**wsm users** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chgroup** fast path to run this command.

Restrictions on Changing Groups

To ensure the security of group information, there are restrictions on using the **chgroup** command. Only the root user or users with UserAdmin authorization can use the **chgroup** command to change any group. These changes include:

- Make a group an administrative group by setting the **admin** attribute to true.
- Change any attributes of an administrative group.
- Add users to an administrative group's administrators list.

An administrative group is a group with the **admin** attribute set to true. Members of the **security** group can change the attributes of nonadministrative groups including adding users to the list of administrators.

The Attributes

You change attributes by specifying an *Attribute=Value* parameter. If you have the proper authority you can set the following group attributes:

adms Defines the users who can perform administrative tasks for the group, such as setting the members and administrators of the group. This attribute is ignored if **admin=true**, since only the root user can alter a group defined as administrative. The *Value* parameter is a list of comma−separated user login names. If you do not specify a *Value* parameter, all the administrators are removed.

admin Defines the administrative status of the group. Possible values are:

true Defines the group as administrative. Only the root user can change the attributes of groups defined as administrative.

false Defines a standard group. The attributes of these groups can be changed by the root user or a member of the security group. This is the default value.

id The group ID. The *Value* parameter is a unique integer string. Changing this attribute compromises system security and, for this reason, you should not change this attribute.

users A list of one or more users in the form: *User1*,*User2*,...,*Usern*. Separate group member names with commas. Each user must be defined in the database configuration files. You cannot remove users from their primary group.

The **adms** and **admin** attributes are set in the **/etc/security/group** file. The remaining attributes are set in the **/etc/group** file. If any of the attributes you specify with the **chgroup** command are invalid, the command makes no changes at all.

Security

Access Control: This command should grant execute (x) access only to the root user and the security group. This command should be installed as a program in the trusted computing base (TCB). The command should be owned by the root user with the **setuid** (SUID) bit set.

Files Accessed:

Mode	File
rw	**/etc/group**
rw	**/etc/security/group**
r	**/etc/passwd**

Auditing Events:

Event	Information
GROUP_Change	group, attributes

Examples

1. To add `sam` and `carol` to the `finance` group, which currently only has `frank` as a member, enter:

   ```
   chgroup users=sam,carol,frank  finance
   ```

2. To remove `frank` from the `finance` group, but retain `sam` and `carol`, and to remove the administrators of the `finance` group, enter:

   ```
   chgroup users=sam,carol adms= finance
   ```

In this example, two attribute values were changed. The name `frank` was omitted from the list of members, and the value for the `adms` attribute was left blank.

Files

/usr/bin/chgroup	Specifies the path to the **chgroup** command.
/etc/group	Contains the basic attributes of groups.
/etc/security/group	Contains the extended attributes of groups.
/etc/passwd	Contains the basic attributes of users.

Related Information

The **chfn** command, **chgrpmem** command, **chsh** command, **chuser** command, **lsgroup** command, **lsuser** command, **mkgroup** command, **mkuser** command, **passwd** command, **pwdadm** command, **rmgroup** command, **rmuser** command, **setgroups** command, **setsenv** command.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

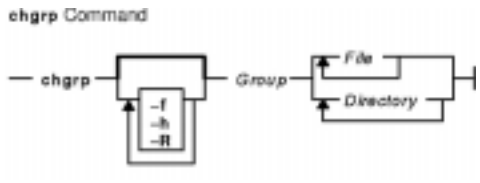
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

chgrp Command

Purpose

Changes the group ownership of a file or directory.

Syntax



chgrp [**−f**] [**−h**] [**−R**] *Group* { *File ...* | *Directory ...* }

Description

The **chgrp** command changes the group associated with the specified file or directory to the specified group name or group ID number. When a symbolic link is encountered and you have not specified the **−h** flag, the **chgrp** command changes the group ownership of the file or directory pointed to by the link and not the group ownership of the link itself.

If you specify the **−h** flag, the **chgrp** command has the opposite effect and changes the group ownership of the link itself and not that of the file or directory pointed to by the link.

If you specify both the **−h** flag and the **−R** flag, the **chgrp** command descends the specified directories recursively, and when a symbolic link is encountered, the group ownership of the link itself is changed and not that of the file or directory pointed to by the link.

Flags

−f Suppresses all error messages except usage messages.

−h Changes the group ownership of an encountered symbolic link and not that of the file or directory pointed to by the symbolic link.

−R Descends directories recursively, setting the specified group ID for each file. When a symbolic link is encountered and the link points to a directory, the group ownership of that directory is changed but the directory is not further traversed.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. 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.

2. To change the group ownership of the directory named `proposals,` and of all the files and subdirectories under it, to `staff`:

```
chgrp –R staff proposals
```

The group access permissions for `proposals` and for all the files and subdirectories under it now apply to the `staff` group.

Files

/usr/bin/chgrp The **chgrp** command
/etc/group File that identifies all known groups

Related Information

The **chown** command, **groups** command.

The **chown** subroutine, **fchown** subroutine.

Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices* describes system security.

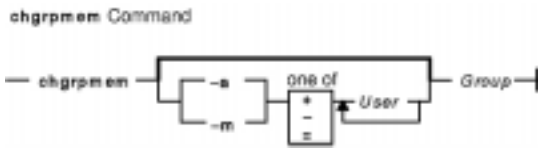
File Ownership and User Groups in *AIX Version 4.3 System User's Guide: Operating System and Devices* introduces file ownership and permissions to access files and directories.

chgrpmem Command

Purpose

Changes the administrators or members of a group.

Syntax



chgrpmem [{ { **−a** | **−m** } { + | − | = } *User* ... }] *Group*

Description

The **chgrpmem** command changes the administrators or members of the group specified by the *Group* parameter. Use this command to add, delete, or set a group's members or administrators list. You cannot remove users from their primary group. A user's primary group is maintained in the **/etc/passwd** file. If you specify only a group with the **chgrpmem** command, the command lists the group's members and administrators.

To add, delete, or set a user as a group administrator, specify the **−a** flag. Otherwise, to add, delete, or set a user as a group member, specify the **−m** flag. You must specify one of these flags and an operator to change a user's group membership. The operators do the following:

+ Adds the specified user.
− Deletes the specified user.
= Sets the list of administrators or members to the specified user.

You can specify more than one *User* parameter at a time. To do this, specify a comma−separated list of user names.

See the **chgroup** command for a list of restrictions that apply to changing group information.

Flags

−a Changes a group's administrators list.
−m Changes the group's members list.

Security

Access Control: All users should have execute (x) access to this command since the command itself enforces the access rights. This command should be installed as a program in the trusted computing base (TCB). The command should be owned by the **security** group with the **setgid** (SGID) bit set.

Files Accessed:

Mode File

x	**/usr/bin/chgroup**
r	**/etc/passwd**
r	**/etc/group**
rw	**/etc/security/group**

Examples

1. To remove `jones` as an administrator of the `f612` group, enter:

   ```
   chgrpmem –a – jones f612
   ```

2. To add members `davis` and `edwards` to group `f612`, enter:

   ```
   chgrpmem –m + davis,edwards f612
   ```

3. To list members and administrators of group `staff`, enter:

   ```
   chgrpmem staff
   ```

Files

/usr/bin/chgrpmem	Specifies the path to the **chgrpmem** command.
/etc/passwd	Contains the basic attributes of users.
/etc/group	Contains the basic attributes of groups.
/etc/security/group	Contains the extended attributes of groups.

Related Information

The **chfn** command, **chgroup** command, **chsh** command, **chuser** command, **lsgroup** command, **lsuser** command, **mkgroup** command, **mkuser** command, **passwd** command, **pwdadm** command, **rmgroup** command, **rmuser** command, **setgroups** command, **setsenv** command.

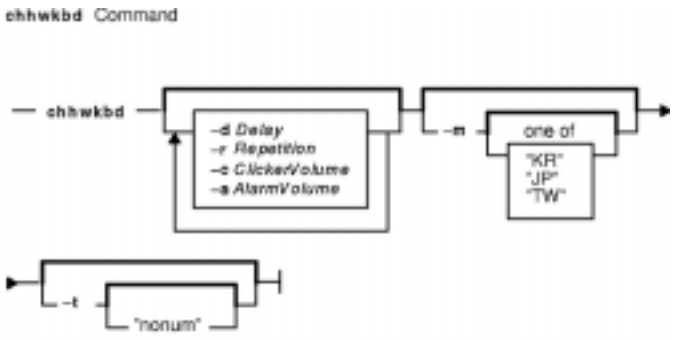
For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

chhwkbd Command

Purpose

Changes keyboard attributes stored in the Object Data Manager (ODM) database.

Syntax



chhwkbd [**−d***Delay*] [**−r** *Repetition*] [**−c** *ClickerVolume*] [**−a** *AlarmVolume*] [**−m** ["KR" | "JP" | "TW"]] [**−t** ["nonum"]]

Description

The **chhwkbd** command changes the following keyboard attributes stored in the ODM database:

- Repetition delay
- Repetition rate
- Clicker volume
- Alarm volume
- Korean, Japanese, and Chinese keyboard identification
- Numeric pad emulation enable/disable

Changes to the keyboard attributes take effect after system restart.

You can use the Web−based System Manager Devices application (**wsm devices** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chgkbd** fast path to run this command.

Flags

−a*AlarmVolume* Sets the alarm volume to the specified value. Values for the *AlarmVolume* variable are defined below:

0
off

1
low

2
medium

3
high

−c*ClickerVolume* Sets the clicker volume to the specified value. Values for the

ClickerVolume variable are defined below:

0
: off

1
: low

2
: medium

3
: high

−d*Delay*
: Sets the keyboard repetition delay to the specified value. The *Delay* variable can be 250, 500, 750, or 1000 msec. The default value is 500 msec.

−m ["KR" | "JP" | "TW"]
: Provides extended keyboard identification for the following keyboards:

"KR"
: Korean keyboard

"JP"
: Japanese keyboard

"TW"
: Chinese keyboard

Use the **−m** flag without specifying a value to remove extended keyboard identification.

> **Note:** This flag is valid only when an IBM RS/6000 106−key keyboard or an IBM PS/2 keyboard or equivalent keyboard is attached to the workstation.

The **−m** flag is set automatically when the locale is selected using SMIT.

−r*Repetition*
: Sets the rate of repetition to the specified value. The *Repetition* variable can be an integer from 2 to 30 inclusive. The default value is 11 characters per second.

−t ["nonum"]
: Enables or disables numeric pad emulation. To enable numeric pad emultaion, specify the "nonum" parameter. Use the **−t** flag without specifying a value to disable numeric pad emulation.

Notes:

1. This flag is valid only when an IBM PS/2 keyboard or equivalent keyboard is attached to the workstation.
2. "nonum" means no numeric keypad.

Examples

1. To change the keyboard repetition delay rate to 250 msec, enter:

   ```
   chhwkbd −d 250
   ```

2. To change the keyboard repetition rate to 30 characters per second, enter:

   ```
   chhwkbd −r 30
   ```

File

/usr/bin/chhwkbd Contains the **chhwkbd** command.

Related Information

Low Function Terminal (LFT) Subsystem Overview in *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*.

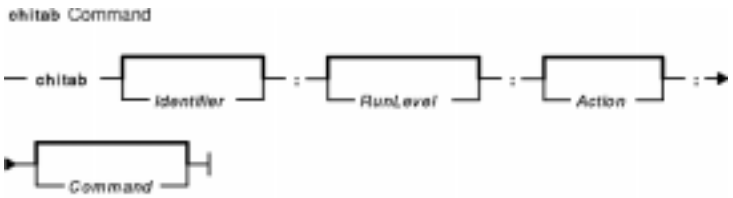
Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

chitab Command

Purpose

Changes records in the **/etc/inittab** file.

Syntax



chitab { [*Identifier*] **:** [*RunLevel*] **:** [*Action*] **:** [*Command*] }

Description

The **chitab** command changes a record in the **/etc/inittab** file. The *Identifier***:***Run Level***:***Action***:***Command* parameter string is the new entry to the **/etc/inittab** file. You can search for a specific record by using fields in the *Identifier* portion of the parameter string. The command finds the specified *Identifier* and changes that record.

Note: The **chitab** command can not comment out an entry in the **/etc/inittab** file.

Parameters

The *Identifier***:***Run Level***:***Action***:***Command* parameter string specifies a record in the **/etc/inittab** file where the following parameters apply:

Action A 20−character parameter that informs the **init** command how to process the *Command* parameter you specify. The **init** command recognizes the following actions:

boot
Read this record only when the system boots and reads the **/etc/inittab** file. The **init** command starts the process. Do not wait for the process to stop, and when it does stop, do not restart the process. The run level for this process should be the default, or it must match the run level specified by the **init** command at startup time.

bootwait
Read this record only when the system boots and reads the **/etc/inittab** file. The **init** command starts the process. Wait for it to stop, and when it does stop, do not restart the process.

hold
When the process identified in this record is terminated, do not start a new one. The **hold** action can only be activated by the **phold** command.

initdefault
Start the process identified in this record only when the **init** command is originally invoked. The **init** command uses this line to determine which run level to originally enter. It does this by taking the highest run level specified in the *RunLevel* field and using that as its initial state. If the *RunLevel* parameter is empty, this is interpreted as 0123456789, and the **init** command enters a run level of **9**. If the **init** command does not find an **initdefault** line in the **/etc/inittab** file, it requests an initial run level from the operator

at initial program load (IPL) time.

off
If the process identified in this record is currently running, send the warning signal **SIGTERM** and wait 20 seconds before sending the **SIGKILL** kill signal. If the process is nonexistent, ignore this line.

once
When the **init** command enters the run level specified for this record, start the process, do not wait for it to stop, and when it does stop, do not restart the process. If the system enters a new run level while the process is running, the process is not restarted.

ondemand
Functionally identical to **respawn**. If the process identified in this record does not exist, start the process. If the process currently exists, do nothing and continue scanning the **/etc/inittab** file. Specify this action to perform the **respawn** action when using **a**, **b**, or **c** run levels.

powerfail
Start the process identified in this record only when the **init** command receives a **SIGPWR** power fail signal.

powerwait
Start the process identified in this record only when the **init** command receives a **SIGPWR** power fail signal, and wait until it stops before continuing to process the **/etc/inittab** file.

respawn
If the process identified in this record does not exist, start the process. If the process currently exists, do nothing and continue scanning the **/etc/inittab** file.

sysinit
Start the process identified in this record before the **init** command tries to access the console. For example, you might use this to initialize devices.

wait
When the **init** command enters the run level specified for this record, start the process and wait for it to stop. While the **init** command is in the same run level, all subsequent reads of the **/etc/inittab** file ignore this object. If you are operating in a diskless environment, specifying the **wait** action causes your system to boot more quickly.

Command A 1024−character field specifying the shell command.

Identifier A 14−character parameter that uniquely identifies an object. The *Identifier* must be unique. If the *Identifier* is not unique, the command is unsuccessful. The *Identifier* cannot be changed; if you try to change it, the command is unsuccessful.

RunLevel A 20−character parameter defining the run levels in which the *Identifier* can be processed. Each process started by the **init** command can be assigned one or more run levels in which it can be started.

Examples

To change the run level of a record for `tty2`, enter:

```
"chitab tty002:23:respawn:/usr/sbin/getty /dev/tty"
```

The quotes are required when the record being added has spaces or tabs.

Files

/etc/inittab Controls which processes the **init** command starts.

Related Information

The **init** command, **lsitab** command, **mkitab** command, **rmitab** command.

chkbd Command

Purpose

Changes the software keyboard map to be loaded into the system at the next IPL (Initial Program Load).

Syntax

chkbd *KeyMapPathName*

Description

The **chkbd** command changes the default software keyboard map loaded at system IPL. The *KeyMapPathname* parameter provides the location of the software keymap file. This pathname can be absolute or simply the filename. If only the filename is specified then the command will look for it in the default directory **/usr/lib/nls/loc**.

> **Note:** This command can be used only on an LFT display.

For a list of all available keyboard maps, use the **lskbd** command.

You can use the Web−based System Manager Devices application (**wsm devices** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chkbd** fast path to run this command.

Parameter

KeyMapPathName Provides the location of the software keymap file.

Files

/bin/chkbd Contains the **chkbd** command.
/usr/lib/nls/loc Contains the keyboard directory.

Related Information

AIX Version 4 Keyboard Technical Reference

Low Function Terminal (LFT) Subsystem Overview in *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*.

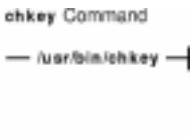
Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

chkey Command

Purpose

Changes your encrypting key.

Syntax



/usr/bin/chkey

Description

The **chkey** command prompts you for a password and uses it to encrypt a new encryption key. Once the key is encrypted, the **ypupdated** daemon updates the **/etc/publickey** file.

Related Information

The **keylogin** command, **newkey** command.

The **keyserv** daemon, **ypupdated** daemon.

The **/etc/publickey** file.

How to Export a File System Using Secure NFS, How to Mount a File System Using Secure NFS in *AIX Version 4.3 System Management Guide: Communications and Networks.*

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Network Information Service (NIS) in *AIX Version 4.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

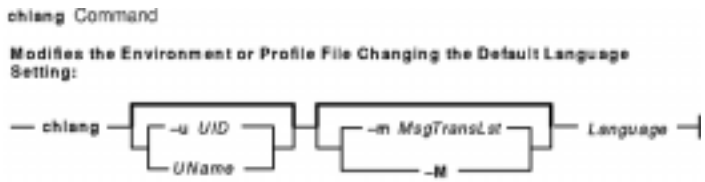
chlang Command

Purpose

Changes the language settings for system or user.

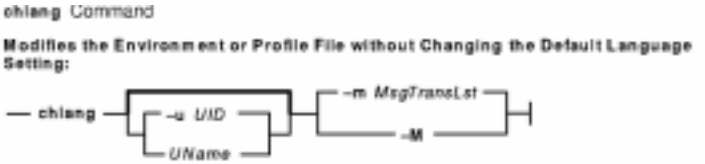
Syntax

To Modify the Environment or Profile File Changing the Default Language Setting:



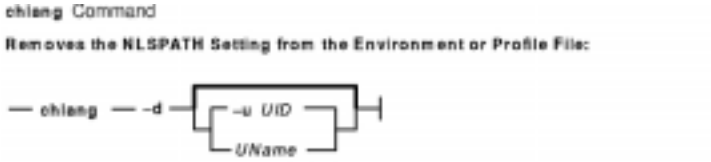
chlang [**−u** *UID* | *Uname*] [**−m** *MsgTransLst* | **−M**] *Language*

To Modifiy the Environment or Profile File without Changing the Default Language Setting:



chlang [**−u** *UID* | *Uname*] **−m** *MsgTransLst* | **−M**

To Remove the NLSPATH Setting from the Environment or Profile File:



chlang −d [**−u** *UID* | *UName*]

Description

The **chlang** command is a high−level shell command that changes the language settings for either the entire system or an individual user. If the effective id of the invoker is root and the **−u** option was not used, the language settings will be changed for the entire system in the **/etc/environment** file. If the effective id of the invoker is not root, or if the **−u** option was used, the language settings will be changed for an individual user in the user's **.profile** file.

When **chlang** is run with a language and no options, the **LANG** environment variable will be set to the language specified.

When chlang is run with the **−m** option, the **LANG** and **NLSPATH** environment variables will be set. In addition, the **LC_MESSAGES** variable will be set to the first value specified in the *MsgTransLst* of the **−m** flag if it is different from the **Language** parameter and the **Language** parameter has a system supplied translation available.

When **chlang** is run with the **−d** option, the **NLSPATH** environment variable will be removed.

Notes:

1. Changes made to the NLS environment by **chlang** are not immediate when either **/etc/environment** or the user's **.profile** are modified. Changes to **/etc/environment** requires rebooting the sytsem. Changes to a user's **.profile** requires logging in again or running the **.profile** file.
2. When modifying a user's configuration file, if the user uses the C shell (**/usr/bin/csh**) their **.cshrc** file will be modified rather than the **.profile** file.

Flags

−d	Used to remove the **NLSPATH** environment variable. This option will remove **NLSPATH** from either **/etc/environment** or the user's **.profile**. If **NLSPATH** was not currently in the file being modified, a warning message will be displayed.
−m *MsgTransLst*	Used to make modifications to the **NSLPATH** environment variable. *MsgTransLst* is a colon−separated list of message translations (locale names) that indicates the message translation hierarchy required for the system or user. If the first language in the list is different from the **Language** parameter and **Language** parameter has system supplied translation, then the **LC_MESSAGES** environment variable will be set to that first value. If the first language−territory in the list is the same as the language being set, the **LC_MESSAGES** environment variable will be removed. All entries in the list become hard coded directories in the **NLSPATH** environment.
−M	Used to reset the **LC_MESSAGES** environment variable and set the **NLSPATH** environment variable to the default translation hierarchy, which is: **`/usr/lib/nls/msg/%L/%N:`** **`/usr/lib/nls/msg/%L/%N.cat:`**
−u*UID* or *UName*	Used to make modification to an individual user. The user can be specified by either user id number or user login name. If the effective id of **chlang** is root, the **−u** parameter must be used to change the language environment for any specific user ID, including root itself (no **−u** parameter in this case will update the **/etc/environment** file rather than root's **.profile**). If the effective id is not root, the **−u** parameter is not needed. If it is specified, it must be equal to the effective id of the invoker.
Language	This is the language−territory (locale name) that will become the locale setting for the LANG environment variable.

Exit Status

0 Indicates successful completion.

>0 Indicates an error occurred.

Examples

1. Assume the preferred locale is Norwegian, and the language translations in order of preference are Norwegian, Swedish, and English. The command to achieve this for user *amcleod* is as follows:

   ```
   chlang −u amcleod −m no_NO:sv_SE:en_US no_NO
   ```

   The following settings would be made in the **.profile** for user *amcleod*. Because the first language in the message translation list is Norwegian, as is the **Language** parameter, **LC_MESSAGES** would not be set by **chlang**. If **LC_MESSAGES** had been set, it would be removed:

   ```
   LANG=no_NO

   NLSPATH=/usr/lib/nls/msg/%L/%N:
           /usr/lib/nls/msg/no_NO/%N:
           /usr/lib/nls/msg/sv_SE/%N:
           /usr/lib/nls/msg/en_US/%N:
           /usr/lib/nls/msg/%L/%N.cat:
           /usr/lib/nls/msg/no_NO/%N.cat:
           /usr/lib/nls/msg/sv_SE/%N.cat:
           /usr/lib/nls/msg/en_US/%N.cat
   ```

2. Assume the preferred locale is French, and the language translations in order of preference are French Canadian and English. To achieve this for a non−root user enter:

   ```
   chlang −m fr_CA:en_US fr_FR
   ```

   The following settings would be made in the **.profile** file for the user invoking **chlang**. Because the first language in the message translation list is different from the cultural convention (locale), **LC_MESSAGES** is set by **chlang**.

   ```
   LANG=fr_FR

   LC_MESSAGES=fr_CA

   NLSPATH=/usr/lib/nls/msg/%L/%N:
           /usr/lib/nls/msg/fr_CA/%N:
           /usr/lib/nls/msg/en_US/%N:
           /usr/lib/nls/msg/%L/%N.cat:
           /usr/lib/nls/msg/fr_CA/%N.cat:
           /usr/lib/nls/msg/en_US/%N.cat
   ```

3. Assume that a system administrator (root authority) in Spain is configuring a system from another country, and needs to change the default language environment so the machine operates properly in its new location. To change the default in the **/etc/environment** file, enter:

   ```
   chlang −m es_ES es_ES
   ```

   The following settings would be made in the **/etc/environment** file.

   ```
   LANG=es_ES

   NLSPATH=/usr/lib/nls/msg/%L/%N:
           /usr/lib/nls/msg/es_ES/%N:
           /usr/lib/nls/msg/%L/%N.cat:
           /usr/lib/nls/msg/es_ES/%N.cat
   ```

Files

/usr/bin/chlang	Change language command
/etc/environment	Specifies basic environment for all processes
$HOME/.profile	Specifies environment for specific user needs

Related Information

The **environment** file, **profile** file format.

National Language Support Overview for System Management in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

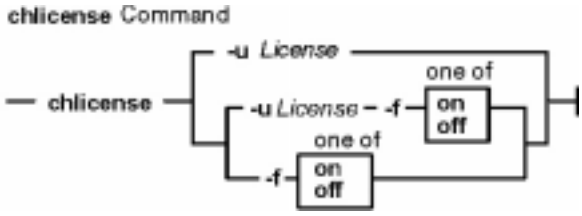
Understanding Locale Environment Variables in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

chlicense Command

Purpose

Changes the number of fixed licenses and the status of the floating licensing of the system.

Syntax



chlicense [**−u***License*] [**−f** {**on** | **off**}]

> **Note:** At least one flag must be specfied with the **chlicense** command.

Description

There are two types of user licensing: fixed and floating. Fixed licensing is always enabled and the number of licenses can be changed using **−u** flag of the **chlicense** command. Floating licensing is enabled or disabled using the **−f** flag.

Flags

> **Note:** At least one flag must be specfied with the **chlicense** command.

−f Changes the status of the floating licensing of the system. The status must be either **on** or **off**. The status of **on** enables the floating licensing and **off** disalbes the floating licensing. The **−f** flag is optional.

−u *License* Changes the number of fixed licenses on a system. The value of *License* must be a number greater than 0. The **−u** flag is optional.

Examples

1. To enable the floating licensing for the system, enter:

   ```
   chlicense −f on
   ```

2. To disable the floating licensing for the system, enter:

   ```
   chlicense −f off
   ```

3. To change the number of fixed linces to 125 and to enable floating licensing on the system, enter:

   ```
   chlicense −u 125 −f on
   ```

Related Information

The **lslicense** and **monitord** daemon.

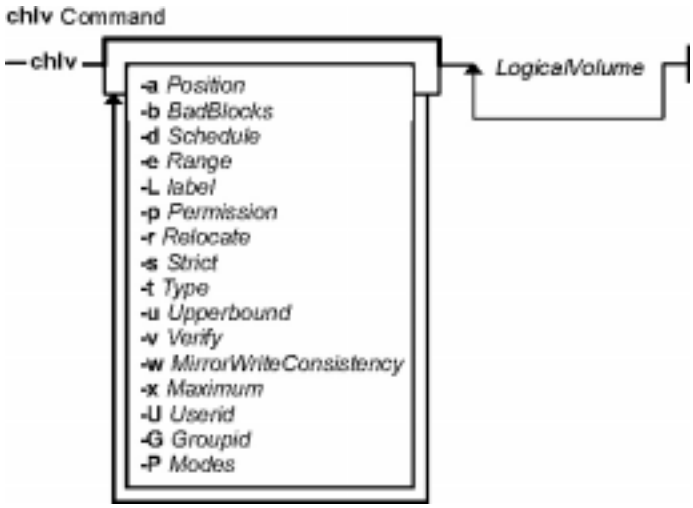
chlv Command

Purpose

Changes only the characteristics of a logical volume.

Syntax

To Change the Characteristics of a Logical Volume



chlv [**−a** *Position*] [**−b** *BadBlocks*] [**−d** *Schedule*] [**−e** *Range*] [**−L** *label*] [**−p** *Permission*] [**−r** *Relocate*] [**−s** *Strict*] [**−t** *Type*] [**−u** *Upperbound*] [**−v** *Verify*] [**−w** *MirrorWriteConsistency*] [**−x** *Maximum*] [**−U** *Userid*] [**−G** *Groupid*] [**−P** *Modes*] *LogicalVolume ...*

To Change the name of a Logical Volume

chlv −n*NewLogicalVolume LogicalVolume*

> **Note:** Changing the name of a logged logical volume requires that you run the **chfs −a** *Logname=NewLogName* on each filesystem using that log.

Description

> **Attention:** The name change option of this command is not allowed if the volume group is varied on in concurrent mode.

The **chlv** command changes the characteristics of a logical volume according to the command flags. The *LogicalVolume* parameter can be a logical volume name or logical volume ID. Each current characteristic for a logical volume remains in effect unless explicitly changed with the corresponding flag.

The changes you make with the **−a**, **−e**, **−s**, and **−u**flags take effect only when new partitions are allocated or partitions are deleted. The other flags take effect immediately.

To change the name of a logical volume, use the **−n** flag and use the *NewLogicalVolume* parameter to represent the new logical volume name. Do not use other flags with this syntax.

If the *volume group* which contains logical volume being changed is in big vg format, **U**, **G**, and **P** flags can

be used to set the ownership, group and permissions respectively, of the special device files. Only root user will be able to set these values. If the *volume group* is exported, these values can be restored upon import if **R** flag is specified with importvg command.

Notes:

1. Changes made to the logical volume are not reflected in the file systems. To change file system characteristics, use the **chfs** command.
2. To use this command, you must either have root user authority or be a member of the **system** group.

You can use the Web−based System Manager Volumes application (**wsm lvm** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chlv** fast path to run this command.

Flags

Note: When changing the characteristics of a striped logical volume, the **−d**, **−e**, and **−u** flags are not valid.

−a*Position*
Sets the intraphysical volume allocation policy (the position of the logical partitions on the physical volume). The *Position* variable is represented by one of the following:

m Allocates logical partitions in the outer middle section of each physical volume. This is the default position.

c Allocates logical partitions in the center section of each physical volume.

e Allocates logical partitions in the outer edge section of each physical volume.

ie Allocates logical partitions in the inner edge section of each physical volume.

im Allocates logical partitions in the inner middle section of each physical volume.

−b*BadBlocks*
Sets the bad−block relocation policy. The *BadBlocks* variable is represented by one of the following:

y Causes bad−block relocation to occur.

n Prevents bad block relocation from occurring.

−d*Schedule*
Sets the scheduling policy when more than one logical partition is written. Must use parallel or sequential to mirror striped lv. The *Schedule* variable is represented by one of the following:

p Establishes a parallel scheduling policy.

s Establishes a sequential scheduling policy.

When specifying policy of parallel or sequential strictness, set to "s" for super strictness.

−e*Range*
Sets the interphysical volume allocation policy (the number of physical volumes to extend across, using the volumes that provide the best allocation). The value of the *Range* variable is limited by the *Upperbound* variable, set with the **−u** flag, and is represented by one of the following:

x Allocates logical partitions across the maximum number of physical volumes.

m Allocates logical partitions across the minimum number of physical volumes.

−G*Groupid*
Specifies group ID for the logical volume special file.

−L*Label* Sets the logical volume label. The maximum size of the *Label* variable is 127 characters.

−n*NewLogicalVolume* Changes the name of the logical volume to that specified by the *NewLogicalVolume* variable. Logical volume names must be unique systemwide and can range from 1 to 15 characters.

−p*Permission* Sets the access permission to read−write or read−only. The *Permission* variable is represented by one of the following:

w Sets the access permission to read−write.

r Sets the access permission to read−only.

−P*Modes* Specifies permissions (file modes) for the logical volume special file.

−r *Relocate* Sets the reorganization flag to allow or prevent the relocation of the logical volume during reorganization. The *Relocate* variable is represented by one of the following:

y Allows the logical volume to be relocated during reorganization. If the logical volume is striped, the **chlv** command will not let you change the relocation flag to **y**.

n Prevents the logical volume from being relocated during reorganization.

−s*Strict* Determines the strict allocation policy. Copies of a logical partition can be allocated to share or not to share the same physical volume. The *Strict* variable is represented by one of the following:

y Sets a strict allocation policy, so copies of a logical partition cannot share the same physical volume.

n Does not set a strict allocation policy, so copies of a logical partition can share the same physical volume.

s Sets a super strict allocation policy, so that the partitions allocated for one mirror cannot share a physical volume with the paritions from another mirror

Note: When changing a non superstrict logical volume to a superstrict logical volume you must use the **−u** flag.

−t*Type* Sets the logical volume type. The maximum size is 31 characters. If the logical volume is striped, you cannot change *Type* to boot.

−U*Userid* Specifies user ID for the logical volume special file.

−u*Upperbound* Sets the maximum number of physical volumes for new allocation. The value of the *Upperbound* variable should be between one and the total number of physical volumes. When using striped logical volumes or super strictness the upper bound indicates the maximum number of physical volumes allowed for each mirror copy.

−v*Verify* Sets the write−verify state for the logical volume. Causes all writes to the logical volume either to be verified with a follow−up read or not to be verified with a follow−up read. The *Verify* variable is represented by one of the following:

y Causes all writes to the logical volume to be verified with a follow−up read.

n Causes all writes to the logical volume not to be verified with a follow−up read.

−w *MirrorWriteConsistency*

y Turns on mirror write consistency which ensures data consistency among mirrored copies of a logical volume during normal I/O processing.

n No mirror write consistency. See the **−f** flag of the **syncvg** command.

−x *Maximum* Sets the maximum number of logical partitions that can be allocated to the logical volume. The maximum number of logical partitions per logical volume is 32,512.

Examples

1. To change the interphysical volume allocation policy of logical volume `lv01`, enter:

   ```
   chlv  -e m
   lv01
   ```

   The interphysical volume allocation policy is set to minimum.

2. To change the type of logical volume `lv03`, enter:

   ```
   chlv  -t copy  lv03
   ```

3. To change the permission of logical volume `lv03` to read-only, enter:

   ```
   chlv  -p r  lv03
   ```

   Logical volume `lv03` now has read-only permission.

4. To change the type to `paging` and the maximum number of physical volumes for logical volume `lv03`, enter:

   ```
   chlv -t paging -u 10 lv03
   ```

   The change in the type of logical volume takes effect immediately, but the change in the maximum number of physical volumes does not take effect until a new allocation is made.

5. To change the allocation characteristics of logical volume `lv07`, enter:

   ```
   chlv -a e -e x -r y -s n -u 5 lv07
   ```

Files

/usr/sbin Directory where **chlv** command resides.

Related Information

The **chfs** command, **extendlv** command, **lslv** command, **mklv** command, **mklvcopy** command, **reorgvg** command, **rmlvcopy** command, **syncvg** command.

The Logical Volume Storage Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and allocation characteristics.

The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the structure, main menus, and tasks that are done with SMIT.

Setting up and running Web-based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

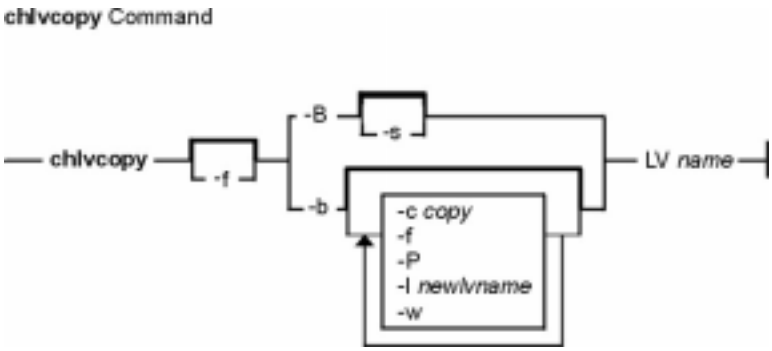
AIX HACMP/6000 Concepts and Facilities.

chlvcopy Command

Purpose

Marks or unmarks mirror copy as a split mirror.

Syntax



chlvcopy [**−f**] { **−B** [**−s**] } | { **−b** [**−c***copy*] [**−f**] [**−P**] [**−l***newlvname*] [**−w**] } LV*name*

Description

Notes:

1. To use this command, you must either have root user authority or be a member of the system group.
2. If persistence is used either by using the **−P** flag or by creating a child backup logical volume device by using the **−l** flag, it will cause the volume group to be usable only on 4.3.2.0 or later versions of AIX. This is true even after removal of split mirror copy designation of the parent logical volume and the child backup logical volumes.
3. For **chvlcopy** to be successful in a concurrent volume group environment, all the concurrent nodes must be at AIX Version 4.3.2 or later.

All partitions of a logical volume must be freshed before **chvlcopy** can mark a mirror copy as a split mirror. Only one copy may be designated as an online split mirror copy.

Although the **chlvcopy** command can mark online split mirror copies on logical volumes that are open (including logical volumes containing mounted file systems), this is not recommended unless the application is at a known state at the time the copy is marked as a split mirror. The split mirror copy is internally consistent at the time the **chlvcopy** command is run, but consistency is lost between the logical volume and the split mirror copy if the logical volume is accessed by multiple processes simultaneously and the application is not at a known state. When marking an open logical volume, data may be lost or corrupted. Logical volumes should be closed before marking online split mirror copies in order to avoid a potential corruption window.

If the persistence flag is not set to prevent the loss of backup data, the volume group should be set to not automatically varyon and the **−n** flag should be used with **varyonvg** to prevent stale partitions from being resynced. If the persistence flag (**−P**) is set, the following applies: In the event of a crash while an online split mirror copy exists (or multiples exist), the existence of copies is retained when the system is rebooted.

Flags

−b	Marks a mirror copy as a split mirror copy.
−c *copy*	Mirror copy to mark as split mirror copy. The allowed values of copy are 1, 2, or 3. If this option is not specified the default for copy is the last mirror copy of the logical volume.
−B	Unmarks a mirror as split mirror copy. It will also attempt to remove the child backup logical volume, if one was created with the **−l** option.
−f	Forces split mirror copy even if there are stale partitions. If used with the **−B** option, the child backup logical volume if one was created with the **−l** option, will be removed with the **force** option.
−l*newlvname*	New name of the backup logical volume. Specifying the **−l** flag also sets the persistence option, allowing applications to access split mirror copy via *newlvname*.
−P	Maintains information about the existence of an online split mirror copy across a reboot and also allows other nodes (in a concurrent mode environment) to be aware of the existence of the online split mirror copy.
−s	Starts a background **syncvg** for the logical volume.
−w	Allows split mirror copy to be writable (default is to create the split mirror copy as READ ONLY).
LV*name*	Logical volume to act on.

Related Information

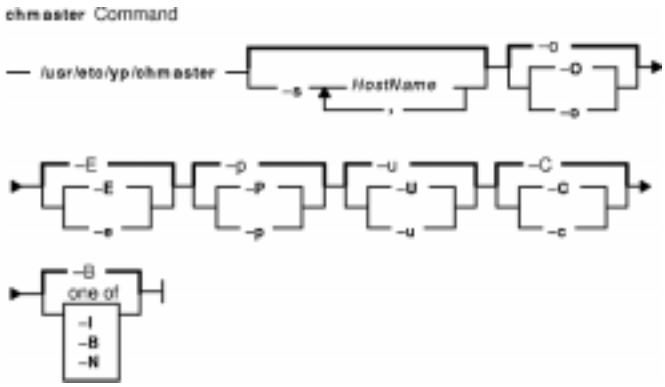
The **readlvcopy** and **chfs** commands.

chmaster Command

Purpose

The **chmaster** command executes the **ypinit** command and restarts the NIS daemons to change a master server.

Syntax



/usr/etc/yp/chmaster [**−s** *HostName* [**,** *HostName* ...]] [**−O** |**−o**] [**−E** |**−e**] [**−P**|**−p**] [**−U**|**−u**] [**−C**|**−c**] [**−I**|**−B**| **−N**]

Description

The **chmaster** command invokes the **ypinit** command to update the NIS maps for the current domain, assuming that the domain name of the system is currently set. After the **ypinit** command completes successfully, the **chmaster** command comments or uncomments the entries in the **/etc/rc.nfs** file for the **ypserv** command, **yppasswdd** command, **ypupdated** command, and **ypbind** command.

You can use the Web−based System Manager Network application (**wsm network** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chmaster** fast path to run this command.

Flags

−B	Updates the **/etc/rc.nfs** file to start the appropriate daemons, invokes the **ypinit** command, and starts the daemons.
−C	Starts the **ypbind** daemon along with the **ypserv** daemon. This flag is the default.
−c	Suppresses the start of the **ypbind** daemon.
−E	Exits from the **ypinit** command and the **chmaster** command if errors are encountered. This flag is the default.
−e	Suppresses an exit from the **ypinit** command and the **chmaster** command if errors are encountered.
−I	Directs the **chmaster** command to change the **/etc/rc.nfs** file to start the appropriate daemons on the next system restart. The execution of the **ypinit** command occurs when this command is invoked.
−N	Invokes the **ypinit** command and starts the appropriate daemons. No changes

	are made to the **/etc/rc.nfs** file.
−O	Overwrites existing maps for this domain.
−o	Prevents the overwriting of NIS maps. This flag is the default.
−P	Starts the **yppasswdd** daemon along with the **ypserv** daemon.
−p	Suppresses the start of the **yppasswdd** daemon. This flag is the default.
−s*HostName* [**,** *HostName*]	Specifies the slave host names for the slave for this master server. The **chmaster** command automatically adds the current host to this list.
−U	Starts the **ypupdated** daemon along with the **ypserv** daemon.
−u	Suppresses the start of the **ypupdated** daemon. This flag is the default.

Examples

To invoke the **ypinit** command to rebuild the NIS maps for the current domain, enter:

```
chmaster −s chopin −O −p −u −B
```

In this example, the `chmaster` command overwrites the existing maps and the **yppasswdd** and **ypupdated** daemons are not started. The host name `chopin` is specified to be a slave server.

Files

/etc/rc.nfs	Contains the startup script for the NFS and NIS daemons.
/var/yp/*domainname*	Contains the NIS maps for the NIS domain.

Related Information

The **mkclient** command, **rmyp** command, **smit** command, **ypinit** command.

The **ypbind** daemon, **yppasswdd** daemon, **ypserv** daemon, **ypupdated** daemon.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Network Information Service (NIS) in *AIX Version 4.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

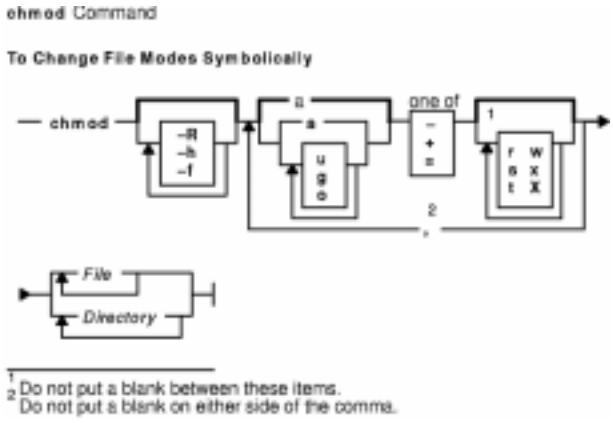
chmod Command

Purpose

Changes file modes.

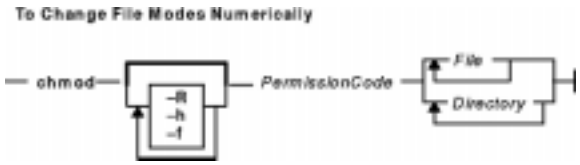
Syntax

To Change File Modes Symbolically



chmod [**−R**] [**−h**] [**−f**] [[**u**] [**g**] [**o**] | [**a**]] { { **−** | **+** | **=** } [**r**] [**w**] [**x**] [**X**] [**s**] [**t**] } { *File ...* | *Directory ...* }

To Change File Modes Numerically



chmod [**−R**] [**−h**] [**−f**] *PermissionCode* { *File ...* | *Directory ...* }

Description

The **chmod** command modifies the mode bits and the extended access control lists (ACLs) of the specified files or directories. The mode can be defined symbolically or numerically (absolute mode).

When a symbolic link is encountered and you have not specified the **−h** flag, the **chmod** command changes the mode of the file or directory pointed to by the link and not the mode of the link itself. If you specify the **−h** flag, the **chmod** command prevents this mode change.

If you specify both the **−h** flag and the **−R** flag, the **chmod** command descends the specified directories recursively, and when a symbolic link is encountered, the mode of the file or directory pointed to by the link is not changed.

Flags

−f Suppresses all error reporting except invalid permissions and usage statements.

−h Suppresses a mode change for the file or directory pointed to by the encountered symbolic link.

> **Note:** This behavior is slightly different from the behavior of the **−h** flag on the **chgrp** and **chown** commands because mode bits cannot be set on symbolic links.

−R Descends only directories recursively, as specified by the pattern *File*...|*Directory*.... The **−R** flag changes the file mode bits of each directory and of all files matching the specified pattern. See Example 6.

When a symbolic link is encountered and the link points to a directory, the file mode bits of that directory are changed but the directory is not further traversed.

Symbolic Mode

To specify a mode in symbolic form, you must specify three sets of flags.

> **Note:** Do not separate flags with spaces.

The first set of flags specifies who is granted or denied the specified permissions, as follows:

u File owner.

g Group and extended ACL entries pertaining to the file's group.

o All others.

a User, group, and all others. The **a** flag has the same effect as specifying the **ugo** flags together. If none of these flags are specified, the default is the **a** flag and the file creation mask (umask) is applied.

The second set of flags specifies whether the permissions are to be removed, applied, or set:

− Removes specified permissions.

+ Applies specified permissions.

= Clears the selected permission field and sets it to the permission specified. If you do not specify a permission following **=**, the **chmod** command removes all permissions from the selected field.

The third set of flags specifies the permissions that are to be removed, applied, or set:

r Read permission.

w Write permission.

x Execute permission for files; search permission for directories.

X Execute permission for files if the current (unmodified) mode bits have at least one of the user, group, or other execute bits set. The **X** flag is ignored if the *File* parameter is specified and none of the execute bits are set in the current mode bits.

Search permission for directories.

s Set−user−ID−on−execution permission if the **u** flag is specified or implied. Set−group−ID−on−execution permission if the **g** flag is specified or implied.

t For directories, indicates that only file owners can link or unlink files in the specified directory. For files, sets the **save−text** attribute.

Numeric or Absolute Mode

The **chmod** command also permits you to use octal notation for the mode. The numeric mode is the sum of one or more of the following values:

4000 Sets user ID on execution.

2000 Sets group ID on execution.
1000 Sets the link permission to directories or sets the **save−text** attribute for files.
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.

Notes:

1. Specifying the mode numerically disables any extended ACLs. Refer to "Access Control Lists" in *AIX Version 4.3 System User's Guide: Operating System and Devices* for more information.
2. Changing group access permissions symbolically also affects the extended ACL entries. The group entries in the ACL that are equal to the owning group of the file are denied any permission that is removed from the mode. Refer to "Access Control Lists" for more information.
3. You can specify multiple symbolic modes separated with commas. Operations are performed in the order they appear from left to right.
4. You must specify the mode symbolically when removing the set−group−ID−on−execution permission from directories.

Security

Access Control: This program should be installed as a normal user program in the Trusted Computing Base.

Only the owner of the file or the root user can change the mode of a file.

Exit Status

This command returns the following exit values:

0 The command executed successfully and all requested changes were made.
>0 An error occurred.

Examples

1. To add a type of permission to several files:

   ```
   chmod  g+w  chap1  chap2
   ```

   This adds write permission for group members to the files `chap1` and `chap2`.

2. To make several permission changes at once:

   ```
   chmod  go−w+x  mydir
   ```

   This denies group members and others the permission to create or delete files in `mydir` (**go−w**) and allows group members and others to search `mydir` or use it in a path name (**go+x**). This is equivalent to the command sequence:

```
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 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 `cmd` shell command file, then you can run it by entering:

   ```
   cmd
   ```

   > **Note:** Depending on the **PATH** shell variable, you may need to specify the full path to the `cmd` file.

4. To use Set−ID Modes:

   ```
   chmod  ug+s  cmd
   ```

   When the `cmd` command is executed, the effective user and group IDs are set to those that own the `cmd` file. Only the effective IDs associated with the child process that runs the `cmd` command are changed. The effective IDs of the shell session remain unchanged.

   This feature allows you to permit access to restricted files. Suppose that the `cmd` program has the Set−User−ID Mode enabled and is owned by a user called `dbms`. The user `dbms` is not actually a person, but might be associated with a database management system. The user `betty` does not have permission to access any of `dbms`'s data files. However, she does have permission to execute the `cmd` command. When she does so, her effective user ID is temporarily changed to `dbms`, so that the `cmd` program can access the data files owned by the user `dbms`.

   This way the user `betty` can use the `cmd` command 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 sets read and write permission for the owner, and it sets read−only mode for the group and others. This also removes all extended ACLs that might be associated with the file.

6. To recursively descend directories and change file and directory permissions given the tree structure:

   ./dir1/dir2/file1

   ./dir1/dir2/file2

   ./dir1/file1

   enter this command sequence:

   ```
   chmod −R 777 f*
   ```

   which will change permissions on ./dir1/file1.

But given the tree structure of:

./dir1/fdir2/file1

./dir1/fdir2/file2

./dir1/file3

the command sequence:

```
chmod -R 777 f*
```

will change permissions on:

./dir1/fdir2

./dir1/fdir2/file1

./dir1/fdir2/file2

./dir1/file3

File

/usr/bin/chmod Contains the **chmod** command .

Related Information

The **acledit** command, **aclget** command, **aclput** command, **chown** command, **chgrp** command, **ls** command.

The **chmod** subroutine, **fchmod** subroutine.

File Ownership and User Groups in *AIX Version 4.3 System User's Guide: Operating System and Devices* introduces file ownership and permissions to access files and directories.

Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices* describes system security.

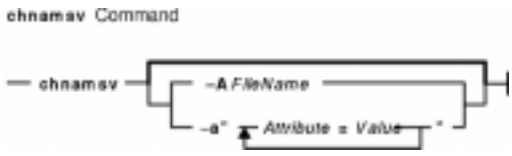
Trusted Computing Base Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices* describes the part of the system that is responsible for enforcing system information security policies.

chnamsv Command

Purpose

Changes TCP/IP−based name service configuration on a host.

Syntax



chnamsv [**−a"***Attribute=Value ...***"** | **−A** *FileName*]

Description

The **chnamsv** high−level command changes a TCP/IP−based name service configuration on a host. The command changes the **/etc/resolv.conf** file only. The command does not change the name server database.

If you change the name service configuration for a client, the **chnamsv** command calls the **namerslv** low−level command to change the **resolv.conf** configuration file appropriately.

You can use the Web−based System Manager Network application (**wsm network** fast path) run this command. You could also use the System Management Interface Tool (SMIT) **smit namerslv** fast path to run this command.

Flags

−A *FileName*
: Specifies name of file containing the named server initialization information.

−a"*Attribute=Value...***"**
: Specifies a list of attributes and their corresponding values to be used for updating the named server initialization files in the database.

  Attributes can be either of the following:

  domain
  : The domain name of the changed named server

  nameserver
  : The Internet address of the changed name server

Examples

1. To update the named server initialization files, enter the command in the following format:

   ```
   chnamsv   −a"domain=austin.century.com   nameserver=192.9.200.1"
   ```

   In this example the domain name and name server address are updated. The previous domain and name server are overwritten.

2. To update name server initialization files according to information in another file, enter the command in the following format:

```
chnamsv   -A   namsv.file
```

In this example, the file that contains the updated information is `namsv.file`.

Files

/etc/resolv.conf Contains DOMAIN name server information for local resolver routines.

Related Information

The **namerslv** command.

TCP/IP Name Resolution in *AIX Version 4.3 System Management Guide: Communications and Networks*.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

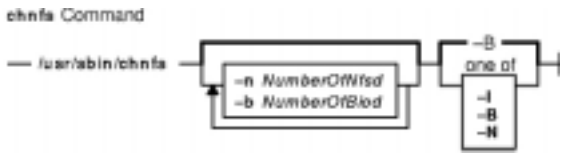
TCP/IP Reference in *AIX Version 4.3 System Management Guide: Communications and Networks*.

chnfs Command

Purpose

Changes the configuration of the system to invoke a specified number of **biod** and **nfsd** daemons.

Syntax



/usr/sbin/chnfs [**−n***NumberOfNfsd*] [**−b***NumberOfBiod*] [**−I** | **−B** | **−N**]

Description

The **chnfs** command invokes the number of **biod** and **nfsd** daemons specified. The **chnfs** command does this by changing the objects in the SRC database. These changes take place at different times depending on the flags chosen.

Flags

−B	Temporarily stops the daemons currently running on the system, modifies the SRC database code to reflect the new number, and restarts the daemons indicated. This flag is a default.
−b*NumberOfBiod*	Specifies the number of **biod** daemons to run on the system.
−I	Changes the objects in the SRC database so that the number of daemons specified will be run during the next system restart.
−N	Temporarily stops the daemons currently running on the system and restarts the number of daemons indicated.
−n*NumberOfNfsd*	Specifies the number of **nfsd** daemons to run on the system.

Examples

To set the number of **nfsd** daemons to 10 and the number of **biod** daemons to 4, enter:

```
chnfs    −n    10    −b    4    −I
```

This change will be made for the next system restart.

Related Information

The **mknfs** command, **rmnfs** command.

The **biod** daemon, **nfsd** daemon.

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

NFS Installation and Configuration in *AIX Version 4.3 System Management Guide: Communications and Networks*.

List of NFS Commands in *AIX Version 4.3 System Management Guide: Communications and Networks*.

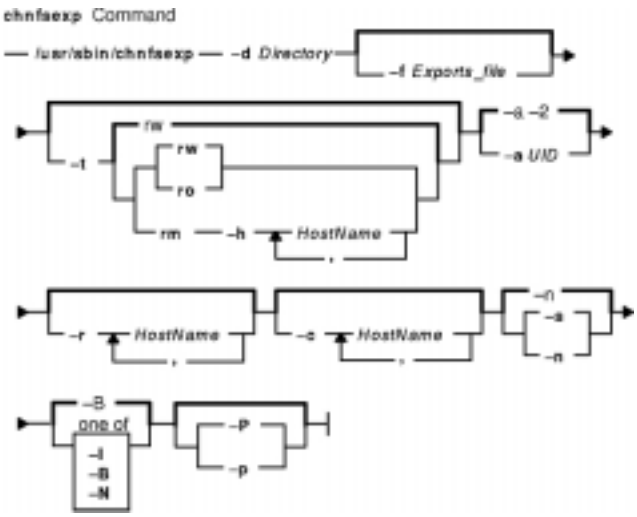
System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

chnfsexp Command

Purpose

Changes the options used to export a directory to NFS clients.

Syntax



/usr/sbin/chnfsexp−d *Directory* [**−f***Exports_file*] [**−t** [{ **rw** | **ro** | **rm−h** *HostName* [**,***HostName ...*] }]] [**−a** *UID*] [**−r** *HostName* [**,***HostName ...*]] [**−c** *HostName* **,** *HostName ...*]] [**−s** | **−n**] [**−I** | **−B** | **−N**] [**−P** | **−p**]

Description

The **chnfsexp** command takes as a parameter a directory that is currently exported to NFS clients and changes the options used to export that directory. The options specified on the command line will replace those currently being used.

Flags

−a*UID*	Uses the *UID* parameter as the effective user ID only if a request comes from an unknown user. The default value of this option is −2. **Note:** Root users (uid 0) are always considered "unknown" by the NFS server, unless they are included in the root option. Setting the value of *UID* to −1 disables anonymous access.
−B	Updates the entry in the **/etc/exports** file and the **exportfs** command is executed to again export the directory immediately.
−c*HostName* [**,***HostName*] ...	Gives mount access to each of the clients listed. A client can either be a host or a netgroup. The default is to allow all hosts access.
−d*Directory*	Specifies the exported directory that is to be changed.
−f*Exports_file*	Specifies the full path name of the exports file to use if other than the **/etc/exports** file.
−h*Hostname* [**,***HostName*] ...	Specifies which hosts have read−write access to the directory. This option is valid only when the directory is exported read−mostly.

−I	Adds an entry in the **/etc/exports** file so that the next time the **exportfs** command is run, usually during system restart, the directory will be exported.
−N	Does not modify the entry in the **/etc/exports** file but the **exportfs** command is run with the correct parameters so that the export is changed.
−n	Does not require client to use the more secure protocol. This flag is the default.
−P	Specifies that the exported directory is to be a public directory. This flag only applies to AIX Version 4.2.1 or later.
−p	Specifies that the exported directory is not a public directory. This flag only applies to AIX Version 4.2.1 or later.
−r *HostName* [**,***HostName*] ...	Gives root users on specified hosts access to the directory. The default is for no hosts to be granted root access.
−s	Requires clients to use a more secure protocol when accessing the directory.
−t *Type*	Specifies one of the following types of mount access allowed to clients: **rw** Exports the directory with read−write permission. This is the default. **ro** Exports the directory with read−only permission. **rm** Exports the directory with read−mostly permission. If this type is chosen, the **−h** flag must be used to specify hosts that have read−write permission.

Examples

1. To change the list of hosts that have access to an exported directory and to make this change occur immediately and upon each subsequent system restart, enter:

   ```
   chnfsexp     −d     /usr     −t     rw     −c     host1,host3,host29,grp3,grp2     −B
   ```

   In this example, the `chnfsexp` command changes the attributes of the `/usr` directory to give read and write permission to the `host1`, `host3`, and `host29` hosts, and the `grp3` and `grp2` netgroups.

2. To change the list of hosts that have access to an exported directory, to specify the path name of the exports file, and to make this change occur immediately and upon each subsequent system restart, enter:

   ```
   chnfsexp     −d     /usr     −t     rw     −c     host1,host3,host29,grp3,grp2
                −f     /etc/exports.other     −B
   ```

In this example, the `chnfsexp` command changes the attributes of the `/usr` directory to give read and write permission to the `host1`, `host3`, and `host29` hosts: the `grp3` and `grp2` netgroups; and specifies the path name of the exports file as `/etc/exports.other`.

Files

/etc/exports Lists directories the server can export.

Related Information

The **exportfs** command, **mknfsexp** command, **rmnfsexp** command.

Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

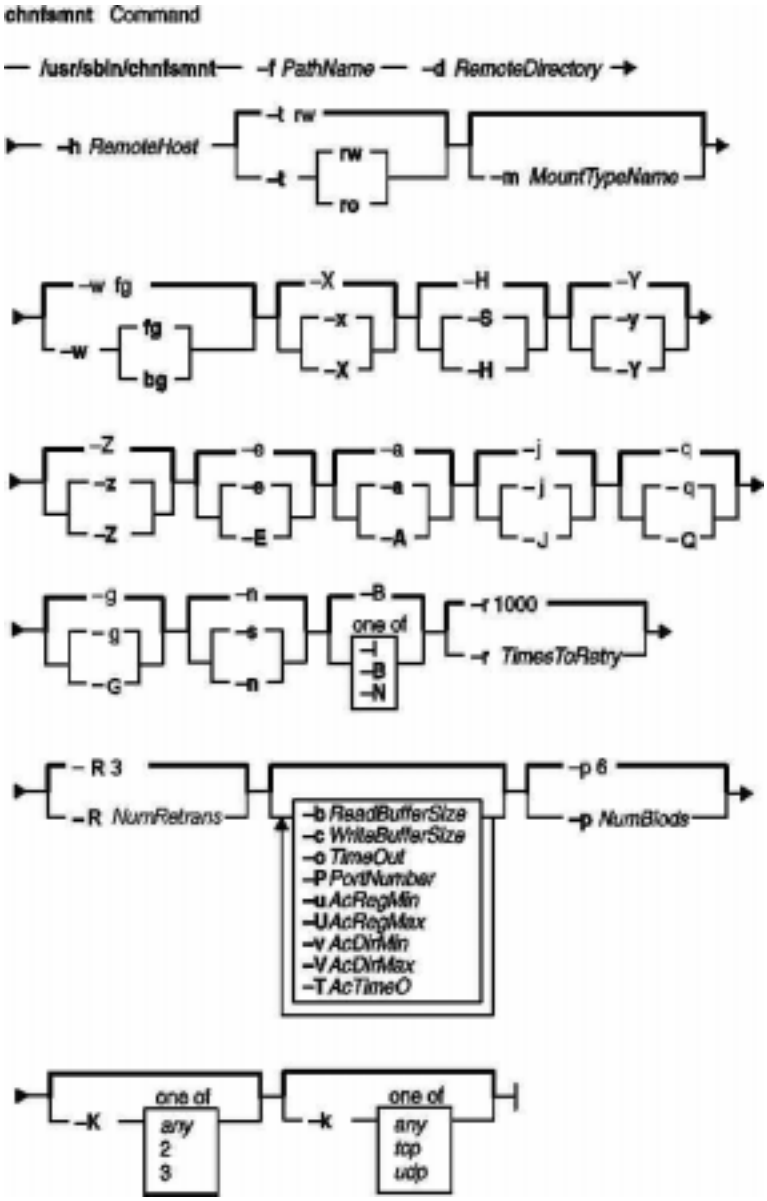
List of NFS Commands.

chnfsmnt Command

Purpose

Changes the options used to mount a directory from an NFS server.

Syntax



/usr/sbin/chnfsmnt **−f** *PathName* **−d** *RemoteDirectory* **−h** *RemoteHost* [**−t** { **rw** | **ro** }] [**−m** *MountTypeName*] [**−w** { **fg** | **bg** }] [**−X** | **−x**] [**−S** | **−H**] [**−Y** | **−y**] [**−Z** | **−z**] [**−e** | **−E**] [**−a** | **−A**] [**−j** | [**−J**] [**−q** | [**−Q**] [**−g** | [**−G**] [**−s** | **−n**] [**−I** | **−B** | **−N**] [**−r** *TimesToRetry*] [**−R** *NumRetrans*] [**−b** *ReadBufferSize*] [**−c** *WriteBufferSize*] [**−o** *TimeOut*] [**−P** *PortNumber*] [**−u** *AcRegMin*] [**−U** *AcRegMax*] [**−v** *AcDirMin*] [**−V** *AcDirMax*] [**−T** *AcTimeO*] [**−p** *NumBiods*] [**−K** { *any* | *2* | *3* }] [**−k** { *any* | *tcp* | *udp* }]

Description

The **chnfsmnt** command changes the mount options of a currently mounted file system. However, before you can change the attributes of a mount, either the **/etc/filesystems** file must contain an entry for the file system or the file system needs to be currently mounted from an NFS server. This command unmounts the directory, changes the specified options, and mounts the directory with the new options.

Flags

−A	The **/etc/filesystems** entry for this file system will specify that it should be automatically mounted at system restart.
−a	The **/etc/filesystems** entry for this file system specifies that it should not be automatically mounted at system restart. This is the default.
−B	Modifies the entry in the **/etc/filesystems** file and remounts the file system using the flags and parameters specified. This flag is the default.
−b *ReadBufferSize*	Indicates the size of the read buffer in *N* bytes.
−c*WriteBufferSize*	Indicates the size of the write buffer in *N* bytes.
−d *RemoteDirectory*	Specifies the directory that will be mounted on the path name specified.
−E	Allows keyboard interrupts on hard mounts.
−e	Prevents keyboard interrupts on hard mounts. This flag is the default.
−f *PathName*	Specifies the mount point for the directory.
−G	Directs any file or directory created on the file system to inherit the group ID of the parent directory.
−g	Does not direct new files or directories created on the file system to inherit the group ID of the parent directory. This is the default.
−H	Makes the mount a hard mount, which causes the client to continue trying until the server responds.
−h *RemoteHost*	Specifies the NFS server that is exporting the directory.
−I	Changes the entry in the **/etc/filesystems** file but does not remount the directory.
−J	Indicates that **acl**s are used on this mount.
−j	Indicates that **acl**s are not used on this mount. This is the default.
−K	Specifies the NFS version used for this NFS mount. This flag only applies to AIX Version 4.2.1 or later. Options are: *any* Uses the mount command to determine the correct match, first attempting the highest NFS version available. 2 Specifies NFS Version 2. 3 Specifies NFS Version 3.
−k	Specifies the transport protocol used for the mount. This flag only applies to AIX Version 4.2.1 or later. Options are: *any* Uses the mount command to select the protocol to use. TCP protocol is the preferred protocol. *tcp* Specifies the TCP protocol. *udp* Specifies the UDP protocol.
−m *MountTypeName*	Corresponds to the *type* field in the stanza of the entry in the **/etc/filesystems** file. When the **mount −t** command *MountTypeName* is issued, all of the currently unmounted file systems with a field type equal to the string are mounted.
−N	Prevents modification of the corresponding entry in the **/etc/filesystems** file if it exists. If the directory is currently mounted, it is unmounted and then mounted again with the flags and parameters specified.

−n	Instructs the mount not to use a more secure protocol. This flag is the default.
−o *TimeOut*	Indicates the length of the NFS time out in *N* tenths of a second.
−P *PortNumber*	Indicates the IP port number for the server.
−p *NumBiods*	Specifies the number of **biod** daemons that are allowed to work on a particular file system. The default is 6.
−Q	Requests that no posix pathconf information be exchanged and made available on an NFS Version 2 mount. Requires a mount Version 2 **rpc.mountd** at the NFS server.
−q	Specifies that no posix pathconf information is exchanged if mounted as an NFS Version 2 mount. This is the default.
−r *TimeToRetry*	Indicates the number of times to retry a mount. The default is 1000.
−R *NumRetrans*	Specifies, for a soft mount, the number of times that a request is to be transmitted if it is not acknowledged by the server. If the request goes unacknowledged after *NumRetrans* transmissions, the client gives up on the request. If this flag is not specified, the default value of 3 is used.
−S	Makes the mount a soft mount, which means that the system returns an error if the server does not respond.
−s	Instructs the mount to use a more secure protocol.
−T*AcTimeO*	Sets minimum and maximum time allowed for regular files and directories to *AcTimeO* seconds. If this option is specified, the other cached attribute times are overridden.
−t	Specifies whether the directory will be mounted as read−write or read−only. **rw** Mounts the directory read−write. This type is the default for the system. **ro** Mounts the directory read−only.
−U*AcRegMax*	Holds cached attributes for no more than *AcRegMax* seconds after file modification.
−u*AcRegMin*	Holds cached attributes for at least *AcRegMin* seconds after file modification.
−V*AcDirMax*	Holds cached attributes for no more than *AcDirMax* seconds after directory update.
−v*AcDirMin*	Holds cached attributes for at least *AcDirMin* seconds after directory update.
−w { **fg** | **bg** }	Indicates whether the mount should be attempted in the foreground (fg) or background (bg). If **bg** is specified and the attempt to mount the directory fails, the mount will be tried again in the background. The **fg** parameter is the default.
−X	Specifies that the server does support long device numbers. This is the default.
−x	Specifies that the server does not support long device numbers.
−Y	Indicates that the execution of **suid** and **sgid** programs are allowed in this file system. This is the default.
−y	Indicates that the execution of **suid** and **sgid** programs is not allowed in this file system.
−Z	Indicates that device access through this mount is allowed. This is the default.
−z	Indicates that device access through this mount is not allowed.

Examples

To change a mount to read−only, enter:

```
chnfsmnt −f /usr/man −d /usr/man −h host1 −t ro
```

In this example, the `chnfsmnt` command changes the attributes of the mounted directory to read−only.

Files

/etc/filesystems Lists the remote file systems to be mounted during the system restart.

Related Information

The **mknfsmnt** command, **mount** command, **rmnfsmnt** command.

How to Mount an NFS File Explicitly in *AIX Version 4.3 System Management Guide: Communications and Networks*.

List of NFS Commands in *AIX Version 4.3 System Management Guide: Communications and Networks*.

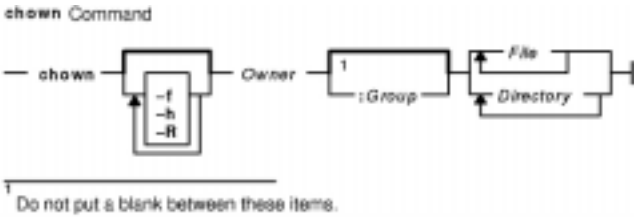
Network File System (NFS) Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

chown Command

Purpose

Changes the owner or group associated with a file.

Syntax



chown [**−f**] [**−h**] [**−R**] *Owner* [**:***Group*] { *File ...* | *Directory ...* }

Description

The **chown** command changes the owner of the file specified by the *File* parameter to the user specified by the *Owner* parameter. The value of the *Owner* parameter can be a user ID or a login name found in the **/etc/passwd** file. Optionally, a group can also be specified. The value of the *Group* parameter can be a group ID or a group name found in the **/etc/group** file.

Only the root user can change the owner of a file. You can change the group of a file only if you are a root user or if you own the file. If you own the file but are not a root user, you can change the group only to a group of which you are a member.

When a symbolic link is encountered and you have not specified the **−h** flag, the **chown** command changes the ownership of the file or directory pointed to by the link and not the ownership of the link itself.

If you specify the **−h** flag, the **chown** command has the opposite effect and changes the ownership of the link itself and not that of the file or directory pointed to by the link.

If you specify the **−R** flag, the **chown** command recursively descends the specified directories.

If you specify both the **−h** flag and the **−R** flag, the **chown** command descends the specified directories recursively, and when a symbolic link is encountered, the ownership of the link itself is changed and not that of the file or directory pointed to by the link.

Flags

−f Suppresses all error messages except usage messages.

−h Changes the ownership of an encountered symbolic link and not that of the file or directory pointed to by the symbolic link.

−R Descends directories recursively, changing the ownership for each file. When a symbolic link is encountered and the link points to a directory, the ownership of that directory is changed but the directory is not further transversed.

Security

Access Control: This program should be installed as a normal user program in the Trusted Computing Base.

Exit Status

This command returns the following exit values:

0 The command executed successfully and all requested changes were made.
>0 An error occurred.

Examples

1. To change the owner of the file `program.c:`

   ```
   chown jim program.c
   ```

   The user access permissions for `program.c` now apply to `jim`. As the owner, `jim` can use the **chmod** command to permit or deny other users access to `program.c.`

2. To change the owner and group of all files in the directory `/tmp/src` to owner `john` and group `build`:

   ```
   chown -R john:build /tmp/src
   ```

Files

/usr/bin/chown	The **chown** command
/etc/group	File that contains group IDs
/etc/passwd	File that contains user IDs

Related Information

The **chgrp** command, **chmod** command.

The **chown** subroutine, **fchown** subroutine.

The File Ownership and User Groups in *AIX Version 4.3 System User's Guide: Operating System and Devices* introduces file ownership and permissions to access files and directories.

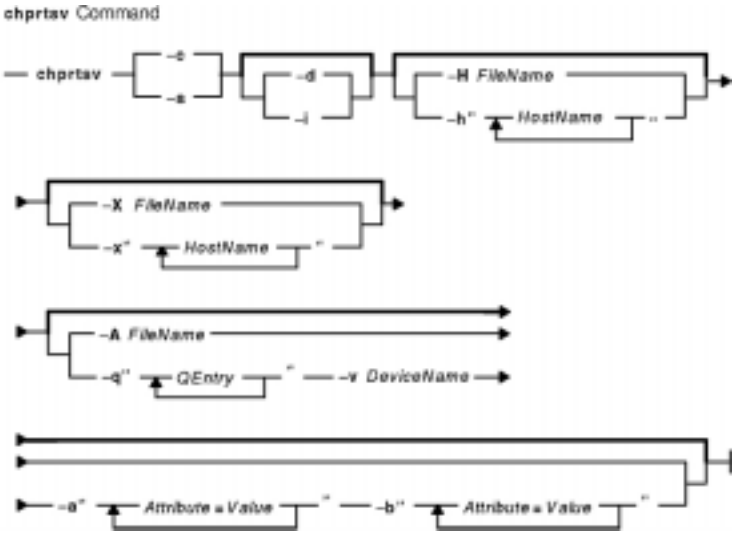
The Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices* describes system security.

chprtsv Command

Purpose

Changes a print service configuration on a client or server machine.

Syntax



chprtsv **−c** | **−s** [**−d** | **−i**] [**−h"***HostName...***"** | **−H** *FileName*] [**−x"***HostName...***"** | **−X** *FileName*] [**−q"***QEntry***"−v** *DeviceName* **−a"***Attribute* =*Value...***"** **−b"***Attribute* =*Value...***"** | **−A** *FileName*]

Description

The **chprtsv** high−level command changes print service configuration on a client or server machine.

To change print service for a client, the **chprtsv** command does the following:

1. Disables the client spool queue with the **chque** and **chquedev** commands.
2. Changes the appropriate entries in the **/etc/qconfig** file with the **chque** and **chquedev** commands.
3. Enables the client spool queue with the **chque** and **chquedev** commands.

To change print service for a server, the **chprtsv** command does the following:

1. Calls the **ruser** low−level command to change remote users configured on the print server, if necessary.
2. Calls the **chque** and **chquedev** commands to change the print queues and entries in the **qconfig** file, if necessary.
3. Calls the SRC **refresh** command to restart the **lpd** and **qdaemon** servers.

If you want to change the attributes of a queue, you must specify the queue name and the attributes associated with the queue. If you want to change the attributes of the queue device, you must specify queue name, queue device name, and the attributes associated with the queue device.

The changes you make with the **chprtsv −i** command go into effect on the system database and on the current active system.

If you want the changes you make to go into effect at system startup time without affecting the current system, use the **chprtsv−d** command to change only TCP/IP and its associated network interfaces in the system database only.

Flags

−A *FileName*
Specifies the name of the file containing **qconfig** command−related entries.

−a "*Attribute*=*Value*..."
Specifies a list of attributes with corresponding values to be used for updating the spooler's **qconfig** file or object class. The list should be enclosed in quotes. Valid attribute types follow:

acctfile (true/false)
Identifies the file used to save **print** accounting information. The default value of **false** suppresses accounting. If the named file does not exist, no accounting is done.

device
Identifies the symbolic name that refers to the device stanza.

discipline
Defines the queue−serving algorithm. The default, **fcfs**, means first come, first served. A value of **sjn** means shortest job next.

host
Specifies the name of the host from which to print. (The name of this host must be the same as the name specified by the *HostName* variable.)

l_statfilter
Translates long queue−status information from non−AIX format to AIX format.

s_statfilter
Translates short queue−status information from non−AIX format to AIX format.

up (true/false)
Defines the state of the queue. The default **true** indicates that it is running. A value of **false** indicates that it is not.

−b"*Attribute*=*Value*..."
Specifies a list of attributes with corresponding values for device stanza corresponding values to be used for updating the spooler's **qconfig** file or object class. The list should be enclosed in quotes. Valid attribute types follow:

access (write/both)
Specifies the type of access the backend has to the file specified by the **file** field. The **access** file has a value of **write** if the backend has write access to the file, or a value of **both** if the backend has both read and write access. This field is ignored if the file field has a value of **false**.

align (true/false)
Specifies whether the backend sends a form−feed control before starting the job if the printer has been idle. The default is **false**.

backend
Specifies the full path name of the backend, optionally followed by the flags and parameters to be passed to it.

feed
Specifies the number of separator pages to print when the device becomes idle, or takes a value of **never**, which

indicates that the backend is not to print separator pages.

file
Identifies the special file where the output of the backend is to be redirected. The default values of **false** indicates no redirection. In this case, the backend opens the output file.

header (never/always/group)
Specifies whether a header page prints before each job or group of jobs. The default is a value of **never** which indicates no header page. To produce a header page before each job, specify a value of **always**. To produce a header before each group of jobs for the same user, specify a value of **group**.

trailer (never/always/group)
Specifies whether a trailer page prints after each job or group of jobs. The default value of **never** indicates no trailer page. To produce a trailer page after each job, specify a value of **always**. To produce a trailer after each group of jobs for the same user, specify a value of **group**.

−c	Specifies to the **chprtsv** command to reconfigure print service for a client machine.
−d	Specifies that changes be reflected in the system database only, so that they can take effect at the next system startup.
−H *FileName*	Specifies the name of a file containing a list of host names to be included.
−h"*HostName...***"**	Specifies a list of host names to be included on the current list of remote users who can use the print server. Note that the queuing system does not support multibyte host names.
−i	Specifies that the change be reflected not only in the database, but also in the current running system.
−q"*QEntry***"**	Specifies a **qconfig** file entry to be removed.
−s	Specifies that print service reconfiguration is to be performed for a server machine.
−v *DeviceName*	Specifies a list of device stanzas to be changed.
−X *FileName*	Specifies the name of a file containing a list of host names to be excluded.
−x"*HostName...***"**	Specifies a list of host names to be excluded on the current list of remote users who can use the print server.

Examples

To reconfigure a print server, specify that the changes will take effect at the next startup, specify the file containing the host names, and then exclude some of those hosts, enter:

```
chprtsv −s −d −H ruser.inc −x "host1,host2,host3"
```

Files

/etc/qconfig	Contains configuration information for the printer queuing system.
/etc/hosts.lpd	Specifies foreign hosts that can print on the local host.

Related Information

The **chque** command, **chquedev** command, **ruser** command.

The **lpd** daemon, **qdaemon** daemon.

TCP/IP Reference in *AIX Version 4.3 System Management Guide: Communications and Networks*.

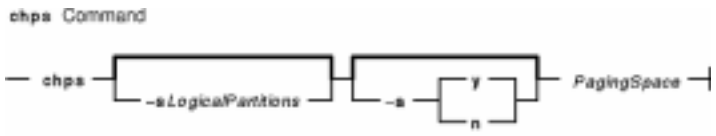
TCP/IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

chps Command

Purpose

Changes attributes of a paging space.

Syntax



chps [**−s***LogicalPartitions*] [**−a** { **y** | **n** }] *PagingSpace*

Description

The **chps** command changes attributes of a specific paging space. The *PagingSpace* parameter specifies the name of the paging space to be changed.

To change the size of a Network File System (NFS) paging space, the size of the file that resides on the server must first be changed and then the **swapon** command used to notify the client of the change in size of the paging space.

You can use the Web−based System Manager Devices application (**Devices** fast path) to change device characteristics. You could also use the System Management Interface Tool (SMIT) **smit chps** fast path to run this command.

> **Note:** The primary paging space is hardcoded in the boot record. Therefore, the primary paging space will always be activated when the system is restarted. The **chps** command is unable to deactivate the primary paging space.

Flags

−a Specifies to use a paging space at the next system restart.
- **y** Specifies that the paging space is active at subsequent system restarts.
- **n** Specifies that the paging space is inactive at subsequent system restarts.

−s*LogicalPartitions* Specifies the number of logical partitions to add.

Examples

1. To change the size of the `myvg` paging space, enter:

   ```
   chps −s4 myvg
   ```

   This adds four logical partitions to the `myvg` paging space.

2. To define the `PS02` paging space as configured and active at subsequent system restarts, enter:

   ```
   chps −a y PS02
   ```

This specifies that the `PS02` paging space is to be active at subsequent system restarts.

Files

/etc/swapspaces Specifies the paging space devices activated by the **swapon −a** command.

Related Information

The **lsps** command, **mkps** command, **rmps** command, **swapon** command.

The Paging Space Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains paging space and its allocation policies.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* provides information on working with files.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the structure, main menus, and tasks that are done with SMIT.

chpv Command

Purpose

Changes the characteristics of a physical volume in a volume group.

Syntax

chpv [**−a** *Allocation*] [**−v** *Availability*] [**−c**] *PhysicalVolume ...*

Description

> **Attention:** This command is not allowed if the volume group is varied on in concurrent mode.

The **chpv** command changes the state of the physical volume in a volume group by setting allocation permission to either allow or not allow allocation and by setting the availability to either available or removed. This command can also be used to clear the boot record for the given physical volume. Characteristics for a physical volume remain in effect unless explicitly changed with the corresponding flag.

> **Note:** To use this command, you must either have root user authority or be a member of the **system** group.

You can use the Web−based System Manager Volumes application (**wsm lvm** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chpv** fast path to run this command.

Flags

−a *Allocation* Sets the allocation permission for additional physical partitions on the physical volume specified by the *PhysicalVolume* parameter. Either allows (yes) the allocation of additional physical partitions on the physical volume, or prohibits (no) the allocation of additional physical partitions on the physical volume. The *Allocation* variable can be either:

y
Allows the allocation of additional physical partitions on the physical volume.

n
Prohibits the allocation of additional physical partitions on the physical volume. The logical volumes that reside on the physical volume can still be accessed.

−c Clears the boot record of the given physical volume.

−v *Availability* Sets the availability of the physical volume. If you set the availability to closed, logical input and output to the physical volume are stopped. You should close a physical volume when the physical volume is removed from operation. Access to physical volume data by the file system or the virtual memory manager is stopped, but you can continue to use the system management commands. The *Availability* variable can be either:

a
Makes a physical volume available for logical input and output.

r

Makes a physical volume unavailable (removed) for logical input and output. If the physical volume is required in order to maintain a volume group quorum, an error occurs and the physical volume remains open.

Examples

1. To close physical volume `hdisk03`, enter:

   ```
   chpv -v r hdisk03
   ```

   The physical volume is closed to logical input and output until the **-v a** flag is used.

2. To open physical volume `hdisk03`, enter:

   ```
   chpv -v a hdisk03
   ```

   The physical volume is now open for logical input and output.

3. To stop the allocation of physical partitions to physical volume `hdisk03`, enter:

   ```
   chpv -a n hdisk03
   ```

   No physical partitions can be allocated until the **-a y** flag is used.

4. To clear the boot record of a physical volume `hdisk3`, enter:
   ```
   chpv -c hdisk3
   ```

Files

/usr/sbin Directory where the **chpv** command resides.

/tmp Directory where temporary files are stored while the command is running.

Related Information

The **lspv** command.

The Logical Volume Storage Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and understanding the allocation characteristics.

Setting up and running Web-based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the structure, main menus, and tasks that are done with SMIT.

AIX HACMP/6000 Concepts and Facilities.

chque Command

Purpose

Changes the queue name.

Syntax

chque **−q** *Name* [**−a**'*Attribute*=*Value*' ...]

Description

The **chque** command changes the queue name by changing the stanza in the **qconfig** file specified by the **−q** flag. Within that stanza, each attribute that matches one of the *Attribute* = *Value* pairs given on the command line will be replaced by the one on the command line. If no match is found, the *Attribute* = *Value* pair is added to the end of the stanza. The device attribute cannot be changed.

You can use the Web−based System Manager Printer Queues application (**wsm printers** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chque** fast path to run this command.

> **Note:** Do not edit the **qconfig** file while there are active jobs in any queue. Editing includes both manual editing and use of the **chque**, **mkque**, **rmque**, **mkquedev**, **rmquedev**, or **chquedev** commands. It is recommended that all changes to the **qconfig** file be made using these commands. However, if manual editing is desired, first issue the **enq −G** command to bring the queuing system and the **qdaemon** to a halt after all jobs are processed. Then edit the **qconfig** file and restart the **qdaemon** with the new configuration.

Flags

−a'*Attribute* = *Value*'	Specifies the '*Attribute* = *Value*' to be added or replaced by the one entered on the command line. For a list of valid attributes, refer to the **/etc/qconfig** file.
−q*Name*	Specifies the current *Name* of the queue and of the stanza in the **qconfig** file that is to be changed.

Examples

To change the name of the host to `fred` for queue `lp0`, enter:

```
chque −qlp0 −a 'host = fred'
```

Files

/usr/bin/chque	Contains the **chque** command.
/etc/qconfig	Contains the configuration file.

Related Information

The **chquedev** command, **lsque** command, **mkque** command, **rmque** command.

The **qconfig** file.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

Changing / Showing Queue Characteristics in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Overview for System Management in *AIX Version 4.3 Guide to Printers and Printing*.

Printer−Specific Information in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Support in *AIX Version 4.3 Guide to Printers and Printing*.

Spooler Overview for System Management in *AIX Version 4.3 Guide to Printers and Printing*.

Virtual Printer Definitions and Attribute in *AIX Version 4.3 Guide to Printers and Printing*.

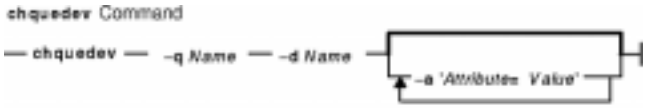
Printer Colon File Conventions in *AIX Version 4.3 Guide to Printers and Printing*.

chquedev Command

Purpose

Changes the printer or plotter queue device names.

Syntax



chquedev −**q***Name*−**d***Name* [−**a**'*Attribute* = *Value*'...]

Description

The **chquedev** command changes the printer or plotter queue device names by changing the device stanza in the **qconfig** file specified by the −**d**, and −**q** flags. Within that stanza, each attribute that matches one of the '*Attribute* = *Value*' flags given on the command line is replaced by the one entered on the command line. If no match is found, '*Attribute* = *Value*' is added to the end of the stanza.

You can use the Web−based System Manager Printer Queues application (**wsm printers** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chquedev** fast path to run this command.

> **Note:** Do not edit the **qconfig** file while there are active jobs in any queue. Editing includes both manual editing and use of the **mkque**, **rmque**, **chque**, **mkquedev**, **rmquedev**, or **chquedev** commands. It is recommended that all changes to the **qconfig** file be made using these commands. However, if manual editing is desired, first issue the **enq −G** command to bring the queuing system and the **qdaemon** to a halt after all jobs are processed. Then edit the **qconfig** file and restart the **qdaemon** with the new configuration.

Flags

−**a**'*Attribute* = *Value*'	Specifies the stanza lines to change or add. For a list of valid attributes, see the **qconfig** file.
−**d***Name*	Specifies the device *Name* in the queue to be changed.
−**q***Name*	Specifies the queue *Name* in which to change the device stanza.

Examples

To change the `ps` device stanza on the `lp0` queue to contain the line `backend = 'piobe −x −y'`, enter:

```
chquedev −qlp0 −d ps −a backend        =        'piobe −x −y'
```

> **Note:** The −**x** flag and the −**y** flag in this example are flags for the **piobe** command.

Files

/usr/bin/chquedev Contains the chquedev command.
/etc/qconfig Contains the configuration file.

Related Information

The **chque** command, **lsquedev** command, **mkquedev** command, **rmquedev** command, **piobe** command.

The **qconfig** file.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

Changing / Showing Queue Characteristics in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Overview for System Management in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Specific Information in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Support in *AIX Version 4.3 Guide to Printers and Printing*.

Spooler Overview for System Management in *AIX Version 4.3 Guide to Printers and Printing.*

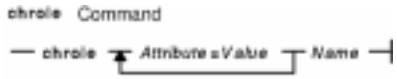
Printer Colon File Conventions in *AIX Version 4.3 Guide to Printers and Printing*.

chrole Command

Purpose

Changes role attributes. This command applies only to AIX Version 4.2.1 and later.

Syntax



chrole*Attribute*=*Value ... Name*

Description

The **chrole** command changes attributes for the role identified by the *Name* parameter. The role name must already exist. To change an attribute, specify the attribute name and the new value with the *Attribute*=*Value* parameter.

If you specify a single incorrect attribute or attribute value with the **chrole** command, the command does not change any attribute. You can use the Web−based System Manager Users application (**wsm users;** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chrole** fast path to run this command.

Restrictions on Modifying Roles

To ensure the integrity of the role information, only users with the **RoleAdmin** authorization can modify the attributes of a role.

Attributes

If you have the proper authority, you can set the following user attributes:

authorizations List of additional authorizations required for this role beyond those defined by the roles in the **rolelist** attribute. The *Value* parameter is a list of authorization names, separated by commas.

groups List of groups to which a user should belong, in order to effectively use this role. This attribute is for information only and does not automatically make the user a member of the list of groups. The *Value* parameter is a list of group names, separated by commas.

msgcat Contains a message catalog number for referencing the **msgnum** attribute. The *Value* parameter is an integer.

msgnum Contains the index into a message catalog for a description of the role. The *Value* parameter is an integer.

rolelist Lists the roles implied by this role. The *Value* parameter is a list of role names, separated by commas.

screens Lists the SMIT screen identifiers allowing roles to be mapped to various SMIT screens. The *Value* parameter is a list of SMIT screen identifiers, separated by commas.

visibility Specifies the role's visibility status to the system. The *Value* parameter is an integer. Possible

values are:

1 The role is enabled, displayed, and selectable. Authorizations contained in this role are applied to the user. If the attribute does not exist or has no value, the default value is 1.

0 The role is enabled and displayed as existing, but *not* selectable through a visual interface. Authorizations contained in this role are applied to the user.

−1 The role is disabled. Authorizations contained in this role are *not* applied to the user.

Security

Files Accessed:

Mode	File
rw	**/etc/security/roles**
r	**/etc/security/user.roles**

Auditing Events:

Event	Information
ROLE_Change	role, attribute

Examples

1. To change the authorizations of the role `ManageUserBasic` to **PasswdAdmin**, enter:

   ```
   chrole authorizations=PasswdAdmin ManageUserBasic
   ```

Files

/etc/security/roles	Contains the attributes of roles.
/etc/security/user.roles	Contains the role attribute of users.

Related Information

The **lsrole** command, **mkrole** command, **rmrole** command, **chuser** command, **lsuser** command, **mkuser** command.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

Security Administration in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

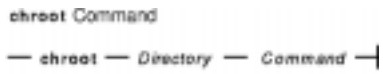
Administrative Roles Overview in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

chroot Command

Purpose

Changes the root directory of a command.

Syntax



chroot *DirectoryCommand*

Description

> **Attention:** If special files in the new root directory have different major and minor device numbers than the real root directory, it is possible to overwrite the file system.

The **chroot** command can be used only by a user operating with root user authority. If you have root user authority, the **chroot** command changes the root directory to the directory specified by the *Directory* parameter when performing the *Command*. The first / (slash) in any path name changes to *Directory* for the specified *Command* and any of its children.

The *Directory* path name is always relative to the current root. Even if the **chroot** command is in effect, the *Directory* path name is relative to the current root of the running process.

A majority of programs may not operate properly after the **chroot** command runs. For example, the commands that use the shared libraries are unsuccessful if the shared libraries are not in the new root file system. The most commonly used shared library is the **/usr/ccs/lib/libc.a** library.

The **ls −l** command is unsuccessful in giving user and group names if the current root location makes the **/etc/passwd** file beyond reach. In addition, utilities that depend on localized files (**/usr/lib/nls/***) may also be unsuccessful if these files are 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.

Parameters

Command Specifies a command to run with the **chroot** command.

Directory Specifies the new root directory.

Examples

> **Attention:** The commands in the following examples may depend on shared libraries. Ensure that the shared libraries are in the new root file system before you run the **chroot** command.

1. To run the **pwd** command with the **/usr/bin** directory as the root file system, enter:

   ```
   mkdir /usr/bin/lib
   ```

```
cp /usr/ccs/lib/libc.a /usr/bin/lib

chroot /usr/bin pwd
```

2. To run a Korn shell subshell with another file system as the root file system, enter:

   ```
   chroot /var/tmp /usr/bin/ksh
   ```

   This makes the directory name / (slash) refer to the `/var/tmp` for the duration of the `/usr/bin/ksh` command. It also makes the original root file system inaccessible. The file system on the `/var/tmp` file must contain the standard directories of a root file system. In particular, the shell looks for commands in the `/bin` and `/usr/bin` files on the `/var/tmp` file system.

   Running the `/usr/bin/ksh` command creates a subshell that runs as a separate process from your original shell. Press the END OF FILE (Ctrl−d) key sequence 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).

3. To create a file relative to the original root, not the new one, enter:

   ```
   chroot directory Command > file
   ```

Files

/etc/passwd	Specifies file that contains basic user attributes.
/usr/ccs/lib/libc.a	Specifies the standard I/O library and the standard C library.
/usr/ccs/lib/libcurses.a	Specifies the curses library.
/usr/lib/liblvm.a	Specifies the LVM (Logical Volume Manager) library.
/usr/ccs/lib/libm.a	Specifies the math library.
/usr/lib/libodm.a	Specifies the ODM (Object Data Manager) library.
/usr/sbin/chroot	Contains the **chroot** command.

Related Information

The **ksh** command, **ls** command.

The **chdir** subroutine, **chroot** subroutine.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains file system types, management, structure, and maintenance.

chsec Command

Purpose

Changes the attributes in the security stanza files.

Syntax

chsec [**−f***File*] [**−s** *Stanza*] [**−a** *Attribute* = *Value* ...]

Description

The **chsec** command changes the attributes stored in the security configuration stanza files. These security configuration stanza files have attributes that you can specify with the *Attribute* = *Value* parameter:

- **/etc/security/environ**
- **/etc/security/group**
- **/etc/security/lastlog**
- **/etc/security/limits**
- **/etc/security/login.cfg**
- **/usr/lib/security/mkuser.default**
- **/etc/security/passwd**
- **/etc/security/portlog**
- **/etc/security/user**

When modifying attributes in the **/etc/security/environ**, **/etc/security/lastlog**, **/etc/security/limits**, **/etc/security/passwd**, and **/etc/security/user** files, the stanza name specified by the *Stanza* parameter must either be a valid user name or `default`. When modifying attributes in the **/etc/security/group** file, the stanza name specified by the *Stanza* parameter must either be a valid group name or `default`. When modifying attributes in the **/usr/lib/security/mkuser.default** file, the *Stanza* parameter must be either `admin` or `user`. When modifying attributes in the **/etc/security/portlog** file, the *Stanza* parameter must be a valid port name. When modifying attributes in the **/etc/security/login.cfg** file, the *Stanza* parameter must either be a valid port name, a method name, or the **usw** attribute.

When modifying attributes in the **/etc/security/login.cfg** or **/etc/security/portlog** file in a stanza that does not already exist, the stanza is automatically created by the **chsec** command.

You cannot modify the **password** attribute of the **/etc/security/passwd** file using the **chsec** command. Instead, use the **passwd** command.

Only the root user or a user with an appropriate authorization can change administrative attributes. For example, to modify administrative group data, the user must be root or have GroupAdmin authorization.

Flags

−a*Attribute* = *Value* Specifies the attribute to modify and the new value for that attribute. If you do not specify the value, the attribute is removed from the given stanza.

−f *File*	Specifies the name of the stanza file to modify.
−s*Stanza*	Specifies the name of the stanza to modify.

Security

Access Control: This command grants execute access only to the root user and the security group. The command has the trusted computing base attribute and runs the **setuid** command to allow the root user to access the security databases.

Files Accessed:

Mode	File
rw	**/etc/security/environ**
rw	**/etc/security/group**
rw	**/etc/security/lastlog**
rw	**/etc/security/limits**
rw	**/etc/security/login.cfg**
rw	**/usr/lib/security/mkuser.default**
rw	**/etc/security/passwd**
rw	**/etc/security/portlog**
rw	**/etc/security/user**

Auditing Events:

Event	Information
USER_Change	user name, attribute
GROUP_Change	group name, attribute
PORT_Change	port, attribute

Examples

1. To change the **/dev/tty0** port to automatically lock if 5 unsuccessful login attempts occur within 60 seconds, enter:

   ```
   chsec −f /etc/security/login.cfg −s /dev/tty0 −a logindisable=5 −a logininterval=60
   ```

2. To unlock the **/dev/tty0** port after it has been locked by the system, enter:

   ```
   chsec −f /etc/security/portlog −s /dev/tty0 −a locktime=0
   ```

3. To allow logins from 8:00 a.m. until 5:00 p.m. for all users, enter:

   ```
   chsec −f /etc/security/user −s default −a logintimes=:0800−1700
   ```

4. To change the CPU time limit of user `joe` to 1 hour (3600 seconds), enter:

   ```
   chsec −f /etc/security/limits −s joe −a cpu=3600
   ```

Files

/usr/bin/chsec	Specifies the path to the **chsec** command.
/etc/security/environ	Contains the environment attributes of users.
/etc/security/group	Contains extended attributes of groups.

/etc/security/lastlog	Defines the last login attributes for users.
/etc/security/limits	Defines resource quotas and limits for each user.
/etc/security/login.cfg	Contains port configuration information.
/usr/lib/security/mkuser.default	Contains the default values for new users.
/etc/security/passwd	Contains password information.
/etc/security/portlog	Contains unsuccessful login attempt information for each port.
/etc/security/user	Contains the extended attributes of users.

Related Information

The **chgroup** command, **chuser** command, **grpck** command, **login** command, **lsgroup** command, **lssec** command, **lsuser** command, **mkgroup** command, **mkuser** command, **passwd** command, **pwdck** command, **rmgroup** command, **rmuser** command, **su** command, **usrck** command.

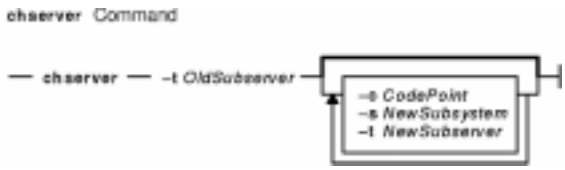
The **getgroupattr** subroutine, **getportattr** subroutine, **getuserattr** subroutine, **getuserpw** subroutine, **putgroupattr** subroutine, **putportattr** subroutine, **putuserattr** subroutine, **putuserpw** subroutine.

chserver Command

Purpose

Changes a subserver definition in the subserver object class.

Syntax



chserver−**t***OldSubserver* [−**c***CodePoint*] [−**s***NewSubsystem*] [−**t***NewSubserver*]

Description

The **chserver** command modifies an existing subserver definition in the subserver object class. It can change subserver types, the owning subsystem, or the subserver code point.

Flags

−**c***CodePoint* Specifies the *CodePoint* integer that identifies the subserver. This is the value used by the subsystem to recognize the subserver. The **chserver** command is unsuccessful if the *CodePoint* already exists for the existing subsystem name and no new subsystem name is entered. It is also unsuccessful if the *NewSubsystem* name and subserver *CodePoint* exist in the subserver object class. The limit for the *CodePoint* storage is the same as a short integer (1 through 32,768).

−**s***NewSubsystem* Specifies the name that uniquely identifies the *NewSubsystem* to the subserver it belongs to. The **chserver** command is unsuccessful if one of the following occurs:

- The *NewSubsystem* name is not known in the subsystem object class.
- The *NewSubsystem* name is known in the subsystem object class but uses signals as its communication method.
- The *NewSubsystem* name already exists with the existing subserver *CodePoint* value in the Subserver Type object class, and no subserver *CodePoint* value is entered.
- A new subserver *CodePoint* is entered, with the *NewSubsystem* name and subserver *CodePoint* already existing in the Subserver Type object class.

−**t***NewSubserver* Specifies the name that uniquely identifies the *NewSubserver*. The **chserver** command is unsuccessful if the *NewSubserver* type is already known in the subserver object class.

−**t***OldSubserver* Specifies the name that uniquely identifies the existing subserver. The **chserver** command is unsuccessful if the *OldSubserver* type is not known in the subserver object class.

Security

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the **chserver** command will generate the following audit record (event) every time the command is executed:

Event	Information

SRC_Chserver Lists in an audit log the name of the subsystem and the fields that have been changed.

See "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more details about how to properly select and group audit events, and how to configure audit event data collection.

Examples

1. To change the subserver type, enter:

   ```
   chserver -t old -t new
   ```

   This changes the subserver type from the `old` subserver type to the `new` subserver type.

2. To change the owning subsystem, enter:

   ```
   chserver -t old -s srctest
   ```

   This changes the owning subsystem to `srctest`.

3. To change the subserver type, subsystem, and subserver code point, enter:

   ```
   chserver -t old -t new -s srctest -c 1234
   ```

This changes the subserver type from the `old` to the `new` subserver type, the owning subsystem to `srctest`, and the subserver code point to `1234`.

Files

/etc/objrepos/SRCsubsys Specifies the SRC Subsystem Configuration object class.
/etc/objrepos/SRCsubsvr Specifies the SRC Subserver Configuration object class.

Related Information

The **auditpr** command, **mkserver** command, **rmserver** command, **startsrc** command, **stopsrc** command, **traceson** command, **tracesoff** command.

Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Defining Your Subsystem to the SRC in *AIX General Programming Concepts: Writing and Debugging Programs*.

System Resource Controller (SRC) Overview for Programmers in *AIX General Programming Concepts: Writing and Debugging Programs*.

chservices Command

Purpose

Changes the contents of the **/etc/services** file.

Syntax

To Add or Activate an Entry:

chservices [**−a**] **−v***ServiceName***−p***protocol***−n***port* [**−u**"*Alias ...*"]

To Change an Entry:

chservices−c−v*ServiceName***−p***protocol***−n***port* [**−V** *NewServiceName*] [**−P** *NewProtocol*]
[**−N** *NewPort*] [**−u**"*Alias ...*"]

To Deactivate an Entry:

chservices−d−v*ServiceName***−p***protocol***−n***port* [**−V***NewServiceName*] [**−u** *Alias ...*"]

Description

The **chservices** command adds, deletes, or changes entries in the **/etc/services** file. These entries are related to known services used in the DARPA Internet and also related to information used by the **inetd** server. The entries for the **inetd** server determine how the system handles Internet service requests.

The **chservices** command manipulates the following entries for known services:

- The official Internet service name specified by the *ServiceName* variable.
- The port number, specified by the *port* variable, used for the service.
- The transport protocol, specified by the *protocol* variable, used for the service.
- A list of unofficial names, specified by the *Alias* variable, used by the service.

Flags

−a	Adds or activates an entry in the **/etc/services** file. If the requested service exists in the file, the **−a** flag uncomments the line. If the line does not exist, the **−a** flag adds the line to the file. This is the default action.
−c	Changes an entry in the **/etc/services** file.
−d	Deactivates an entry in the **/etc/services** file by commenting the line in the file.
−N *NewPort*	Specifies a socket port number.
−n *port*	Specifies a socket port number.
−P *NewProtocol*	Specifies a new protocol name for a current protocol name.
−p *protocol*	Specifies the protocol.
−V *NewName*	Specifies a new service name.
−v *ServiceName*	Specifies the service name.
−u "*Alias...*"	Specifies a list of aliases.

Note: Adding or keeping comments on lines modified with the **chservices** command is not supported.

Security

Access Control: Only the root user and members of the system group have access to this command.

Examples

1. To add the service, `gregsapp`, as a `udp` service on port 1423, enter:

   ```
   chservices -a -v gregsapp -p udp -n 1423
   ```

2. To add the service, `gregsapp`, as a `udp` service on port 1423 with an alias of `fredsapp`, enter:

   ```
   chservices -a -v gregsapp -p udp -n 1423 -u
   "fredsapp"
   ```

3. To change the port of the service specified as `gregsapp` with a `udp` protocol to 1456, enter:

   ```
   chservices -c -v gregsapp -p udp -N 1456
   ```

4. To deactivate the `gregsapp` service on `udp` port 1456 by commenting it out, enter:

   ```
   chservices -d -v gregsapp -p udp -n 1456
   ```

Files

/usr/sbin/chservices Contains the **chservices** command.
/etc/services Contains services information for the **inetd** daemon.

Related Information

The **chsubserver** command.

The **inetd** daemon, **fingerd** daemon, **ftpd** daemon, **rexecd** daemon, **rlogind** daemon, **rshd** daemon, **syslogd** daemon, **talkd** daemon, **telnetd** daemon, **tftpd** daemon.

The **inetd.conf** file format, **protocols** file format, **services** file format.

The SMIT Interface for TCP/IP in *AIX Version 4.3 System Management Guide: Communications and Networks*.

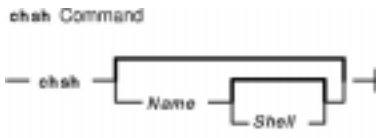
TCP/IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

chsh Command

Purpose

Changes a user's login shell.

Syntax



chsh [*Name* [*Shell*]]

Description

The **chsh** command changes a user's login **shell** attribute. The **shell** attribute defines the initial program that runs after a user logs in to the system. This attribute is specified in the **/etc/passwd** file. By default, the **chsh** command changes the login shell for the user who gives the command.

The **chsh** command is interactive. When you run the **chsh** command, the system displays a list of the available shells and the current value of the **shell** attribute. Then, the system prompts you to change the shell. You must enter the full path name of an available shell.

If you have execute permission for the **chuser** command, you can change the login shell for another user. To change the login shell for another user, specify a *Name* parameter. Valid shells are defined in the usw stanza of the **/etc/security/login.cfg** file. The default list of valid shells is: **/usr/bin/ksh**, **/usr/bin/sh**, **/usr/bin/bsh**, **/usr/bin/csh** but your system manager may have defined more.

Security

Access Control: All users should have execute (x) access to this command since the program enforces its own access policy. This command should be installed as a program in the trusted computing base (TCB). The command should be owned by the **security** group with the **setgid** (SGID) bit set.

Files Accessed:

Mode	File
x	**/usr/bin/chuser**
r	**/etc/security/login.cfg**
rw	**/etc/passwd**

Examples

1. To change the shell that runs after you log in to the system, enter:

   ```
   chsh
   ```

   Information similar to the following appears:

```
current available shells:
 /usr/bin/sh
 /usr/bin/bsh
 /usr/bin/csh
 /usr/bin/ksh:
current login shell:
 /usr/bin/ksh
change (y/n)? >
```

Indicate that a change should be made by entering `y` after the change `(y/n)?` prompt. Then, add the name of the shell you want when the `to?` prompt appears, as in the following example:

```
change (y/n)? > y
to? > /usr/bin/csh
```

The next time you log in, the **/usr/bin/csh** shell appears.

2. To change the shell to `/usr/bin/ksh` for `kim`, enter:

```
chsh kim /usr/bin/ksh
```

Files

/usr/bin/chsh	Specifies the path to the **chsh** command.
/usr/bin/chuser	Changes user information.
/etc/passwd	Contains the basic user attributes.
/etc/security/login.cfg	Contains login configuration information.

Related Information

The **chgroup** command, **chgrpmem** command, **chuser** command, **lsgroup** command, **lsuser** command, **mkgroup** command, **mkuser** command, **passwd** command, **pwdadm** command, **rmgroup** command, **rmuser** command, **setgroups** command, **setsenv** command.

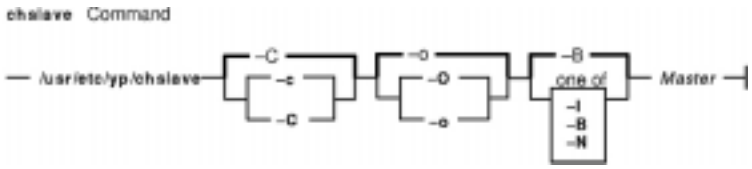
Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices* describes the identification and authentication of users, discretionary access control, the trusted computing base, and auditing.

chslave Command

Purpose

Re−executes the **ypinit** command to retrieve maps from a master server and re−starts the **ypserv** daemon to change the slave server.

Syntax



/usr/etc/yp/chslave [**−C** |**−c**] [**−O** | **−o**] [**−I** | **−B** | **−N**] *Master*

Description

The **chslave** command re−invokes the **ypinit** command to retrieve maps from the master server you specify on the command line. The **ypserv** daemon is re−started after the **ypinit** command has completed successfully. The *Master* parameter specifies the host name of the master server. The master server specified can be the master server currently in use or a new master server that is configured and running.

You can use the Web−based System Manager Network application (**wsm network** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chslave** fast path to run this command.

Flags

−B Invokes the **ypinit** command and starts the **ypserv** daemon. If the **ypserv** daemon is already running, this flag will cause the **ypinit** command to kill the daemon and then restart it. This flag is the default.

−C Invokes the **ypinit** command with the **−n** flag. The **chslave** command continues on errors. This flag is the default.

−c Stops execution when errors occur.

−I Executes the **ypinit** command immediately but does not start or restart the **ypserv** daemon.

−O Overwrites any maps that exist in the domain.

−o Prevents the overwrite of maps that exist in the domain. This flag is the default.

−N Invokes the **ypinit** command and restarts the **ypserv** daemon.

Examples

To retrieve maps from the master server named `host91`, enter:

```
chslave −O −B host91
```

This will overwrite any existing maps for the current domain.

Files

/etc/rc.nfs	Contains the startup script for NFS and NIS daemons.
/var/yp/domainname	Contains the NIS maps for the NIS domain.

Related Information

The **chmaster** command, **mkclient** command, **mkslave** command, **rmyp** command, **smit** command, **ypinit** command.

The **ypbind** daemon, **yppasswdd** daemon, **ypserv** daemon, **ypupdated** daemon.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Network Information Service (NIS) in *AIX Version 4.3 Network Information Services (NIS and NIS+) Guide*.

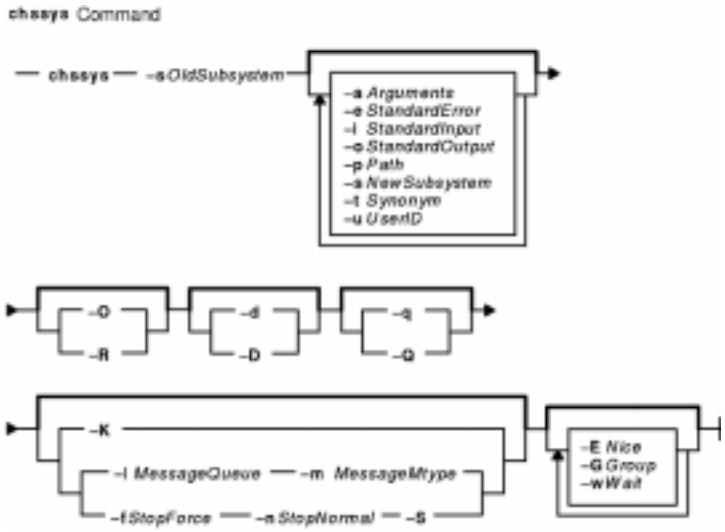
NIS Reference.

chssys Command

Purpose

Changes a subsystem definition in the subsystem object class.

Syntax



chssys−**s***OldSubsystem* [−**a***Arguments*] [−**e***StandardError*] [−**i***StandardInput*] [−**o** *StandardOutput*] [−**p***Path*] [−**s***NewSubsystem*] [−**t***Synonym*] [−**u***UserID*] [−**O** | −**R**] [−**d** | −**D**] [−**q** | −**Q**] [−**K** | [−**I***MessageQueue*−**m***MessageMtype* | −**f** *StopForce*−**n***StopNormal*−**S**] [−**E***Nice*] [−**G***Group*] [−**w***Wait*]

Description

The **chssys** command modifies an existing subsystem definition in the subsystem object class. If a new subsystem name is entered, the Subserver Type object class and the Notify object class are modified to reflect the new subsystem name.

> **Note:** Any auditing performed by the System Resource Controller (SRC) when actions are taken for the subsystem is logged against the login ID of the user who created the subsystem by using the **mkssys** command. For example, if you are logged in with root user authority, the subsystem is added with root user authority as the audit account.

Flags

−**a***Arguments*	Specifies any arguments that must be passed to the program executed as the subsystem. These command *Arguments* are passed by the SRC to the subsystem according to the same rules used by the shell. Quoted strings are passed as a single argument, and blanks outside a quoted string delimit arguments. Single and double quotes can be used.
−**d**	Specifies that an inactive subsystem is displayed when the **lssrc**−**a** command request (status all) or the **lssrc**−**g** command request (status group) is made.
−**D**	Specifies that an inactive subsystem is not displayed when status all or status group requests are made.
−**e***StandardError*	Specifies where the subsystem standard error data is placed.

−E*Nice*	Specifies the *Nice* value. The *Nice* parameter changes the execution priority of the subsystem. The valid values are 0 through 39 (ordinary *Nice* values mapped to all positive numbers). If the **−E** flag is not present, the subsystem priority defaults to 20. Values between 0 and 19 are reserved for users with root authority.
−f*StopForce*	Specifies the signal sent to the subsystem when a forced stop of the subsystem is requested. Use only when the subsystem uses signals for communication. The **chssys** command is unsuccessful if the *StopForce* parameter specifies an invalid signal. The **−n** and **−S** flags must follow this flag.
−G*Group*	Specifies that the subsystem belongs to the group specified by the *Group* parameter and responds to all group actions on the group.
−i*StandardInput*	Specifies where the subsystem *StandardInput* is routed. This field is ignored when the subsystem uses sockets for communication.
−K	Specifies that the subsystem uses sockets as its communication method.
−I*MessageQueue*	Specifies that the subsystem uses message queues as its communication method. The *MessageQueue* parameter specifies the message queue key for creating the message queue for the subsystem. Use the **ftok** subroutine with the subsystem path name as input to generate a unique key. The **−m** flag must follow this flag.
−m*MessageMtype*	Specifies the *MessageMtype* key that the subsystem expects on packets sent to the subsystem by the SRC. Use only when the subsystem uses message queues for communication. The *MessageMtype* must be greater than 0. This flag must be preceded by the **−l** flag.
−n*StopNormal*	Specifies the signal sent to the subsystem when a normal stop of the subsystem is requested. Use only when the subsystem uses signals for communication. The **chssys** command is unsuccessful if the *StopNormal* parameter specifies an invalid signal. This flag must be preceded by the **−f** flag and followed by the **−S** flag.
−o*StandardOutput*	Specifies where the subsystem *StandardOutput* is placed.
−O	Specifies that the subsystem is not restarted if it stops abnormally.
−p*Path*	Specifies the absolute *Path* to the subsystem program.
−q	Specifies that the subsystem can have multiple instances running at the same time.
−Q	Specifies that multiple instances of the subsystem are not allowed to run at the same time.
−R	Specifies that the subsystem is restarted if it stops abnormally.
−s*NewSubsystem*	Specifies the new name that uniquely identifies the subsystem. Any subservers or notify methods defined for the old subsystem's name are redefined for the *NewSubsystem* name. The **chssys** command is unsuccessful if the *NewSubsystem* name is already known in the subsystem object class.
−s*OldSubsystem*	Specifies the current name that uniquely identifies the subsystem. The **chssys** command is unsuccessful if the *OldSubsystem* name is not known in the subsystem object class.
−S	Specifies that the subsystem uses signals as its communication method. You cannot define subservers for the subsystem name when your communication method is signals. If a subserver is defined for the subsystem, the subserver definitions are deleted from the subserver object class. This flag must be preceded by the **−f** and **−n** flags.
−t*Synonym*	Specifies an alternate name for the subsystem. The **chssys** command is unsuccessful if the *Synonym* name is already known in the subsystem object class.
−u*UserID*	Specifies the user ID for the subsystem. The *UserID* that creates the subsystem is used for security auditing of that subsystem.
−w*Wait*	Specifies the time, in seconds, allowed to elapse between a stop cancel (**SIGTERM**) signal and a subsequent **SIGKILL** signal. Also used as the time limit for restart actions. If the subsystem stops abnormally more than twice in the time limit specified by the *Wait* value, it is not automatically restarted.

Security

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the **chssys** command will generate the following audit record (event) every time the command is executed:

Event	Information
SRC_Chssys	Lists in an audit log the name of the subsystem and the fields that have been changed.

See "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for details about properly selecting and grouping audit events, and configuring audit event data collection.

Examples

1. To change the subsystem name, enter:

   ```
   chssys −s srctest −s inetd
   ```

   This changes the subsystem name from `srctest` to `inetd`.

2. To change the communication type to sockets, enter:

   ```
   chssys −s srctest −K
   ```

   This changes the communication type for the subsystem to sockets.

3. To change the communication type to message queues, enter:

   ```
   chssys −s srctest −l 123456 −m 789
   ```

   This changes the communication type for the subsystem to message queues, with a message queue key of `123456` and a subsystem message type of `789`.

4. To change the communication type to signals, enter:

   ```
   chssys −s srctest −S −n 30 −f 31
   ```

   This changes the communication type for the subsystem to signals, with a normal stop signal of `30` and a force stop signal of `31`.

5. To change the command arguments, enter:

   ```
   chssys −s srctest −a "−a 123 −b \"4 5 6\" −c '7 8 9'"
   ```

This places `−a` as the first argument, `123` as the second, `−b` as the third, `4 5 6` as the fourth, `−c` as the fifth, and `7 8 9` as the sixth argument to the `srctest` subsystem.

Files

/etc/objrepos/SRCsubsys	Specifies the SRC Subsystem Configuration object class.
/etc/objrepos/SRCsubsvr	Specifies the SRC Subserver Configuration object class.
/etc/objrepos/SRCnotify	Specifies the SRC Notify Method object class.
/dev/SRC	Specifies the **AF_UNIX** socket file.
/dev/.SRC−unix	Specifies the location for temporary socket files.

Related Information

The **auditpr** command, **lssrc** command, **mkssys** command, **rmssys** command.

Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

System Resource Controller Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Defining Your Subsystem to the SRC in *AIX General Programming Concepts: Writing and Debugging Programs*.

System Resource Controller (SRC) Overview for Programmers in *AIX General Programming Concepts: Writing and Debugging Programs*.

chsubserver Command

Purpose

Changes the contents of the /etc/inetd.conf file or similar system configuration file.

Syntax

To Add or Activate a Server or Subserver Entry:

chsubserver [**−a**] **−v***ServiceName***−p***protocol* [**−t***socket_type*][**−w***WaitIndicator*] [**−u***user*] [**−g***program*] [**−r***server*] [**−C***ConfigFile*] [*program*] [*args*]

To Change a Server Entry:

chsubserver−c−v*ServiceName***−p***protocol* [**−t***SocketType*] [**−w***WaitIndicator*] [**−u***user*] [**−g***program*] [**−V***NewServiceName*] [**−P***NewProtocol*] [**−T***NewSocketType*] [**−W***NewWaitIndicator*] [**−U***NewUser*] [**−G***NewProgram*] [**−r***server*] [**−C***ConfigFile*] [*program*] [*args*]

To Deactivate a Server Entry or an inetd Subserver Entry:

chsubserver−d−v*ServiceName***−p***protocol* [**−t***SocketType*] [**−w***WaitIndicator*] [**−u***user*] [**−g***program*] [**−r***server*] [**−C***ConfigFile*] [*program*] [*args*]

Description

The **chsubserver** command adds, deletes, or changes entries in the **/etc/inetd.conf** system configuration file, which is the default, or a similar configuration file. These entries are related to known services used in the DARPA Internet and also related to information used by the **inetd** server. The entries for the **inetd** server determine how the system handles Internet service requests.

The **chsubserver** command also allows the user to refresh a server using the **−r** flag. The server specified is sent a **SIGHUP** signal to reread its configuration file. This allows you to edit the configuration file and have the changes take effect immediately.

Each service entry contains information about known services and information used by the **inetd** server. The **chsubserver** command manipulates the following entries for known services and for **inetd** server or other subserver information:

- The official Internet service name specified by the *ServiceName* variable.
- The transport protocol, specified by the *protocol* variable, used for the service.
- The type of socket, specified by the *SocketType* variable, associated with the service. The socket types associated with a service can be stream sockets or datagram sockets. Use only the **nowait** flag with stream sockets. Use either the **wait** or **nowait** flag with datagram sockets.
- A **wait** or **nowait** flag, specified by the *WaitIndicator* variable. The **wait** or **nowait** flag indicates whether the **inetd** server waits for a datagram server to release the socket before continuing to listen at the socket.
- The user name, specified by the *user* variable, that the **inetd** server uses to start a subserver.

You can use the Web−based System Manager System application (**wsm system** fast path) run this command. You could also use the System Management Interface Tool (SMIT) **smit inetdconf** fast path to run this command.

Flags

−a	Adds or activates an entry in the configuration file. If the requested service exists in the configuration file, the **−a** flag uncomments the line. If the line does not exist, the **−a** flag adds the line to the configuration file. This is the default action.
−c	Changes an entry in the configuration file.
−C	Specifies a configuration file similar to **/etc/inetd.conf**.
−d	Deactivates an entry in the configuration file by commenting the line in the file.
−G *NewProgram*	Replaces the existing program to start.
−g *Program*	Specifies the program to start..
−P *NewProtocol*	Specifies a new protocol name for a current protocol name.
−p *protocol*	Specifies the protocol.
−r *server*	Sends a **SIGHUP** to the specified server.
−T *NewSocketType*	Replaces the existing type of socket, either a value of **stream** for stream sockets or a value of **dgram** for datagram sockets.
−t *SocketType*	Specifies a type of socket, either a value of **stream** for stream sockets or a value of **dgram** for datagram sockets.
−U *NewUser*	Replaces the existing user name.
−u *user*	Specifies a user name.
−V *NewName*	Specifies a new service name.
−v *ServiceName*	Specifies the service name.
−W *NewWaitIndicator*	Replaces the existing *WaitIndicator*.
−w *WaitIndicator*	Specifies either single−thread service with a value of **wait** or multithread service with a value of **nowait**.

Security

Access Control: Only the root user and members of the system group have access to this command.

Examples

1. To uncomment the uucp line in the /etc/inetd.conf file, enter:

   ```
   chsubserver −a −v uucp −p tcp
   ```

2. To add a line to the **/etc/inetd.conf** file that describes the gregserv service and runs the program `/usr/sbin/gregserv` as root over the udp protocol with stream sockets and arguments of ftpd, enter in one line:

   ```
   chsubserver −a −r inetd −v gregserv −p udp −t stream −w nowait −u
   root −g /usr/sbin/gregserv ftpd
   ```

   The **inetd** does not wait for confirmation. After adding the line to the file, the **inetd** program will be sent a **SIGHUP** signal.

3. To change the existing service from using stream sockets to using dgram sockets in the **/tmp/inetd.conf** file, enter in one line:

   ```
   chsubserver −c −v gregserv −p udp −t stream −T dgram −C /tmp/inetd.conf
   ```

4. To comment the gregserv service over udp in the **/etc/inetd.conf** file, enter:

   ```
   chsubserver −d −v gregserv −p udp
   ```

Files

/usr/sbin/chsubserver	Contains the **chsubserver** command.
/etc/inetd.conf	Contains configuration information for the **inetd** daemon.

Related Information

The **chservices** command.

The **inetd** daemon, **fingerd** daemon, **ftpd** daemon, **rexecd** daemon, **rlogind** daemon, **rshd** daemon, **syslogd** daemon, **talkd** daemon, **telnetd** daemon, **tftpd** daemon.

The **inetd.conf** file format, **protocols** file format, **services** file format.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

The SMIT Interface for TCP/IP in *AIX Version 4.3 System Management Guide: Communications and Networks*.

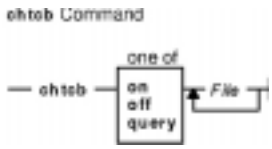
TCP /IP Daemons in *AIX Version 4.3 System Management Guide: Communications and Networks*.

chtcb Command

Purpose

Changes or queries the **trusted computing base** attribute of a file.

Syntax



chtcb { **on** | **off** | **query** } *File* ...

Description

The **chtcb** command changes or queries the **trusted computing base** (TCB) attribute of the files you specify with the *File* parameter. The following alternatives are valid:

on	Enables the **trusted computing base** attribute.
off	Disables the **trusted computing base** attribute, if set.
query	Displays the value of the **trusted computing base** attribute.

This command should be executed on the trusted path.

Security

Access Control: This command should grant execute (x) access to the root user and members of the security group. The command should have the **trusted computing base** attribute.

Examples

1. To identify the `plans` file as part of the trusted computing base (TCB), set the **trusted computing base** attribute to the `on` value by entering the following:

   ```
   chtcb on plans
   ```

   The `plans` file now can be executed from the trusted path.

2. To query whether the `plans` file is part of the trusted computing base (TCB), enter:

   ```
   chtcb query plans
   ```

   When the status appears, you know that the `plans` file is part of the trusted computing base if the TCB attribute is set to the **on** value.

3. To remove the `plans` file from the trusted computing base (TCB), enter:

   ```
   chtcb off plans
   ```

Files

/usr/sbin/chtcb Contains the **chtcb** command.

Related Information

The **tsh** command, **tsm** command, **tvi** command.

The **chmod** subroutine.

For more information about the identification and authentication of users, discretionary access control, the trusted computing base, and auditing, refer to Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

chtun Command

Purpose

Changes a tunnel definition.

Syntax

chtun **−t***tunnel_ID* **−v{4|6}** [**−s***src_host_IP_address*] [**−d***dst_host_IP_address*] [**−m***pkt_mode*] [**−f***fw_address* [**−x***dst_mask*]] [**−e***src_esp_algo*] [**−a***src_ah_algo]*] [**−p***src_policy*] [**−E***dst_esp_algo*] [**−A***dst_ah_algo]*] [**−P***dst_policy*] [**−l***lifetime*] [**−k***src_esp_key*] [**−h***src_ah_key*] [**−K***dst_esp_key*] [**−H***dst_ah_key*] [**−r***refresh*] [**−i** {**y**|**n**}] [**−n***src_esp_spi*] [**−u***src_ah_spi*] [**−N***dst_esp_spi*] [**−U***dst_ah_spi*] [**−b***src_enc_mac_algo*] [**−c***src_enc_mac_key*] [**−B***dst_enc_mac_algo*] [**−C***dst_enc_mac_key*]

Description

Use the **chtun** command to change a definition of a tunnel between a local host and a tunnel partner host. If a flag is not specified, then the value given for the **gentun** command should stay the value for that field. It may also change the auto−generated filter rules created for the tunnel by the **gentun** command.

Flags

−A*dst_ah_algo]*	(**manual** tunnel only) Authentication algorithm, which is used by the destination for IP packet encryption. The valid values for **−A** depend on which authentication algorithms have been installed on the host. The list of all the authentication algorithms can be displayed by issuing the **ipsecstat −A** command. For IBM tunnels, this flag is invalid.
−a*src_ah_algo]*	Authentication algorithm, used by source host for IP packet authentication. The valid values for **−a** depend on which authentication algorithms have been installed on the host. The list of all authentication algorithms can be displayed by issuing the **ipsecstat −A** command. For IBM tunnels, this will apply to both inbound and outbound traffic in the tunnel.
−B*dst_enc_mac_algo*	(**manual** tunnel only) Destination ESP Authentication Algorithm (New header format only). The valid values for **−B** depend on which authentication algorithms have been installed on the host. The list of all the authentication algorithms can be displayed by issuing the **ipsecstat −A** command.
−b*src_enc_mac_algo*	(**manual** tunnel only) Source ESP Authentication Algorithm (New header format only). The valid values for **−b** depend on which authentication algorithms have been installed on the host. The list of all the authentication algorithms can be displayed by issuing the **ipsecstat −A** command.
−C*dst_enc_mac_key*	(**manual** tunnel only) Destination ESP Authentication Key (New header format only). It must be a hexadecimal string started with "0x".
−c*src_enc_mac_key*	(**manual** tunnel only) Source ESP Authentication Key (New header format only). It must be a hexdecimal string started with "0x".
−d*dst_host_IP_address*	Destination Host IP address. For a host−host tunnel, this value is the IP address of the destination host interface to be used by the tunnel. For a host−firewall−host tunnel, this is the IP address of a destination host behind the firewall. A host name is also valid and the first IP address returned by the name server for the host name will be used.
−E*dst_esp_algo*	(**manual** tunnel only) Encryption algorithm, which is used by the destination for IP

packet encryption. The valid values for **−E** depend on which encryption algorithms have been installed on the host. The list of all the encryption algorithms can be displayed by issuing the **ipsecstat −E** command. For IBM tunnels, this flag is invalid.

−e*src_esp_algo* Encryption algorithm, used by source host for IP packet encryption. The valid values for **−e** depend on which encryption algorithms have been installed on the host. The list of all encryption algorithms can be displayed by issuing the **ipsescstat −E** command. For IBM tunnels, this will apply to both inbound and outbound traffic in the tunnel.

−f*fw_address* IP address of the firewall that is between source and destination hosts. A tunnel will be established between the source and the firewall. Therefore the corresponding tunnel definition must be made in the firewall host. A host name can also be specified with this flag, and the first IP address returned by name server for the host name will be used.

The **−m** flag is forced to use default value (**tunnel**) if **−f** is specified.

−H*dst_ah_key* The Key String for destination AH. The input must be a hexdecimal string started with "0x".

−h*src_ah_key* The Key String for source AH. The input must be a hexdecimal string started with "0x".

−i (IBM tunnel only) Initiator Flag, identifies which partner starts the session negotiations. Specifying a value of **y** causes the local host to try to initiate a session with the target host. That session is used to run the session key exchange protocol. A value of **n** causes the local host to wait for the target host to initiate the session key exchange. If both partners are identified as the tunnel initiator, a deadlock resolution algorithm resolves the conflict. At least one of the partners must be set as the initiator in order for the tunnel to operate.

−K*dst_esp_key* The Key String for destination ESP. The input must be a hexdecimal string started with "0x".

−k*src_esp_key* The Key String for the source ESP. It is used by the source to create the tunnel as well as the session key if IBM tunnel is used. The input must be a hexdecimal string started with "0x".

−l*lifetime* Key Lifetime, specified in minutes.

For IBM tunnels, this value indicates the time a session key may be used. The value specified affects performance of the tunnel. For example, the smaller the value, the more often a new key is computed and exchanged with the tunnel partner. Generally, values used for CDMF should be smaller than those used for DES due to the strength of the encryption algorithms.

A new session key is automatically generated after every session key lifetime expires. The generated session keys are used by the encryption and authentication algorithms. The old and new keys are valid for an overlapped period of time determined by the Session Key Refresh Overlap Time. This is so that messages generated with the old key, which are in−transit in the network, can be decrypted or validated on arrival even after a new key computation. If the key lifetime is n minutes, both the old key and new key are valid during the last Refresh Overlap Time minutes of the n minutes.

The valid values for IBM tunnels are 1 − 1440.

For **manual** tunnels, the value of this flag indicates the time of operability before the tunnel expires.

	The valid values for **manual** tunnels are 0 − 44640. Value 0 indicates that the **manual** tunnel will never expire.
−m*pkt_mode*	Secure Packet Mode. This value must be specified as **tunnel** or **transport**.
−N*dst_esp_spi*	(**manual** tunnel only) Security Parameter Index for the destination ESP.
−n*src_esp_spi*	(**manual** tunnel only) Security Parameter Index for source ESP. This SPI and the destination IP address is used to determine which security association to use for ESP.
−P*dst_policy*	(**manual** tunnel only) Destination policy, identifies how the IP packet authentication and/or encryption is to be used by destination. If the value of this flag is specified as **ea**, the IP packet gets encrypted before authentication. If specified as **ae**, it gets encrypted after authentication, whereas specifying **e** or **a** alone corresponds to the IP packet being encrypted only or authenticated only. For IBM tunnels, this flag is invalid.
−p*src_policy*	Source policy, identifies how the IP packet authentication and/or encryption is to be used by source. If the value of this flag is specified as **ea**, the IP packet gets encrypted before authentication. If specified as **ae**, it gets encrypted after authentication, whereas specifying **e** or **a** alone corresponds to the IP packet being encrypted only or authenticated only. For IBM tunnels, this will apply to both inbound and outbound traffic in the tunnel.
−r*refresh*	(IBM tunnel only) Session Key Refresh Overlap Time, determines the amount of overlap time of the new key start and an old key expiration. The value specified will be the amount of time in minutes that a previous session key will still be valid after a key refresh has been done. The value specified can not be greater than the Key Lifetime. The valid values are 1 − 720.
−s*src_host_IP_address*	Source Host IP address, IP address of the local host interface to be used by the tunnel. A host name is also valid and the first IP address returned by name server for the host name will be used.
−t*tunnel_ID*	The tunnel identifier (ID), a locally unique, numeric identifier for a particular tunnel definition. The value must match an existing tunnel ID.
−U*dst_ah_spi*	(**manual** tunnel only) Security Parameter Index for the destination AH.
−u*src_ah_spi*	(**manual** tunnel only) Security Parameter Index for source AH. This SPI and the destination IP address is used to determine which security association to use for AH.
−v	The IP version for which the tunnel is created. For IP version 4 tunnels, use the value of **4**. For IP version 6 tunnels, use the value of **6**.
−x*dst_mask*	This flag is used for host−firewall−host tunnels. The value is the network mask for the secure network behind a firewall. The Destination host specified with the **−d** flag is a member of the secure network. The combination of the **−d** and **−x** flags allows source host communications with multiple hosts in the secure network through the source−firewall tunnel, which must be in tunnel Mode. This flag is valid only when **−f** is specified.
−y	(**manual** tunnel only) Replay prevention flag. Replay prevention is valid only when the ESP or AH header is using the new header format (see the **−z** flag). The valid values for the **−y** flag are Y (yes) and N (no).
−z	(**manual** tunnel only) New header format flag. The new header format reserves a field in ESP or AH header for replay prevention and also allows ESP authentication. The replay field is used only when the replay flag (**−y**) is set to Y. The valid values are Y (yes) and N (no).

Related Information

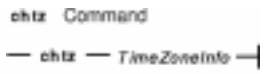
The **exptun** command, **gentun** command, **imptun** command, **lstun** command, **mktun** command, and **rmtun** command.

chtz Command

Purpose

Changes the *TimeZoneInfo* (TZ) environment variable in the **/etc/environment** file.

Syntax



chtz*TimeZoneInfo*

Description

The **chtz** command is a high−level shell command that changes the TZ environment variable in the **/etc/environment** file. The **chtz** command returns a value of `0` if successful and nonzero if unsuccessful.

Files

/etc/environment Contains variables specifying the basic environment for all processes.

Related Information

The **date** command.

The **environment** file.

chuser Command

Purpose

Changes user attributes.

Syntax

chuser*Attribute*=*Value ... Name*

Description

> **Attention:** Do not use the **chuser** command if you have a Network Information Service (NIS) database installed on your system.

The **chuser** command changes attributes for the user identified by the *Name* parameter. The user name must already exist as an alphanumeric string of 8 bytes or less. To change an attribute, specify the attribute name and the new value with the *Attribute*=*Value* parameter. The following files contain user attributes that are set by this command:

- **/etc/passwd**
- **/etc/security/environ**
- **/etc/security/limits**
- **/etc/security/user**
- **/etc/security/user.roles**
- **/etc/security/audit/config**
- **/etc/group**
- **/etc/security/group**

If you specify a single incorrect attribute or attribute value with the **chuser** command, the command does not change any attribute. You can use the Web−based System Manager Users application (**wsm users** fast path) run this command. You could also use the System Management Interface Tool (SMIT) **smit chuser** fast path to run this command.

Restrictions on Changing Users

To ensure the integrity of user information, some restrictions apply when using the **chuser** command. Only the root user or users with UserAdmin authorization can use the **chuser** command to perform the following tasks:

- Make a user an administrative user by setting the **admin** attribute to **true**.
- Change any attributes of an administrative user.
- Add a user to an administrative group.

An administrative group is a group with the **admin** attribute set to **true**. Members of the **security** group can change the attributes of nonadministrative users and add users to nonadministrative groups.

The **chuser** command manipulates local user data only. You cannot use it to change data in registry servers like NIS and DCE.

Attributes

If you have the proper authority you can set the following user attributes:

account_locked Indicates if the user account is locked. Possible values include:

true
The user's account is locked. The values **yes**, **true**, and **always** are equivalent. The user is denied access to the system.

false
The user's account is not locked. The values **no**, **false**, and **never** are equivalent. The user is allowed access to the system.

admin Defines the administrative status of the user. Possible values are:

true
The user is an administrator. Only the root user can change the attributes of users defined as administrators.

false
The user is not an administrator. This is the default value.

admgroups Lists the groups the user administrates. The *Value* parameter is a comma−separated list of group names.

auditclasses Lists the user's audit classes. The Value parameter is a list of comma−separated classes, or a value of **ALL** to indicate all audit classes.

auth1 Lists the primary methods for authenticating the user. The *Value* parameter is a comma−separated list of *Method***;***Name* pairs. The *Method* parameter is the name of the authentication method. The *Name* parameter is the user to authenticate. If you do not specify a *Name* parameter, the name of the invoking login program is used.

Valid authentication methods are defined in the **/etc/security/login.cfg** file. By default, the SYSTEM method and local password authentication are used. The NONE method indicates that no primary authentication check is made.

auth2 Lists the secondary methods used to authenticate the user. The *Value* parameter is a comma−separated list of *Method***;***Name* pairs. The *Method* parameter is the name of the authentication method. The *Name* parameter value is the user to authenticate.

If this attribute is not specified, the default is NONE, indicating that no secondary authentication check is made. Valid authentication methods are defined in the **/etc/security/login.cfg** file. If you do not specify a *Name* parameter, the name of the invoking login program is used.

core Specifies the soft limit for the largest core file a user's process can create. The *Value* parameter is an integer representing the number of 512−byte blocks.

core_hard Specifies the largest core file a user's process can create. The Value parameter is an integer representing the number of 512−byte blocks. This atrribute applies to AIX Version 4.2 or later.

cpu Identifies the soft limit for the largest amount of system unit time (in seconds) that a user's process can use. The *Value* parameter is an integer. The default value is −1 which turns off restrictions.

cpu_hard Identifies the largest amount of system unit time (in seconds) that a user's process can use. The Value parameter is an integer. The default value is −1 which turns off restrictions. This atrribute applies to AIX Version 4.2 or later.

daemon Indicates whether the user specified by the Name parameter can run programs using the **cron** daemon or the **src** (system resource controller) daemon. Possible values are:

true
The user can initiate **cron** and **src** sessions. This is the default.

false
The user cannot initiate **cron** and **src** sessions.

data Specifies the soft limit for the largest data segment for a user's process. The *Value* parameter is an integer representing the number of 512−byte blocks. The minimum allowable value for this attribute is 1272.

data_hard Specifies the largest data segment for a user's process. The Value parameter is an integer representing the number of 512−byte blocks. The minimum allowable value for this attribute is 1272. This atrribute applies to AIX Version 4.2 or later.

dictionlist Defines the password dictionaries used by the composition restrictions when checking new passwords.

The password dictionaries are a list of comma−separated absolute path names, evaluated from left to right. All dictionary files and directories must be write protected from all users except root. The dictionary files are formatted one word per line. The word starts in the first column and terminates with a newline character. Only 7 bit ASCII words are supported for passwords. If you install text processing on your system, the recommended dictionary file is the **/usr/share/dict/words** file.

expires Identifies the expiration date of the account. The Value parameter is a 10−character string in the *MMDDhhmmyy* form, where *MM* = month, *DD* = day, *hh* = hour, *mm* = minute, and *yy* = last 2 digits of the years 1939 through 2038. All characters are numeric. If the *Value* parameter is 0, the account does not expire. The default is 0. See the **date** command for more information.

fsize Defines the soft limit for the largest file a user's process can create or extend. The *Value* parameter is an integer representing the number of 512−byte blocks. To make files greater than 2G, specify `−1` or `unlimited`. The minimum value for this attribute is 8192.

fsize_hard Defines the largest file a user's process can create or extend. The Value parameter is an integer representing the number of 512−byte blocks. To make files greater than 2G, specify `−1` or `unlimited`. The minimum value for this attribute is 8192. This attribute applies to AIX Version 4.2 or later.

gecos Supplies general information about the user specified by the Name parameter. The *Value* parameter is a string with no embedded : (colon) characters and cannot end with the characters '#!'.

groups Identifies the groups the user belongs to. The *Value* parameter is a comma−separated list of group names.

histexpire Defines the period of time (in weeks) that a user cannot reuse a password. The value is a decimal integer string. The default is 0, indicating that no time limit is set. Only an administrative user can change this attribute.

histsize Defines the number of previous passwords a user cannot reuse. The value is a decimal integer string. The default is 0. Only an administrative user can change this attribute.

home Identifies the home directory of the user specified by the Name parameter. The *Value* parameter is a full path name.

id Specifies the user ID. The *Value* parameter is a unique integer string. Changing this attribute compromises system security and, for this reason, you should not change this attribute.

login Indicates whether the user can log in to the system with the **login** command.

Possible values are:

true
The user can log in to the system. This is the default.

false
The user cannot log in to the system.

loginretries
Defines the number of unsuccessful login attempts allowed after the last successful login before the system locks the account. The value is a decimal integer string. A zero or negative value indicates that no limit exists. Once the user's account is locked, the user will not be able to log in until the system administrator resets the user's unsuccessful_login_count attribute in the **/etc/security/lastlog** file to be less than the value of loginretries. To do this, enter the following:

```
chsec -f /etc/security/lastlog -s username -a \
unsuccessful_login_count=0
```

logintimes
Defines the days and times that the user is allowed to access the system. The value is a comma-separated list of entries in one of the following formats:

```
[!]:<time>-<time>

[!]<day>[-<day>][:<time>-<time>]

[!]<month>[<daynum>][-<month>[<daynum>]][:<time>-<time>]
```

Possible values for <day> include mon, tues, w, THU, Friday, sat, and SUNDAY. Indicate the day value as any abbreviated day of the week; however, the abbreviation must be unique with respect to both day and month names. The range of days can be circular, such as Tuesday-Monday. Day names are case insensitive.

Possible values for <time> include times specified in 24-hour military format. Precede the time value with a : (colon) and specify a string of 4 characters. Leading zeros are required. Thus, 0800 (8am) is valid while 800 is not valid. An entry consisting of only a specified time period applies to every day. The start hour must be less than the end hour. The time period cannot flow into the next day.

Possible values for <month> include Jan, F, march, apr, and s. Indicate the month value as any abbreviated month; however, the abbreviation must be unique with respect to both day and month names. The range of months can be circular, such as September-June. Month names are case insensitive.

Possible values for <daynum> include days 1-31 of a month. This value is checked against the specified month. Specify the month value as either a 1 or 2 character string. A month specified without a daynum value indicates the first or last day of the month, depending on if the month is the start or end month specified, respectively.

Entries prefixed with ! (exclamation point) deny access to the system and are called DENY entries. Entries without the ! prefix allow access and are called ACCESS entries. The ! prefix applies to single entries and must prefix each entry. Currently, the system allows 200 entries per user.

This attribute is internationalized. Month and day names can be entered and are displayed in the language specified by the locales variables set for the system. The relative order of the month and day values are also internationalized; the <month><daynum> and <daynum><month> formats are accepted.

The system evaluates entries in the following order:

1. All DENY entries. If an entry matches the system time, the user is denied access and the ALLOW entries are not processed.
2. All ALLOW entries, if no DENY entries exist. If an ALLOW entry matches the system time, the user is allowed access. If an ALLOW entry does not match the system time, the user is denied access. If no ALLOW entry exists, the user is permitted to log in.

maxage Defines the maximum age (in weeks) of a password. The password must be changed by this time. The value is a decimal integer string. The default is a value of 0, indicating no maximum age.

maxexpired Defines the maximum time (in weeks) beyond the **maxage** value that a user can change an expired password. After this defined time, only an administrative user can change the password. The value is a decimal integer string. The default is −1, indicating restriction is set. If the **maxexpired** attribute is 0, the password expires when the **maxage** value is met. If the **maxage** attribute is 0, the **maxexpired** attribute is ignored.

maxrepeats Defines the maximum number of times a character can be repeated in a new password. Since a value of 0 is meaningless, the default value of 8 indicates that there is no maximum number. The value is a decimal integer string.

minage Defines the minimum age (in weeks) a password must be before it can be changed. The value is a decimal integer string. The default is a value of 0, indicating no minimum age.

minalpha Defines the minimum number of alphabetic characters that must be in a new password. The value is a decimal integer string. The default is a value of 0, indicating no minimum number.

mindiff Defines the minimum number of characters required in a new password that were not in the old password. The value is a decimal integer string. The default is a value of 0, indicating no minimum number.

minlen Defines the minimum length of a password. The value is a decimal integer string. The default is a value of 0, indicating no minimum length. The maximum value allowed is 8. This attribute is determined by the **minalpha** attribute added to the **minother** attribute. If the result of this addition is greater than the **minlen** attribute, the minimum length is set to the result.

minother Defines the minimum number of non−alphabetic characters that must be in a new password. The value is a decimal integer string. The default is a value of 0, indicating no minimum number.

nofiles Defines the soft limit for the number of file descriptors a user process may have open at one time. The **Value** parameter is an integer.

nofiles_hard Defines the hard limit for the number of file descriptors a user process may have open at one time. The **Value** parameter is an integer. The default value is −1, which sets the limit to the maximum allowed by the system.

pgrp Identifies the user's primary group. The *Value* parameter must contain a valid group name and cannot be a null value.

pwdchecks Defines the password restriction methods enforced on new passwords. The value is a list of comma−separated method names and is evaluated from left to right. A method name is either an absolute path name or a path name relative to **/usr/lib** of an executable load module.

pwdwarntime Defines the number of days before the system issues a warning that a password change is required. The value is a decimal integer string. A zero or negative value indicates that no message is issued. The value must be less than the difference of the **maxage** and **minage** attributes. Values greater than this difference are ignored and a message is issued when the **minage** value is

reached.

rlogin Permits access to the account from a remote location with the **telnet** or**rlogin** commands. Possible values are:

true
The user account can be accessed remotely. This is the default **rlogin** value.

false
The user cannot be accessed remotely.

roles Lists the administrative roles for this user. The *Value* parameter is a list of role names, separated by commas.

rss The soft limit forthe largest amount of physical memory a user's process can allocate.The *Value* parameter is a decimal integer string specified in units of 512−byte blocks. This value is not currently enforced by the system.

rss_hard The largest amount of physical memory a user's process can allocate. The Value parameter is a decimal integer string specified in units of 512−byte blocks. This value is not currently enforced by the system. This atrribute applies to AIX Version 4.2 or later.

shell Defines the program run for the user at session initiation. The *Value* parameter is a full path name.

stack Specifies the soft limit for the largest process stack segment for a user's process. The *Value* parameter is an integer representing the number of 512−byte blocks to allot. The minimum allowable value for this attribute is 49.

stack_hard Specifies the largest process stack segment for a user's process. The Value parameter is an integer representing the number of 512−byte blocks to allot. The minimum allowable value for this attribute is 49. This atrribute applies to AIX Version 4.2 or later.

su Indicates whether another user can switch to the specified user account with the **su** command. Possible values are:

true
Another user can switch to the specified account. This is the default.

false
Another user cannot switch to the specified account.

sugroups Lists the groups that can use the **su** command to switch to the specified user account. The *Value* parameter is a comma−separated list of group names, or a value of **ALL** to indicate all groups. An ! (exclamation point) in front of a group name excludes that group. If this attribute is not specified, all groups can switch to this user account with the **su** command.

sysenv Identifies the system−state (protected) environment. The *Value* parameter is a set of comma−separated *Attribute*=*Value* pairs as specified in the **/etc/security/environ** file.

tpath Indicates the user's trusted path status. The possible values are:

always
The user can only execute trusted processes. This implies that the user's initial program is in the trusted shell or some other trusted process.

notsh
The user cannot invoke the trusted shell on a trusted path. If the user enters the secure attention key (SAK) after logging in, the login session ends.

nosak
The secure attention key (SAK) is disabled for all processes run by the user. Use this value if the user transfers binary data that may contain the SAK sequence. This is the default value.

on

	The user has normal trusted path characteristics and can invoke a trusted path (enter a trusted shell) with the secure attention key (SAK).
ttys	Lists the terminals that can access the account specified by the *Name* parameter. The *Value* parameter is a comma−separated list of full path names, or a value of **ALL** to indicate all terminals. An ! (exclamation point) in front of a terminal name excludes that terminal. If this attribute is not specified, all terminals can access the user account.
umask	Determines file permissions. This value, along with the permissions of the creating process, determines a file's permissions when the file is created. The default is 022.
usrenv	Defines the user−state (unprotected) environment. The *Value* parameter is a set of comma−separated *Attribute*=*Value* pairs as specified in the **/etc/security/environ** file.

Security

Access Control: This command should grant execute (x) access only to the root user and the security group. This command should be installed as a program in the trusted computing base (TCB). The command should be owned by the root user with the **setuid** (SUID) bit set.

Files Accessed:

Mode	File
rw	**/etc/passwd**
rw	**/etc/security/user**
rw	/etc/security/user.roles
rw	**/etc/security/limits**
rw	**/etc/security/environ**
rw	**/etc/security/audit/config**
rw	**/etc/group**
rw	**/etc/security/group**

Auditing Events:

Event	Information
USER_Change	user, attributes

Examples

1. To enable user `smith` to access this system remotely, enter:

   ```
   chuser rlogin=true smith
   ```

2. To change the expiration date for the `davis` user account to 8 a.m., 1 May, 1995, enter:

   ```
   chuser expires=0501080095 davis
   ```

3. To add `davis` to the groups `finance` and `accounting`, enter:

   ```
   chuser groups=finance,accounting davis
   ```

Files

/usr/bin/chuser	Contains the **chuser** command.
/etc/passwd	Contains the basic attributes of users.
/etc/group	Contains the basic attributes of groups.
/etc/security/group	Contains the extended attributes of groups.
/etc/security/user	Contains the extended attributes of users.
/etc/security/user.roles	Contains the administrative role attributes of users.
/etc/security/lastlog	Contains the last login attributes of users.
/etc/security/limits	Defines resource quotas and limits for each user.
/etc/security/audit/config	Contains audit configuration information.
/etc/security/environ	Contains the environment attributes of users.

Related Information

The **chfn** command, **chgroup** command, **chgrpmem** command, **chsh** command, **lsgroup** command, **lsuser** command, **mkgroup** command, **mkuser** command, **passwd** command, **pwdadm** command, **rmgroup** command, **rmuser** command, **setgroups** command, **setsenv** command, **su** command.

Security Administration in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

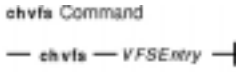
Administrative Roles Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* .

chvfs Command

Purpose

Changes entries in the **/etc/vfs** file.

Syntax



chvfs *VFSEntry*

Description

The **chvfs** command changes **/etc/vfs** file entries by specifying the following fields within the *VFSEntry* parameter. The *VFSEntry* parameter is composed of the following: *VFSName***:***VFSNumber***:***MountHelper***:***FileSystemHelper* **.**

Any of the entries in the *VFSEntry* can be null, with the exception of the *VFSName* entry. If all of the arguments are satisfactory, the entry in the **/etc/vfs** file is changed.

Parameters

VFSEntry	A string in the following format: *VFSName***:***VFSNumber***:***MountHelper***:***FileSystemHelper*
VFSName	Specifies the name of a virtual file system type.
VFSNumber	Specifies the virtual file system type's internal number as known by the kernel.
MountHelper	Specifies the name of the backend used to mount a file system of this type.
FileSystemHelper	Specifies the name of the backend used by certain file system specific commands to perform operations on a file system of this type.

Examples

To change the *FileSystemHelper* for the vfs entry named `newvfs`, enter:

```
chvfs "newvfs:::/etc/helper/testhelper"
```

The missing parameters are left unchanged.

Files

/etc/vfs Contains descriptions of virtual file system types.

Related Information

The **crvfs** command, **lsvfs** command, **mount** command, and **rmvfs** command.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts:*

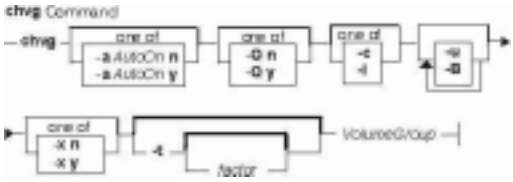
Operating System and Devices explains file system types, management, structure, and maintenance.

chvg Command

Purpose

Sets the characteristics of a volume group.

Syntax



chvg [−**a***AutoOn* { **n** | **y** }] [−**c** | −**l**] [−**Q** { **n** | **y** }] [−**u**] [−**x** { **n** | **y** }] [−**t** [*factor*]] [−**B**] *VolumeGroup*

Description

The **chvg** command specifies whether or not the volume group is automatically activated during the system startup. If there is a volume group that is infrequently used, you may not want it activated at system startup because it uses kernel resources (memory).

MAXPVS 32 (128 if −B flag is used)

> **Note:** To use this command, you must either have root user authority or be a member of the **system** group.

You can use the Web−based System Manager Volumes application (**wsm lvm** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chvg** fast path to run this command.

Flags

−**a***AutoOn* Determines if the volume group is automatically activated during system startup. The *AutoOn* variable can be either of the following:

n The volume group is not automatically activated during system startup.

y The volume group is automatically activated during system startup.

−**B** Changes the volume group to big vg format. This can accommodate up to 128 physical volumes and 512 logical volumes.

Notes:

1. The −**B** flag cannot be used if there are any stale physical partitions or there are any open logical volumes in the volume group.
2. Once the volume group is converted, it cannot be imported into AIX Version 4.3.1 or lower versions.
3. The −**B** flag cannot be used if the volume group is varied on in concurrent mode.
4. There must be enough free partitions available on each physical volume for the VGDA expansion for this operation to be successful.
5. Since the VGDA resides on the edge of the disk and it requires contiguous space for expansion, the free partitions are required on the edge of the disk. If those partitions are allocated for user usage, they will

be migrated to other free partitions on the same disk. The rest of the physical partitions will be renumbered to reflect the loss of the partitions for VGDA usage. This will change the mappings of the logical to physical partitions in all the PVs of this VG. If you have saved the mappings of the LVs for a potential recovery operation, you should generate the maps again after the completion of the conversion operation. Also, if the backup of the VG is taken with the map option and you plan to restore using those maps, the restore operation may fail since the partition number may no longer exist (due to reduction). It is recommended that backup is taken before the conversion, and right after the conversion if the map option is utilized. Since the VGDA space has been increased substantially, every VGDA update operation (creating an LV, changing an LV, adding a PV, and so forth) may have a considerably longer duration.

−c Changes the volume group into a Concurrent Capable volume group. However, the volume group must be varied on in non−concurrent mode for this command to take effect. This flag only applies to AIX Version 4.2 or later.

−l Changes the volume group into a Non−Concurrent Capable volume group. The volume group must be varied on in non−concurrent mode for this command to take effect. This flag only applies to AIX Version 4.3 or later.

−Q Determines if the volume group is automatically varied off after losing its quorum of physical volumes. The default value is yes. The change becomes effective the next time the volume group is activated.

n
The volume group stays active until it loses all of its physical volumes.

y
The volume group is automatically varied off after losing its quorum of physical volumes.

Note: Run the **bosboot** or **savebase** command after the **chvg −Q n** or **chvg −Q y** command to update the boot image.

−t [*factor*] Changes the limit of the number of physical partitions per physical volume, specified by *factor*. *factor* should be between 1 and 16 for 32 disk volume groups and 1 and 64 for 128 disk volume groups.

If *factor* is not supplied, it is set to the lowest value such that the number of physical partitions of the largest disk in volume group is less than *factor* x 1016.

If *factor* is specified, the maximum number of physical partitions per physical volume for this volume group changes to *factor* x 1016.

Notes:

1. If the volume group is created in AIX 3.2/4.1.2 in violation of 1016 physical partitions per physical volume limit, this flag can be used to convert the volume group to a supported state. This will ensure proper stale/fresh marking of partitions.
2. *factor* cannot be changed if there are any stale physical partitions in the volume group.
3. Once volume group is converted, it cannot be imported into AIX Version 4.3 or lower versions.
4. This flag cannot be used if the volume group is varied on in concurrent mode.
5. The maximum number of physical volumes that can be included in this volume group will be reduced to (MAXPVS/factor).

6. Change of the volume group may require the modification of the LVM meta data. In this situation the volume group will be varied off in management mode to ensure the integrity of the Volume group, needing the closure of all open logical volumes in this volume group. Since logical volumes in rootvg cannot be closed, rootvg cannot be converted if it needs modification of the meta−data as part of the **chvg −t** operation.

−u Unlocks the volume group. This option is provided if the volume group is left in a locked state by abnormal termination of another LVM operation (such as the command core dumping, or the system crashing).

Note: Before using the **−u** flag, make sure that the volume group is not being used by another LVM command.

−x Changes the mode which the Concurrent Capable volume group is varied on. The volume group must be varied on in non−concurrent mode for this command to take effect. This flag only applies to AIX Version 4.2 or later.

y

autovaryon the volume group in concurrent mode.

n

autovaryon the volume group in non−concurrent mode.

Note: If the volume group is not created Concurrent Capable, this command has no effect on the volume group.

In order for this auto−varyon into concurrency of the volume group to take effect, you must enter the following line into the **/etc/inittab** file:

```
rc_clvmv:2:wait:/usr/sbin/clvm_cfg 2>&1
```

Attention: This entry must be added after the entry used to initiate **srcmstr**.

Examples

1. To cause volume group `vg03` to be automatically activated during system startup, enter:
   ```
   chvg −a y vg03
   ```

2. To change the volume group `vg03` to a supported state if it is in violation of 1016 physical partitions per physical volume limit, enter
   ```
   chvg −t  vg03
   ```

3. To change the maximum number of physical partitions per physical volume to 2032 and maximum number of physical volumes in volume group `vg03` to 16, enter
   ```
   chvg −t 2 vg03
   ```

Files

/**usr**/**sbin** Directory where the **chvg** command resides.

Related Information

Commands: **bosboot**, **lsvg**, **mkvg**, **savebase**, **varyonvg**.

The Logical Volume Storage Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and allocation characteristics.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the structure, main menus, and tasks that are done with SMIT.

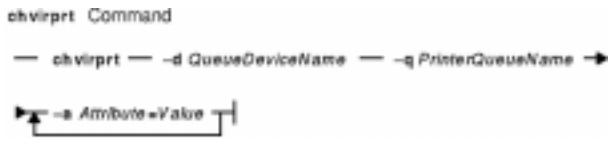
AIX HACMP/6000 Concepts and Facilities.

chvirprt Command

Purpose

Changes the attribute values of a virtual printer.

Syntax



chvirprt −d *QueueDeviceName* **−q** *PrintQueueName* **−a** *Attribute*=*Value* ...

Description

The **chvirprt** command changes attribute values for the virtual printer assigned to *PrintQueueName* and *QueueDeviceName*.

> **Note:** Attribute names for default values of the **qprt** command line flags can be specified by entering the flag letters. For example, to change the default value for the **−w** flag (page width) to 132, enter `w=132`. All other attribute names must be 2 characters long.

You can use the Web−based System Manager Printer Queues application (**wsm printers** fast path) run this command. You could also use the System Management Interface Tool (SMIT) **smit chvirprt** fast path to run this command.

Flags

−a*Attribute*=*Value*	Replaces the value for *Attribute* with *Value*. If *Value* contains one or more spaces, it must be surrounded by quotes ('*Value*'). be the last flag when entering the **chvirprt** command on the command line.
−d*QueueDeviceName*	Specifies the name of the queue device to which the virtual printer is assigned.
−q*PrintQueueName*	Specifies the name of the print queue to which the virtual printer is assigned.

Examples

To change the default page width to `132` characters (the **w** attribute) and specify that user `mary` receives the "intervention required" messages (the **si** attribute) for the virtual printer associated with the `proq` print queue and the `mypro` queue device, enter:

```
chvirprt −q proq −d mypro −a si=mary w=132
```

Files

/etc/qconfig	Configuration file
/usr/sbin/chvirprt	Contains the **chvirprt** command.
/var/spool/lpd/pio/@local/custom/*	Virtual printer attribute files

/var/spool/lpd/pio/@local/ddi/* Digested virtual printer attribute files.

Related Information

The **lsvirprt** command, **mkvirprt** command, **qprt** command, **rmvirprt** command, **smit** command.

The **qconfig** file.

Changing or Showing Characteristics of a Virtual Printer in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Colon File Conventions in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Specific Information in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Support in *AIX Version 4.3 Guide to Printers and Printing*.

Virtual Printer Definitions and Attributes in *AIX Version 4.3 Guide to Printers and Printing*.

Adding a Printer Using the Printer Colon File in *AIX Version 4.3 Guide to Printers and Printing*.

Printer Code Page Translation Tables in *AIX Version 4.3 Guide to Printers and Printing*.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

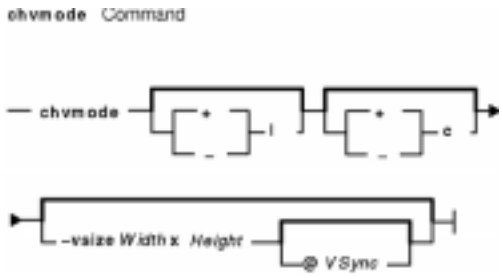
chvmode Command

Purpose

Changes the current output device and viewport size of the X server.

Note: This command is usable only while the X server is running.

Syntax



chvmode [{ + | − } **l**] [{ + | − } **c**] [−**vsize***Width***x***Height* [@ VSync]

Description

The **chvmode** command changes the current output device and viewport size used by the X server.

Viewport size specification is usable only for a CRT display and its resolution has panning option. You can use a Web−based System Manager Devices application (**wsm devices** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) to run this command.

Flags

+/−c	Enables or disables CRT output.
+/−l	Enables or disables LCD output.
−**vsize***Width***x***Height*[@VSync]	Specifies viewport size of CRT display and the vertical synchronization (refresh rate in Hz). If **@***VSync* is not specified, the current vertical synchronization frequency is used.

Security

Access Control: Any User

Auditing Events: None

Exit Status

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

Examples

1. To disable the LCD panel and enable the CRT display, enter:

   ```
   chvmode −l +c
   ```

2. To change the current CRT viewport to be 1024x768, enter:

   ```
   chvmode −vsize 1024x768
   ```

3. To specify VGA mode with high refresh rate of 75Hz, enter:

   ```
   chvmode −vsize 640x480@75
   ```

Files

/usr/bin/X11/chvmode Contains the **chvmode** command.

Related Information

The **lsvmode** command.

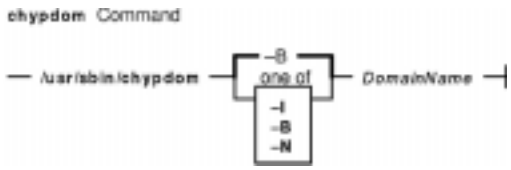
Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

chypdom Command

Purpose

Changes the current domain name of the system.

Syntax



/usr/sbin/chypdom [**−I** | **−B** | **−N**] *DomainName*

Description

The **chypdom** command will change the domain name of the system. The *DomainName* parameter specifies the new domain name for the system.

You can use the Web−based System Manager Network application (**wsm network** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit chypdom** fast path to run this command.

Flags

−I Specifies that the domain name should be changed in the **/etc/rc.nfs** file. With this flag, the domain name will be changed on the next system restart.

−B Specifies that the domain name should be changed now and the **/etc/rc.nfs** file should be updated to reflect the change.

−N Specifies that the domain name should be changed now. No change is made to the **/etc/rc.nfs** file. The **domainname** command is executed to change the domain name of the system.

Examples

To modify the **/etc/rc.nfs** file to set the domain name to `mydomain` on the next system restart, enter:

```
chypdom −I mydomain
```

Files

/etc/rc.nfs Contains the startup script for the NFS and NIS daemons.

Related Information

The **domainname** command, **mkclient** command, **mkmaster** command, **mkslave** command, **smit** command.

Setting up and running Web−based System Management in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

Network Information Service (NIS) in *AIX Version 4.3 Network Information Services (NIS and NIS+) Guide*.

NIS Reference.

ckpacct Command

Purpose

Checks data file size for process accounting.

Syntax

/usr/sbin/acct/ckpacct [*BlockSize*]

Description

The **ckpacct** command checks the size of the active data file, **/var/adm/pacct**. Normally, the **cron** daemon runs this command. If the size of the active data file exceeds the number of blocks specified by the *BlockSize* parameter, the **ckpacct** command invokes the **turnacctswitch** command to turn off process accounting. The default value for the *BlockSize* parameter is 1000.

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

When the **MAILCOM** environment variable is set to **mail root adm**, a mail message is sent both to the **root** and **adm** groups if an error occurs.

Security

Access Control: This command should grant execute (x) access only to members of the **adm** group.

Examples

To automatically check the size of the **/var/adm/pacct** data file, add the following to the **/var/spool/cron/crontabs/root** file:

```
5 * * * * /usr/sbin/acct/ckpacct
```

This example shows the instructions the **cron** daemon reads and acts upon. The **ckpacct** command runs at 5 minutes past every hour (`5 *`) every day. This command is only one of the accounting instructions normally given to the **cron** daemon. See "Setting Up an Accounting System" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more information on typical **cron** accounting entries.

Files

/usr/sbin/acct	The path to the accounting commands
/var/adm/pacct	Current file for process accounting.

Related Information

The **acctcom** command, **acctprc1, acctprc2,** or **accton** command, **turnacct** command.

The **cron** daemon.

The **acct** subroutine.

For more information about the accounting system, the preparation of daily and monthly reports, and the accounting files, see the Accounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

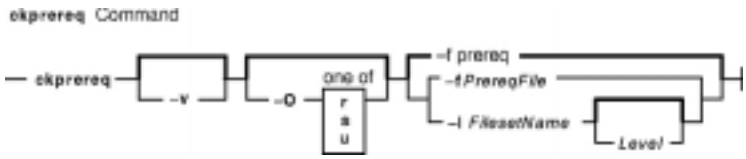
Setting Up an Accounting System in *AIX Version 4.3 System Management Guide: Operating System and Devices* explains the steps you must take to establish an accounting system.

ckprereq Command

Purpose

Verifies that all prerequisite software is available and at the appropriate revision levels.

Syntax



ckprereq [**−v**] [**−O** { **r** | **u** | **s** }] [**−f***PrereqFile* | **−l***FilesetName* [*Level*]]

Description

The **ckprereq** command determines whether the system level is compatible with the software product to be installed or updated.

The **ckprereq** command is designed to be used during the installation procedures of a software product.

When **ckprereq** is invoked with the **−f** flag, the *PrereqFile* parameter specifies a software prerequisite list file. Each record in this file contains information about a prerequisite fileset needed to complete the installation procedure.

When **ckprereq** is invoked with the **−l** flag, the prerequisite information is read from the *ProductName* information in the Software Vital Product Data (SWVPD) database.

If the *PrereqFile* parameter was given with the **−f** flag, then an output file is produced by the **ckprereq** command. The output file overwrites the input file and is a listing of the original input. Any failing lines are marked with a failure code in the first column. The **ckprereq** command ignores the failure codes if an output from a previous **ckprereq** call is used as input.

There are four possible requisite tests: **prereq**, **coreq**, **ifreq**,and **instreq**.

A **prereq** is a test to check that a fileset is installed and at a specified revision level. To be considered installed, the SWVPD entry for the software product must be in the APPLIED, APPLYING, COMMITTED, or COMMITTING state. A **prereq** requires that the fileset also be at the specified revision level before installing the independent fileset.

A **coreq** test is similar to a **prereq**, except that **coreq** tests can be installed in any order, but **prereq** tests require a specific order. If a corequisite software product is not yet installed, the test is ignored and the failure codes are not set because it is assumed that the software product will be installed. The **coreq** test is ignored by the **ckprereq** command. (It is not ignored by the installp command's requisite checking procedures.)

An **ifreq** test is identical to a **coreq**, except that it tests for the revision level only if the fileset is installed. If the fileset is not installed, the **ifreq** test is ignored.

An **instreq** test is treated like a **prereq** test by the **ckprereq** command. The special meaning of instreq is only used by the up−front requisite checks of the **installp** command.

The **installp** command checks corequisite and if−requisite filesets at the completion of an install set, and returns messages for any unsatisfied **coreq** or **ifreq** conditions. An if−requisite condition would be unsatisfied if the if−requisite product is installed, but does not match the revision level specified.

Flags

−f*PrereqFile*	Specifies the file name of a prerequisite list file.
−l*FilesetName*[*Level*]	Specifies the name of the fileset or fileset update under which to look for the prerequisite information from the SWVPD database.
−O {**r**|**u**|**s**}	Specifies the part of the file tree of the software product that is to be checked. If this flag is not specified, the **ckprereq** command uses the value of the **INUTREE** environment variable to determine which part to check. The **INUTREE** environment variable is set by the **installp** command. The **r** option indicates the / (root) part of the software product is checked. The **u** option indicates the **/usr** part of the software product is checked. The **s** option indicates the **/usr/share** part of the software product is checked. Only one part can be checked at a time.
−v	Displays a descriptive message to standard error for each failure in the prerequisite list file.

Return Values

The **ckprereq** command tests the current version, release, modification level, fix level, and fix ID found in the SWVPD and marks the first column in each failing line in the output file with one of the following codes if the test was unsuccessful:

f The test for the fix (level) was unsuccessful.
m The test for the modification level was unsuccessful.
n The fileset is not installed or is set to **broken**.
p The test for the fix ID was unsuccessful.
r The test for the release was unsuccessful.
s There is a syntax error in the *PrereqFile* parameter.
v The test for the version was unsuccessful.

If a serious error occurs, such as an invalid command line or a syntax error in the prerequisite list file, the return code for the **ckprereq** command is 255. Otherwise, the return code is a number that represents the number of tests that failed.

Security

Access Control: You must have root authority to run this command.

Examples

1. To check that the requisite specifications in the file **/tmp/prq.test**, that has the following contents:

   ```
   *prereq bos.rte 4.1.0.0
   *prereq X11.base.rte 4.1.0.0
   ```

   are satisfied, while reporting any failures, enter:

```
ckprereq -vf /tmp/prq.test
```

2. To check all the requisite software listed in the **/usr/lpp/snaserv/prereq2** file for the root part, enter:

```
ckprereq -f /usr/lpp/snaserv/prereq2 -Or
```

3. To check that the requisites of the installed fileset update bos.net.tcp.client at level 4.1.0.1 are met, enter:

```
ckprereq -l bos.net.tcp.client 4.1.0.1
```

Files

/etc/objrepos/product	Database containing information about the software installed in the **/root** part of the file system.
/**usr/lib/objrepos/product**	Database containing information about the software installed in the **/usr** part of the file system.
/usr/share/lib/objrepos/product	Database containing information about the software installed in the **/usr/share** part of the file system.

Related Information

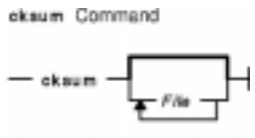
The **installp** command.

cksum Command

Purpose

Displays the checksum and byte count of a file.

Syntax



cksum [*File* ...]

Description

The **cksum** command reads the files specified by the *File* parameter and calculates a 32−bit checksum Cyclic Redundancy Check (CRC) and the byte count for each file. If no files are specified, the **cksum** command reads standard input. The checksum, number of bytes, and file name are written to standard output. If standard input is used, the path name and leading space are omitted.

The **cksum** command can be used to compare a suspect file copied or communicated over noisy transmission lines against an exact copy of a trusted file. The comparison made by the **cksum** command may not be cryptographically secure. However, it is unlikely that an accidentally damaged file will produce the same checksum as the original file.

The **cksum** command uses a different algorithm to calculate the 32−bit checksum CRC than the **sum** command. The **cksum** command uses a CRC algorithm based on the Ethernet standard frame check. For more information on the Ethernet standard, see "Understanding DLCETHER Protocol Support" in *AIX Communications Programming Concepts*.

> **Note:** The **cksum** command is POSIX 1003.2 compliant and the checksum produced is guaranteed to be calculated the same on all POSIX 1003.2 compliant systems.

The following generating polynomial defines CRC checksum encoding:

$$G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

The following procedure mathematically defines the CRC value corresponding to a given file:

1. The *n* bits to be evaluated are considered to be the coefficients of a mod 2 polynomial M(x) of degree *n*−1. These *n* bits are the bits from the file. The most significant bit is the most significant bit of the first octet of the file. The last bit is the least significant bit of the last octet, padded with zero bits (if necessary) to achieve an integral number of octets, followed by one or more octets representing the length of the file as a binary value, least significant octet first. The smallest number of octets capable of representing this integer is used.
2. M(x) is multiplied by x32 (that is, shifted left 32 bits) and divided by G(x) using mod 2 division, producing a remainder R(x) of degree 31.
3. The coefficients of R(x) are considered to be a 32−bit sequence.
4. The bit sequence is complemented, and the result is the CRC.

Exit Status

This command returns the following exit values:

0 All files were processed successfully.
>0 An error occurred.

Examples

To display the checksum and the size, in bytes, of `file1` and `file2`, enter:

```
cksum file1 file2
```

If the checksum of the `file1` file is `3995432187` and contains `1390` bytes, and the checksum of the `file2` file is `3266927833` and contains `20912` bytes, the **cksum** command displays:

```
3995432187       1390    file1
3266927833       20912   file2
```

Files

/usr/bin/cksum Contains the **cksum** command.

Related Information

The **sum** command, **wc** command.

File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* gives an explanation of what a file system is and why to use one.

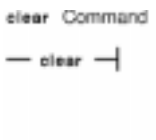
Understanding DLCETHER Protocol Support in *AIX Communications Programming Concepts* provides information on the Ethernet standard.

clear Command

Purpose

Clears the terminal screen.

Syntax



clear

Description

The **clear** command clears your screen, if possible. The **clear** command first checks the **TERM** environment variable for the terminal type. Next, the **/usr/share/lib/terminfo** directory, which contains terminal definition files, is checked to determine how to clear the screen. If the **TERM** environment variable is not set, the **clear** command exits without taking any action.

Examples

To clear your screen, enter:

```
clear
```

Files

/usr/share/lib/terminfo Contains terminal information database.

Related Information

The **tput** command.

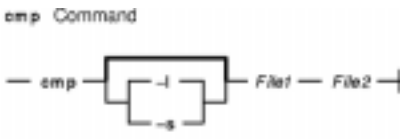
The Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the operating system processes input and output and how to use redirect and pipe symbols.

cmp Command

Purpose

Compares two files.

Syntax



cmp [**−l** | **−s**] *File1 File2*

Description

The **cmp** command compares files designated by the *File1* and *File2* parameters and writes the results to standard output. If you specify a **−** (minus sign) for either the *File1* or *File2* parameter, the **cmp** command reads standard input for that file. Only one file can be read from 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 at which the first difference occurs. If the **−l** flag is specified and 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 **cmp** command notes this. Normally, use the **cmp** command to compare non−text files and the **diff** command to compare text files.

Flags

−l (Lowercase L) Displays, for each difference, the byte number in decimal and the differing bytes in octal.

−s Returns only an exit value. A value of 0 indicates identical files; value of 1 indicates different files; a value of 2 indicates inaccessible file or a missing option.

Exit Status

This command returns the following exit values:

0 The files are identical.

1 The files are different. This value is given even if one file is an initial subsequence of the other (one file is identical to the first part of the other).

>1 An error occurred.

Examples

1. To determine whether two files are identical, enter:

   ```
   cmp prog.o.bak prog.o
   ```

   This compares `prog.o.bak` and `prog.o`. If the files are identical, then a message is not displayed. If the files differ, then the location of the first difference is displayed; for example:

```
prog.o.bak prog.o differ: char 4, line 1
```

If the message `cmp: EOF on prog.o.bak` is displayed, then the first part of `prog.o` is identical to `prog.o.bak`, but there is additional data in `prog.o`.

2. To display each pair of bytes that differ, enter:

   ```
   cmp −l prog.o.bak prog.o
   ```

   This 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 `prog.o.bak` and 141 in `prog.o`, the **cmp** command displays:

   ```
   5 101 141
   ```

3. To compare two files without writing any messages, enter:

   ```
   cmp −s prog.c.bak prog.c
   ```

   This gives an exit value of `0` if the files are identical, a value of `1` if different, or a value of `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.

Files

/usr/bin/cmp Contains the **cmp** command.

Related Information

The **comm** command, **diff** command, **ksh** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

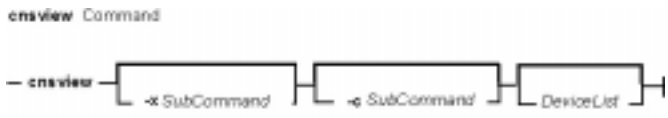
Input and Output Redirection Overview. in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

cnsview Command

Purpose

Provides information about changes to the parameters of a 7318 and its associated devices and drivers.

Syntax



cnsview [**−x***SubCommand*] [**−c***SubCommand*] [*DeviceList*]

Description

The **cnsview** command is normally used with Model P10s, but it can also be used with Model S20s that have P10−style ports. The command can execute subcommands from an interactive session or noninteractively when commands are passed as command−line arguments. Any subcommand that can be entered interactively can also be entered using the command line. For example:

```
cnsview −c "show traffic"
```

displays the number of bytes exchanged over a 7318 port. The argument following the **−c** flag is the subcommand to be executed.

> **Note:** Because this subcommand consists of two words, you must enclose the subcommand in quotes before it is passed to the AIX shell.

The same command can be entered interactively with the following sequence:

```
cnsview
>
>
 show traffic
>
>
 quit
```

The >> (greater than) characters are the **cnsview** command prompt, similar to the AIX shell prompt.

When used interactively, the **cnsview** command reads input from standard input and displays the results of commands, if any, to standard output. You can use AIX file redirection and pipes for both input and output.

> **Note:** Most end users should not have any need to use the **cnsview** command. It is intended primarily for the system administrator.

Flags

−c*SubCommand* Runs the subcommand specified by the *SubCommand* variable, without entering interactive mode.

−x*SubCommand* Runs the subcommand specified by the *SubCommand* variable with debug enhancements, without entering interactive mode.

Selecting Devices

Most of the **cnsview** subcommands apply to a selected device or devices. If you do not otherwise specify a device, the **cnsview** command assumes the serial line or the 7318 currently in use. If you do not specify a 7318 port and attempt to run a command from a TTY device that is not a 7318 port, such as the system console, the command may report an error such as:

```
cnsview: operation not supported on /dev/console
```

You can select one or more devices by listing them on the command line. For example:

```
cnsview −c "set hibaud" /dev/tty2 /dev/tty3
```

This command sets the `hibaud` property on two ports.

When you use the **cnsview** command interactively, you can also specify devices, using the **set device** subcommand. For example:

```
cnsview
>
>
 set device /dev/tty2 /dev/tty3
>
>
 set hibaud
>
>
 quit
```

> **Note:** When you are using the **set device** subcommand interactively, the shell file−name expansion facility is not available. Therefore, you must type in the full file name of each device.

You can select any device known to AIX, but most of the commands only work on 7318 devices.

You can use the **show device** command to display the list of selected devices. For example:

```
cnsview −c "show device"
```

This command shows the device you are currently using.

Physical Port to TTY Mapping

Issue the **cnsview** commands to locate a physical port for a particular device, as shown in the following example.

```
cnsview −c "show devmap" /dev/tty6
```

The output is similar to the following:

type com

station 1
port 5
session 0

```
cnsview -c "show eaddr" /dev/cns01
```

The output is similar to the following:

```
00406e0000bf
```

Where the value of *station* is used to build the device name. For example, a value of 1 for the station parameter is for the device, `/dev/cns01`.

Permissions

Only a 7318 device owner or a root user can run **cnsview** subcommands that affect the device. Subcommands that affect multiple devices such as the **daemon** command can only be run by a root user or a user with write access to the **/dev/cns** file.

cnsview Subcommands

The following subcommands can be used with the **cnsview** command:

explore
Searches the network for 7318s and prints a list of their names, network addresses, and status.

daemon [**start** | **stop** | **init**]
Controls the **cnsview** daemon process that manages all traffic to 7318s. When the daemon is stopped, any attempt to use a 7318 device results in an **ENXIO** error.

The **daemon init** command forces the daemon to reread the 7318 configuration file. If you modify this file after installing the system, issue the **daemon init** command.

help [*Command* [*Property* [*SubProperty*]]]
Lists the available commands or explains how to use a specific command. If the command specified by the *Command* parameter is **set** or **show**, the **help** command lists the properties of the selected device and its types.

ipxping [**−b** | *NetworkAddress*]
Sends an IPX echo message to a NetWare device whose address is specified by the *NetworkAddress* parameter. The **ipxping −b** command broadcasts an echo message. The **−b** flag should be issued with caution because it can generate a large burst of network traffic.

reboot [**−u**] [**−p***LoadFile*]

> **Attention:** Issuing this command immediately stops the 7318 without notice to anyone attached to the unit.

The **reboot** command restarts the 7318. The command session must be in password privilege mode to use this command. The **−u** flag forces the current load image to upload to the load host. The **−p** flag specifies an alternate load file (the *LoadFile* variable) rather than the one specified in the nonvolatile RAM.

reset
Resets a device, which puts the device back into its initial default state, turning off any special modes that might have been applied. Any process

	that has the device open receives a hangup signal.
quit	Terminates the **cnsview** command. **Exit** and **q** are synonyms for **quit**.

set [−]*Property* [*SubProperty*] [*Value*]	Sets the value of a property. Some properties are of type Boolean; that is, they can be true or false, represented by the values of 1 for true and 0 for false. You can set a Boolean property to true by entering:
	`set Property`
	This is equivalent to the command:
	`set Property 1`
	You can set a Boolean property to false by entering:
	`set −Property`
	This is equivalent to the command:
	`set Property 0`
	Note: Do not insert a space between the − (minus) and the property.

show*Property* [*SubProperty*]	Displays the contents of a property to standard output.

Properties

Each device managed by the **cnsview** command has a collection of parameters or properties that the **show** command can display. The **set** command can modify some of these parameters.

Properties can be scalar (meaning they have a single value) lists that have multiple values, or aggregates that are a collection of properties selected by subproperty. Lists are like arrays in a programming language; aggregates are like structures. Scalar properties can be one of the types shown in the table below.

Scalar Properties		
Type	**Description**	**Example Values**
Boolean	True or False	1,0
integer	A 32−bit number	120
string	A text string	/dev/tty2, "now is the time"
oneof	A set of names	internal, external

The **cnsview** command displays properties depending on their type and on the number of devices selected. When you select a single device, a scalar or list property displays as the value or list of values on a line by itself. For example:

```
show device
```

```
/dev/tty2
```

An aggregate displays with the subproperty name followed by a colon and then the value of the subproperty.

For example:

```
show errors

parity: 0
frame:  1
overrun:2
```

An aggregate qualified with a single subproperty displays like a scalar. For example:

```
show errors parity

2
```

When you select multiple devices, scalar values display with the device name and a colon preceding the value. For example:

```
set device /dev/tty2 /dev/tty3

show errors parity

 /dev/tty2:      2
/dev/tty3:      0
```

When you select multiple devices, an aggregate displays as a table. For example:

```
set device /dev/tty2 /dev/tty3

show errors

               parity  frame   overrun

/dev/tty2:       2       0       1
/dev/tty3:       0       1       0
```

Global Properties

The following Global Properties table describes the global properties of the **cnsview** command:

Global Properties			
Property	**Subproperty**	**Type**	**Description**
cnsview	**revision**	string	Current **cnsview** release
device		list	List of selected devices

Serial Port Properties

The following Serial Port Properties table describes the **cnsview** command properties and subproperties for serial ports:

Note: The **−x** flag only accepts the **comstatus**, **comstats**, **sysstats**, **memstats**, and **heapstats** subcommands.

Serial Port Properties			
Property	**Subproperty**	**Type**	**Description**

buddy		Boolean	Enable extended modem controls on port.
errors	**parity**	integer	Number of parity errors received.
	frame	integer	Number of frame errors received.
	overrun	integer	Number of overrun errors received.
extmodem	**dsr**	Boolean	State of the DSR modem control signal.
	ri	Boolean	State of the RI modem control signal.
	aloop	Boolean	State of the analog loopback modem control signal.
	dloop	Boolean	State of the digital loopback modem control signal.
hibaud		Boolean	Enable extended baud rates.
ldisc	**where**	oneof	Where line discipline is currently running:
			Host − line discipline on host.
			Remote − line discipline on 7318.
	hostld	Boolean	Force line discipline to run on host when True.
loopback		oneof	Enable special loopback modes:
			None − loopback not enabled.
			Internal − hardware loopback.
			Software − software loopback.
			Drain − flush outbound data.
modem	**dtr**	Boolean	State of the Data Terminal Ready (DTR) modem control signal.
	dcd	Boolean	State of the Data Carrier Detect (DCD) modem control signal.
	rts	Boolean	State of the Request to Send (RTS) modem control signal.
	cts	Boolean	State of the Clear to Send (CTS) modem control signal.
mscreen		Boolean	Enable multiple screens.
session		integer	Current virtual session number.
slew		oneof	Set slew rate on port:
			Slow − normal slew rate.
			Medium − medium slew rate.
			Fast − faster slew rate (default).
			Super − fastest slew rate.
tpparam	**rate**	integer	Transparent print rate in characters per second.
	enter	string	Terminal transparent print enter string.
	exit	string	Terminal transparent print exit string.

tprint		Boolean	Enable transparent printing.
traffic	**inbytes**	integer	Number of bytes received.
	outbytes	integer	Number of bytes transmitted.
	inmsgs	integer	Number of messages received.
	outmsgs	integer	Number of messages transmitted.
	ioctls	integer	Number of ioctls processed.
type		string	Terminal type in the Object Data Manager (ODM).
comstatus	**name**		Box port name.
	adminStatus		Administrative status, usually enabled.
	operStatus		Operating system, usually up.
	lastChange		Last time something changed on the port.
	inFlowType		Type of input flow control used.
	outFlowType		Type of output flow control used.
	inFlowState		Current state of output flow control.
	outFlowState		Current state of input flow control.
	inChars		Number of total input chars.
	outChars		Number of total output chars.
	adminOrigin		Connection type.
comstats	**ifInOctets**		# Characters in.
	ifOutOctets		# Characters out.
	ifInParityErrors		# of in parity errors.
	ifInFrameErrors		# of in frame errors.
	ifInOverruns		# of in overruns.
	ifOutMessages		
	ifInMessages		
	ifIoctls		
	ifDCDs		Shows DCD toggles
	ifDTRs		Shows DTR toggles
	ifRTSs		Shows RTS toggles
	ifCTSs		Shows CTS toggles
	ifDSRs		Shows DSR toggles
	ifRIs		
	ifALOOPs		

	ifDLOOPs		
	ifExcessiveDCDs		Show times port shut down for DCD.
	ifNoSRs		Show times we ran out of memory.
	ifOpens		Show times port was opened.
	ifInFlows		Shows # input flow ctl, any type.
	ifOutFlows		Shows # output flow ctl, any type.
	ifInBreaks		Shows # break conditions.
	ifOutBreaks		Shows # breaks sent.
	ifInSTREAMSOver		Shows # data loss because of no memory.
	ifUpTime		

Parallel Port Properties

The following Parallel Port Properties table describes the **cnsview** command properties and subproperties for 7318 LPT devices such as **/dev/lpt01**:

Note: The **−x** flag only accepts the **comstatus**, **comstats**, **sysstats**, **memstats**, and **heapstats** subcommands.

Parallel Port Properties			
Property	**Subproperty**	**Type**	**Description**
traffic	**inbytes**	integer	Number of bytes received.
	outbytes	integer	Number of bytes transmitted.
	inmsgs	integer	Number of messages received.
	outmsgs	integer	Number of messages transmitted.
	ioctls	integer	Number of ioctls processed.
lptctl	**selected**	Boolean	True if printer reports selected.
	peb	Boolean	True if printer reports paper empty or busy.
	error	Boolean	True if printer reports error.
	selection	Boolean	True if 7318 is asserting select.
	init	Boolean	True if 7318 is asserting init.
	autofeed	Boolean	True if 7318 is asserting feed.

7318 Properties

7318 properties apply to**/dev/cns***XX*, where the *XX* parameter is the number of the 7318 on the system. Most of these parameters show contents of the nonvolatile RAM in the 7318 and are not directly relevant to the 7318 software package.

> **Note:** The 7318 properties cannot be set using the **cnsview** command. They are for display only.

Properties 7318			
Property	**Subproperty**	**Type**	**Description**
bios	**loadtype**	string	Load protocol used.
	loadhost	string	Host loaded from.
	loadi	string	File name of 7318 image.
	selftest	integer	Hex value of self−test results.
cns	**addr**	string	Network address of 7318.
	psize	integer	Packet size in bytes.
	boot	Boolean	True if 7318 is bootable.
devmap	**type**	oneof	Specifies the porttype
			com
			lpt
	station	integer	7318 unit number
	port	integer	Physical port number
	session	integer	Session number
eaddr		string	Ethernet address.
hwconfig	**memtype**	integer	1MB or 4MB RAM chips.
	memsize	integer	7318 memory size in bytes.
	console	integer	Console number in use.
ipconfig	**ipaddr**	string	Internet address.
	domain	string	Simple Network Management Protocol (SNMP) domain.
ldparam	**loadfile**	string	7318 kernel file name.
	intr	string	Ethernet interface to load from.
	type	string	Type of load to perform.
	srch	Boolean	Search for load host.
	primary	string	Primary load host address.
	secondary	string	Secondary load host address.
	frametype	string	Specifies interface frametype. Possible values are: ETHERNET_II. ETHERNET_802.3. auto.
log		string	Dumb error log.
mac		string	Displays Mac Layer Interface (MAC) statistics.
serialnum	**box_sn**	string	Serial number of 7318.
	main_sn	string	Serial number of main board.

	io_sn	string	Serial number of I/O board.
	ps_sn	string	Serial number of power supply.
software		string	Displays model number and release version.
timezone		integer	Hours west of Greenwich.
sysstats	**model**		Type of box, P10 or S20.
	release		Operating kernal release.
	bios		Bios release.
	nvram		NVRAM release level.
	time		Time of day, may not be set.
	uptime		dd:hh:mm:ss of time since boot.
	idle		Ignore.
	idleRate		Ignore.
	dmaInts		# of dma interrupts, total.
	spintInts		# of clock/modem interrupts, total.
	macInts		# of LAN interrupts, total.
	badInts		# of unknown interrupts.
	maxDead		Highest deadman count, 10ms ticks.
memstats	**/dev/cns01**		
heapstats	**used**		
	free		
	available		
	heapMblks		
	heapMblksRefused		
	free16		
	free256		
	free4096		
	freeBig		
	used16		
	used256		
	used4096		
	usedBig		

Exit Status

This command returns the following exit values:

0 Successful completion.
>0 An error occurred.

Security

Access Control: You must have root authority to run this command.

Related Information

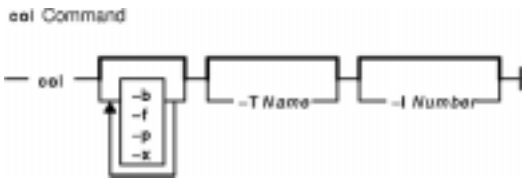
None

col Command

Purpose

Filters for standard output text having reverse line feeds and forward/reverse half−line−feeds.

Syntax



col [**−b**] [**−f**] [**−p**] [**−x**] [**−T** *Name*] [**−l***Number*]

Description

The **col** command reads a text file from standard input and writes to standard output. It performs the line overlays implied by the **flr** commands (reverse line feeds), as well as by the **hlf** and **hlr** commands (forward and reverse half−line−feed, respectively). The **nterm** file format document gives a description of these line−feed commands. Use the **col** command for filtering multicolumn output produced by the **nroff** command, the **.rt** request, and by output from the **tbl** command.

Use the **col** command as an **nroff** backend filter for devices that cannot handle reverse line motions (such as most impact printers). To print correctly, use the **col** command to process outputs from the **tbl** command, the **neqn** command, or explicit reverse motion request files (such as the **.sp −10V** file), or files with 2−column output. Do not process the **nroff** output targeted for the following devices with the **col** command:

- **hplj**
- **ibm4019**
- **ibm5577**
- **ibm5575**

Unless the **−x** flag is given, whenever possible, the **col** command converts white spaces to tabs upon output wherever possible to shorten printing time.

The **col** command, used with the **−T37** file, assumes the ASCII control characters, SO (\017) and SI (\016), begin and end text in an alternate character set. The **col** command remembers the character set each input character belongs to and upon output, generates SI and SO characters as appropriate to ensure that each character is printed in the correct character set.

Upon input, the **col** command accepts only the control characters for the Space, Backspace, Tab, and Return keys; the new−line character; the SI, SO (with the **−T37** file), and VT control characters; and the reverse line feed, forward half−line−feed and reverse half−line−feed characters. The VT control character (\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 nonprinting characters.

Normally, the **col** command ignores any escape sequences that are unknown to it and found in the input. However, the **−p** option can be used to cause the **col** command to output these sequences as regular characters, subject to overprinting from reverse line motions. The use of this option is highly discouraged unless the user is fully aware of the textual position of the escape sequences.

Notes:

1. If the output is being sent to a device that can interpret half−line motions, enter:

   ```
   nroff −Tppds File... | col −f −Tppds
   ```

   Otherwise, for example, enter:

   ```
   nroff −Tlp File... | col −Tlp
   ```

2. The maximum number of lines that can be backed up is 128.
3. No more than 800 characters, including backspaces, are allowed on a line.
4. 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 be displayed in the same position, only the last one that is read is displayed 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 can contain forward half−line−feeds (**hlf**) but not reverse line feeds (**flr** or **hlr**).

−p Displays unknown escape sequences as characters, subject to overprinting from reverse line motions. Normally, the **col** command ignores them.

−x Converts tabs to white space.

−T*Name* Uses the workstation specification indicated by the *Name* variable. *Name* varables for "Terminal Names for Typewriter−like Devices and Line Printers" are discussed in the **nroff** command **−T***Name* flag. The default is **37**.

−l*Number* (lowercase L) Sends the specified number lines of text in memory to a buffer during processing.

Exit Status

The following exit values are returned:

0 Indicates successful completion.

>0 Indicates an error occurred.

Related Information

The **hplj** command, **mm** command, **nroff** command, **ps4014** command, **tbl** command.

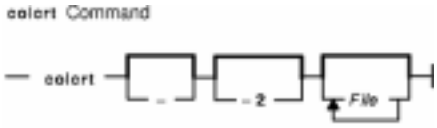
The **nterm** file format.

colcrt Command

Purpose

Filters **nroff** command output for cathode ray tube (CRT) previewing.

Syntax



colcrt [**−**] [**−2**] [*File* ...]

Description

The **colcrt** command filters output from the **nroff** command so that the output can be previewed on a CRT. The **colcrt** command provides virtual half−line−feed and reverse line−feed sequences for terminals without these capabilities. The **colcrt** command changes underline characters to dashes and places these characters and the half−line characters on new lines between the normal output lines.

Notes:

1. Use this command with the **37** viewing device
2. The **−** (minus sign) flag removes all underlining; therefore, a true underline character does not show with the **−** (minus sign) flag.
3. It is not possible to back up more than 102 lines.
4. General overstriking is lost. As a special case, the | (vertical bar) overstruck with the − (dash) or the _ (underline) characters becomes the + (plus sign).
5. Lines are truncated to up to 132 characters.

Parameters

File Specifies a file processed by the **nroff** command for viewing on a CRT.

Flags

− Suppresses underlining. This flag is useful for previewing boxed tables from the **tbl** command.

−2 Causes all half−lines to be printed, effectively double−spacing the output. This is useful when printing output with subscripts and superscripts on a line printer, where half−lines normally are not displayed.

Examples

A typical use of the **colcrt** command is:

```
tbl exum2.n | nroff −ms −T37 | colcrt − | pg
```

Related Information

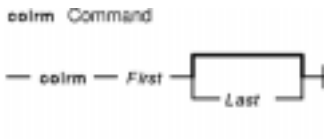
The **col** command, **nroff** command, **pg** command, **tbl** command, **troff** command, **ul** command.

colrm Command

Purpose

Extracts columns from a file.

Syntax



colrm *First* [*Last*]

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, the columns of each line are removed starting with the specified column. If called with two parameters, the columns from the first column to the last column are removed.

Column numbering starts with column 1.

Examples

To remove columns from the `text.fil` file, enter:

```
colrm 6 < text.fil
```

If `text.fil` contains:

```
123456789
```

then the **colrm** command displays:

```
12345
```

Files

/usr/bin/colrm Contains the **colrm** command.

Related Information

The **cut** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* introduces you to files and the way you can work with them.

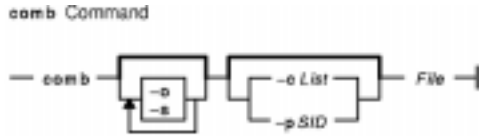
Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how the AIX Operating System processes input and output.

comb Command (SCCS)

Purpose

Combines SCCS deltas.

Syntax



comb [**−o**] [**−s**] [**−c** *List* | **−p** *SID*] *File*

Description

The **comb** command writes to standard output a shell procedure that can combine specified SCCS deltas (SIDs) or all deltas into one delta. You can reduce the size of your Source Code Control System (SCCS) file by running the resulting procedure on the file. To see how much the file will be reduced, run the **comb** program with the **−s** flag. If you specify a directory for the *File* value, the **comb** command performs the requested actions on all SCCS files (that is, those having an **s.** prefix). If you specify a **−** (minus) for the *File* value, 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 an end−of−file character.

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.

> **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

> **Note:** 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 will preserve (see the **get** command **−i***List* flag). The procedure combines all other deltas.

−o Accesses the reconstructed file at the release of the delta to be created for each **get** command **−e** flag 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 for each file listing: the file name, size (in blocks) after combining, original size (also in blocks), and percentage change computed by the formula:

```
100 * (original − combined) / original
```

You should run the **comb** program using this flag and then run its procedure before combining SCCS files in order to judge how much space will actually be saved by the combining process.

Examples

1. To generate a report on how much space will be saved by combining all deltas older than SID `1.4` for sccs file `s.test.c`, enter:

   ```
   comb -p1.4 -s s.test.c
   ```

   Run the report by piping the output of the above command to the **sh** command.

2. To actually perform the combination, enter:

   ```
   comb -p1.4 s.test.c
   ```

Files

s.COMB The name of the reconstructed SCCS file.
comb* Temporary files.

Related Information

The **admin** command, **delta** command, **get** command, **prs** command, **sccshelp** command, **sh** command.

The **sccsfile** file format.

List of SCCS Commands in *AIX General Programming Concepts: Writing and Debugging Programs*.

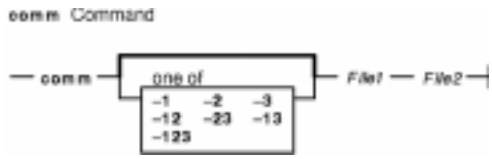
Source Code Control System (SCCS) Overview in *AIX General Programming Concepts: Writing and Debugging Programs*.

comm Command

Purpose

Selects or rejects lines common to two sorted files.

Syntax



comm [**−1 −2 −3**] *File1 File2*

Description

> **Note:** If you specify − (minus) for one of the file names, the **comm** command reads standard input.

The **comm** command reads the *File1* and *File2* parameters and writes, by default, a three−column output to standard output. The columns consist of:

- Lines that are only in *File1*
- Lines that are only in *File2*
- Lines that are in both *File1* and *File2*.

Both *File1* and *File2* should be sorted according to the collating sequence specified by the current National Language environment.

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*).

Exit Status

This command returns the following exit values:

0 All input files were output successfully.

>0 An error occurred.

Examples

1. To display the lines unique to each file and common to both, enter:

   ```
   comm things.to.do things.done
   ```

   If the files `things.to.do` and `things.done` contain the following lists:

```
things.to.do

buy soap
groceries
luncheon
meeting at 3
system update
tech. review

things.done

2nd revision
interview
luncheon
system update
tech. review
weekly report
```

then the **comm** command 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 `things.to.do`. The second column, indented with a tab character, lists the lines found only in `things.done`. The third column, indented with two tabs, lists the lines common to both.

2. To display the lines that appear in only one file, enter:

   ```
   comm -23 things.to.do things.done
   ```

   This suppresses the second and third columns of the **comm** command listing. If the files are the same as in Example 1, then the following is displayed:

   ```
   buy soap
   groceries
   meeting at 3
   ```

File

/usr/bin/comm Contains the **comm** command.

Related Information

The **cmp** command, **diff** command, **sdiff** command, **sort** command, **uniq** command.

The **environment** file.

Understanding Locale in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

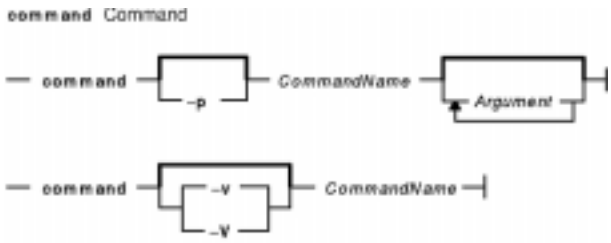
National Language Support Overview for Programming in *AIX General Programming Concepts: Writing and Debugging Programs*.

command Command

Purpose

Executes a simple command.

Syntax



command [**−p**] *CommandName* [*Argument* ...]

command [**−v** | **−V**] *CommandName*

Description

The **command** command causes the shell to treat the specified command and arguments as a simple command, suppressing shell function lookup.

Normally, when a / (slash) does not precede a command (indicating a specific path), the shell locates a command by searching the following categories:

1. special shell built−ins
2. shell functions
3. regular shell built−ins
4. **PATH** environment variable

For example, if there is a function with the same name as a regular built−in, the system uses the function. The **command** command allows you to call a command that has the same name as a function and get the simple command.

The **command −v** and **command −V** commands write to standard output what path name will be used by the shell and how the shell interprets the command type (built−in, function, alias, and so forth). Since the **−v** and **−V** flags produce output in relation to the current shell environment, the **command** command is provided as a Korn shell or POSIX shell regular built−in command. The **/usr/bin/command** command might not produce correct results, because it is called in a subshell or separate command execution environment,. In the following example the shell is unable to identify aliases, subroutines, or special shell commands:

```
(PATH=foo command −v)
nohup command −v
```

Flags

−p Performs the command search using a default value for the **PATH** environment variable that is finds all of the standard commands.

−v Writes to standard output the path name used by the current shell to invoke the specified command, according to the following conventions:

- Commands, regular built−in commands, commands including a / (slash), and any implementation−provided functions found by the **PATH** environment variable are written as absolute path names.
- Shell functions, special built−in commands, regular built−in commands not associated with a **PATH** environment variable search, and shell reserved words are written as just their names.
- Aliases are identified as such, and their definitions are included in the string.

If the specified command name cannot be found, no output is written and the exit status returns a **>0** value.

−V Writes to standard output the command name that will be interpreted by the current shell environment. Although the format of this output is unspecified, The output indicates in which of the following categories the command falls:

- Commands, regular shell commands, and any implementation−provided subroutines found using the **PATH** environment variable are identified as such and written as absolute path names.
- Other shell functions are identified as functions.
- Aliases are identified as such, and their definitions are included in the string.
- Special built−in commands are identified as such.
- Regular built−in commands not associated with the **PATH** environment variable search are identified as such.
- Shell reserved words are identified as such.

Exit Status

When the **−v** or **−V** flag is specified, the following exit values are returned:

0 Successful completion.

>0 The command specified with the *CommandName* parameter could not be found or an error occurred.

When the **−v** or **−V** flag is not specified, the following exit values are returned:

126 The command specified by the *CommandName* parameter was found but could not be invoked.

127 An error occurred in the **command** command, or the command specified by the *CommandName* parameter could not be found.

Otherwise, the **command** command returns the exit status associated with the command specified by the *CommandName* parameter.

Examples

1. To make a version of the **cd** command that prints out the new working directory whenever you change directories, enter:

```
cd () {
        command cd "$@" >/dev/null
        pwd
}
```

2. To start off a secure shell script, one in which the script avoids being spoofed by its parent, enter:

```
IFS='
'
#       The preceding value should be <space><tab><newline>.
#       Set IFS to its default value
```

```
\unalias -a
#       Unset all possible aliases.
#       Note that unalias is escaped to prevent an alias
#       being used for unalias.

unset -f command
#       Ensure command is not a user function.

PATH="$(command -p getconf _CS_PATH):$PATH"
#       Put on a reliable PATH prefix.

# ...
```

At this point, given correct permissions on the directories called by the **PATH** environment variable, the script has the ability to ensure that any command it calls is the intended one.

Files

/usr/bin/ksh	Contains the Korn shell **command** built-in command.
/usr/bin/command	Contains the **command** command.

Related Information

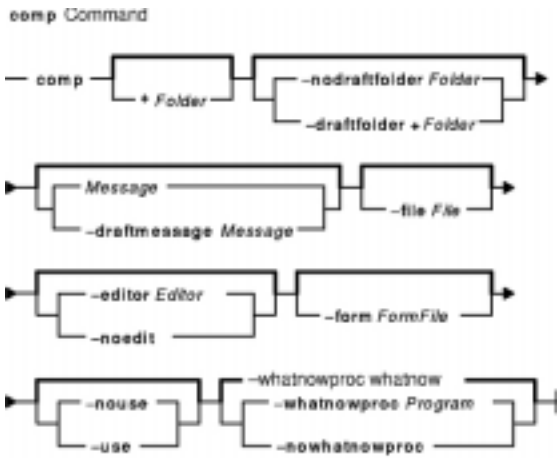
The **ksh** command, **type** command.

comp Command

Purpose

Composes a message.

Syntax



comp [+*Folder*] [−**draftfolder** +*Folder* | −**nodraftfolder** *Folder*]
[*Message* | −**draftmessage** *Message*] [−**file***File*] [−**editor** *Editor* | −**noedit**] [−**form** *FormFile*]
[−**use** | −**nouse**] [−**nowhatnowproc** | −**whatnowproc** *Program*]

Description

The **comp** command starts an editor that assists you in creating and modifying messages. The **comp** command provides a header template, the **/etc/mh/components** file. By default, the specified editor creates a *UserMhDirectory*/**draft** file. If a **draft** file already exists, the **comp** command prompts you for permission to replace or use the existing file. To edit an existing **draft** file without being prompted for permission, specify the −**use** flag.

Once started, the editor prompts you to enter values for each of the message header fields. The **comp** command uses the definitions in the *UserMhDirectory*/**components** file for the header fields. If the file does not exist, the **/etc/mh/components** file is used. You can use the −**form** or +*Folder* flag to specify an alternative header format.

To exit the editor, use the Ctrl−D sequence. When you exit the editor, the **comp** command responds with a `What now?` prompt. From this prompt, you can specify any of the **whatnow** subcommands. To see a list of the available subcommands, press Enter. You can use the subcommands to continue composing the message, direct the disposition of the message, or end the processing of the **comp** command.

> **Note:** A 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.

The −**file**, −**draftfolder**, and −**draftmessage** flags are used to specify existing draft messages. If the −**draftfolder** +*Folder* flag is followed by a *Message* parameter, it is the same as specifying the −**draftmessage** flag. If you wish, you can define a default `Draft−Folder:` entry in your Message Handler (MH) **$HOME/.mh_profile** file.

Flags

−draftfolder +*Folder*
Identifies the folder containing the draft message. If a message is not specified with this flag, the default message is **new**.

−draftmessage *Message*
Identifies the draft message. Specifying a *Message* variable after the **−draftfolder** +*Folder* flag is the same as specifying the **−draftmessage** flag.

−editor *Editor*
Specifies the initial editor for composing the message. If you do not specify the **−editor** flag, the **comp** command selects the default editor specified by the `Editor:` entry in your **$HOME/.mh_profile** file.

−file *File*
Places the draft message in the specified file. If you do not specify the absolute path name for the *File* variable, the **comp** command places the file in the user's MH directory. If a file is specified, the **comp** command prompts you for the disposition of the draft.

+*Folder Message*
Uses the header information from a file in the specified folder. If you specify a folder but no message, the **comp** command uses the current message as the default.

−form *FormFile*
Uses the header fields specified by the *FormFile* variable. The **comp** command treats each line in *FormFile* as a format string.

−help
Lists the command syntax, available switches (toggles), and version information.
Note: For MH, the name of this flag must be fully spelled out.

Message
Specifies a message. Use the following references to specify a message:

Number
Number of the message.

cur *or* **.** *(period)*
Current message. This is the default.

first
First message in a folder.

last
Last message in a folder.

next
Message following the current message.

prev
Message preceding the current message.

−nodraftfolder
Places the draft in the *UserMhDirectory*/**draft** file. This is the default.

−noedit
Suppresses the initial edit. When you specify this flag, you receive the `What now?` prompt.

−nouse
Creates a new message.

−nowhatnowproc
Prevents interaction with the editor and the `What now?` prompt.

−use
Continues composing an existing draft of a message.

−whatnowproc *Program*
Starts the specified program to guide you through the composing tasks. If you specify the **whatnow** command as the value for the *Program* variable, the **comp** command starts an internal **whatnow** procedure, instead of a program with the file name **whatnow**.

Profile Entries

The following entries are entered in the *UserMhDirectory*/**.mh_profile** file:

`Draft−Folder:` Sets the default folder for drafts.

`Editor:` Sets the default initial editor.

`fileproc:` Specifies the program used to refile messages.

`Msg−Protect:` Sets the protection level for the new message files.

`Path:`	Specifies a user's MH directory.
`whatnowproc:`	Specifies the program used to prompt `What now?` questions.

Examples

1. To compose a new message, enter:

   ```
   comp
   ```

   The system prompts you to enter the information for the message fields. To bypass a field, press the Enter key. When the header information is complete, enter the text for the body of the message.

   To finish composing a message and exit the editor, press the Ctrl−D sequence. The following prompt is displayed on your screen:

   ```
   What now?
   ```

   Pressing the Enter key displays a list of the **whatnow** subcommands. If you want to send the message, enter the **send** subcommand after the `What now?` prompt.

2. To compose a new message using the `vi` editor, enter:

   ```
   comp −editor vi
   ```

3. To compose a message using message `8` in the `schedules` folder, enter:

   ```
   comp +schedules 8 −use
   ```

4. To compose a message using a message draft in the `/home/mike/parts` file, enter:

   ```
   comp −file /home/mike/parts
   ```

The system prompts you for the disposition of the file. Press the Enter key for a list of options. Select the appropriate option.

Files

UserMhDirectory/**components**	Specifies the user's default message format. (If it exists, it overrides the system default message format.)
UserMhDirectory/**draft**	Contains the current draft message.
$HOME/.mh_profile	Specifies the user's MH profile.
/etc/mh/components	Identifies the system default message format.
/usr/bin/comp	Contains the **comp** command.

Related Information

The **ali** command, **dist** command, **forw** command, **refile** command, **repl** command, **send** command, **whatnow** command, **whom** command.

The **mh_alias** file format, **mh_profile** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

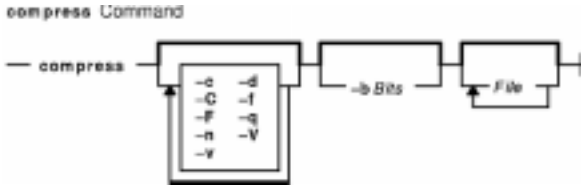
Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

compress Command

Purpose

Compresses data.

Syntax



compress [**−c**] [**−C**] [**−d**] [**−F**] [**−f**] [**−n**] [**−q**] [**−v**] [**−V**] [**−b***Bits*] [*File ...*]

Description

The **compress** command compresses data, using adaptive Lempel−Zev coding to reduce the size of files. Each original file specified by the *File* parameter is replaced when possible by a compressed file with a **.Z** appended to its name. The compressed file retains the same ownership, modes, and modification time of the original file. If no files are specified, the standard input is compressed to the standard output. If compression does not reduce the size of a file, a message is written to standard error and the original file is not replaced.

> **Note:** Files must have correct permissions to be replaced.

The amount of compression depends on the size of the input, the number of bits per code specified by the *Bits* variable, and the distribution of common substrings. Typically, source code or English text is reduced by 50 to 60%. The compression of the **compress** command is usually more compact and takes less time to compute than the compression achieved by Huffman coding (as used in the **pack** command) or adaptive Huffman coding.

Flags

−b *Bits* Specifies the maximum number of bits to use to replace common substrings in the file. The value of the *Bits* variable must be in the range from 9 bits through 16 bits, with the default being 16 bits. When compressing data, the algorithm first uses all of the 9−bit codes (257 through 512) to replace as many substrings as possible. Then it uses all 10−bit codes, and so on, continuing until the limit specified by the **−b** flag is reached.

−c Writes to standard output. No files are changed.

−C Produces output compatible with the Berkeley Software Distribution (BSD) Revision 2.0.

−d Causes the **compress** command to function exactly like the **uncompress** command.

−f or **−F** Forces compression. The **−f** and **−F** flags are interchangeable. Overwrites the *File***.Z** file if it already exists.

After the value of the *Bits* variable is attained, the **compress** command periodically checks the compression ratio. If it is increasing, the **compress** command continues to use the existing code dictionary. However, if the compression ratio decreases, the **compress** command discards the table of substrings and rebuilds it. Rebuilding the table allows the algorithm to adapt to the next block of

the file.

−n Omits the compressed file header from the compressed file.

−q Suppresses the display of compression statistics generated by the **−v** flag. If several **−v** and **−q** flags are on the same command line, the last one specified controls the display of the statistics.

−v Writes the percentage of compression.

−V Writes the current version and compile options to standard error.

Parameters

File Specifies the file to compress.

Return Values

If an error occurs, the exit status is 1. If the **compress** command exits without compressing a file, it exits with a status of 2. Otherwise, the **compress** command exits with a status of 0.

The **compress** command detects an error and exits with a status of 1 if any of the following events occur:

- An input file is not a regular file.
- An input file name is too long to append the **.Z** extension.
- An input file cannot be read or an output file cannot be written.

Exit Status

0 Successful completion.

1 An error occurred.

2 One or more files were not compressed because they would have increased in size (and the **−f** flag was not specified).

>2 An error occurred.

Example

To compress the `foo` file and write the percentage of compression to standard error, enter:

```
compress −v foo
```

The `foo` file is compressed and renamed `foo.Z`.

Related Information

The **compress** command uses the modified Lempel−Zev algorithm described in "A Technique for High Performance Data Compression," Welch, Terry A. *IEEE Computer*, vol. 17, no. 6 (June 1984), pp. 8−19.

The **pack** command, **uncompress** command, **unpack** command, **zcat** command.

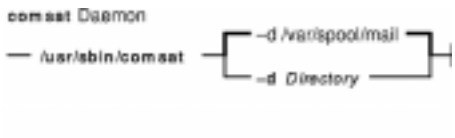
Commands Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

comsat Daemon

Purpose

Notifies users of incoming mail.

Syntax



/usr/sbin/comsat [**−d***Directory*]

Description

The **comsat** daemon is the server that receives reports of incoming mail and notifies users if they have enabled this service with the **biff** command. Started by the **inetd** daemon, the **comsat** daemon is not meant to be used at the command line. The **comsat** daemon 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 has run the **biff y** command, the first 7 lines or 560 characters of the message are displayed on the user's login terminal. Lines that appear to be part of a message header other than the `From:` or `Subject:` lines are not included in the displayed message.

Flags

−d*Directory* Specifies the name of the directory to use as the system mail directory. If the **−d** flag is not specified, the **comsat** daemon uses the **/var/spool/mail** directory as the default system mail directory.

Files

/etc/utmp Contains a list of users who are logged in, including their terminals.

/etc/services Contains a list of Internet network services and the well−known ports where the servers accept connections.

Related Information

The **biff** command.

The **inetd** daemon.

The **inetd.conf** file format.

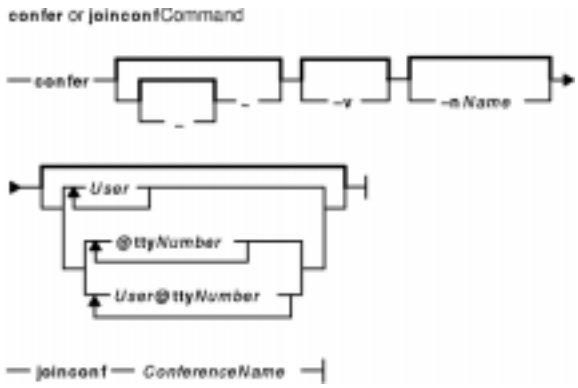
Mail Overview for System Management in *AIX Version 4.3 System Management Guide: Communications and Networks*.

confer or joinconf Command

Purpose

Provides an online conferencing system.

Syntax



confer [[−] **~**] [**−v**] [**−n***Name*] [*User* ... | **@tty***Number* ... | *User***@tty***Number* ...]

joinconf*ConferenceName*

Description

The **confer** command sets up an online, written conference among logged−in users on your local node. To begin a conference, enter the **confer** command at the command prompt.

If you specify the *User* parameter, the user may join the conference from any workstation. That user is notified of the conference at each workstation the user is logged in to. If you specify the **@tty***Number* parameter, any user at the specified workstation may join the conference. If you specify the *User***@tty***Number* parameter, the specified user may join the conference only on the specified workstation.

If the users you specify are logged in and their workstations are writable, they may join the conference by typing the **joinconf** command on their command line. The **joinconf** command informs other conferees as each user joins the conference.

Note: Consoles and pseudoterminals (pts) may be specified with the **@tty***Number* parameter.

The **confer** command gives each conference a unique name. By default, it has the name of the conference leader, with additional letters if necessary. The conference leader can override this default by specifying the **−n***Name* flag.

Unless the conference leader sets up a confidential conference by specifying the **~** (tilde) flag, the **confer** command makes a transcript of the conference. When a user leaves the conference, the **confer** command asks if the user needs a transcript. If so, the system mails the user a copy when the conference concludes. Any participant can make an off−the−record comment in a conference that is otherwise on the record by beginning the line with the **~** flag.

You can run shell commands from within a conference by prefixing them with a | (vertical bar) or an ! (exclamation point). The exclamation point causes the command to run normally; the output is displayed only

at the workstation that runs it. The vertical bar causes the command and all of its standard output and standard error output to become part of the conference, visible to all conferees.

Conferences

Once you join a conference, the text you enter at your workstation displays at all other workstations participating in the conference. You can end your participation in a conference by pressing the Ctrl−D key sequence. This action causes your user ID and the word BYE to be displayed at the workstations of the other conference participants. You will continue to see displayed conference contributions on your workstation until you are excused by each of the active participants.

Conference contributions are normally transmitted one line at a time. If the conference leader specifies the **−v** flag, transmissions are issued one character at a time. This mode sends all user typing errors and hesitations, and imposes a considerably larger load on the system.

> **Note:** Contributions can contain multibyte characters.

Protocol

To prevent confusion caused by several conferees typing at the same time, users should follow some agreed−on protocol. The following is one recommended protocol:

- In order to take the floor, a user presses the Enter key before entering a contribution. This notifies other participants that the user has the floor because the user's name displays in brackets at their respective workstations.
- A user is presumed to have the floor until the user relinquishes it by entering a blank line.
- If two or more users try to claim the floor at the same time, the last person to do so (the one whose name appears last), is assumed to have the floor. The others should immediately relinquish the floor by typing single blank lines.

Subcommands

!excuse [*User* ... | **@tty***Number* ... | *User***@tty***Number* ...]
: Excuses the specified conferees or workstations from the conference. No further conference material is displayed at these workstations.

!~
: Places all contributions from the user who issues the **!~** subcommand off the record until the **!~~** subcommand is issued.

!~~
: Cancels a preceding **!~** subcommand, placing the user's remarks back on the record.

Flags

−n*Name* Assigns a name to the conference. The conference name is used by those users joining the conference so that they can access the right one. The name of the user who starts the conference is the default conference name. The *Name* variable cannot be more than 8 bytes long.

−v Transmits conference messages one character at a time.

~ Sets up the conference off the record. No transcript is recorded. A leading dash is optional with this flag.

Examples

1. To start a conference with users `jeanne` and `ron`, enter:

   ```
   confer jeanne ron
   ```

Running the **confer** command makes you the conference leader, so your login name is also the name of the conference. The **confer** command sends users `jeanne` and `ron` a message inviting them to join your conference and giving them the conference name.

2. To specify workstations that can join the conference, enter:

   ```
   confer jeanne@tty5 @tty10 ron
   ```

   If user `jeanne` is logged in at workstations `tty3`, `tty4`, and `tty5`, she can join the conference from workstation `tty5` only. If user `ron` is logged in at `tty7` and `tty8`, he can join the conference from either `tty7` or `tty8`, or from both. The user logged in at `tty10` is also invited to join the conference.

3. The user at workstation `tty10` decides not to join the conference started by the command in example 2. To excuse this user at `tty10` from the conference, each active participant must enter:

   ```
   !excuse @tty10
   ```

   Although not actively participating in the conference, the user at `tty10` sees the conference dialog on the workstation display until excused by each active conference participant.

4. To join a conference named `karen`, enter:

   ```
   joinconf karen
   ```

   Any text typed becomes part of the dialog, prefixed with the name `Karen`, and displayed at each participant's workstation. Contributions by user `Karen` will also be recorded in the transcript of the conference.

5. If user `ron` decides to join the conference from `tty7`, the conference dialog is also displayed at his other workstation `tty8`, unless everyone (including user `ron`) enters:

   ```
   !excuse ron@tty8
   ```

   User `ron` should enter the **!excuse** subcommand from `tty7`, the workstation he is using for the conference.

6. To make a single−line statement off the record, enter:

   ```
   ~Coffee and donuts at my place.
   ```

   The **confer** command displays lines beginning with a ~ (tilde) at participants' workstations, but does not include them in the record of the conference.

7. To make a multiple−line statement off the record, enter:

   ```
   !~
   Everyone is invited
   to my place after the conference
   for coffee and donuts.
   !~~
   ```

8. To run a shell command privately, without leaving the conference, enter:

   ```
   !li
   ```

   This action lists the current directory without including the **li** command or its output in the conference.

9. To include the output of a shell command in the discussion, enter:

```
|cat notes.conf
```

This action lists the contents of the **notes.conf** file at each participant's workstation, and includes it in the conference record.

10. To send command output to others, off the record, enter:

```
!~|
cat notes.conf
!~~
```

11. To leave the conference, press the Ctrl−D key sequence. If your user name is `karen`, after you press the Ctrl−D key sequence, the message `[karen] BYE` is sent to the other participants. The rest of the discussion continues to be displayed at `karen`'s workstation until each of the other participants enters:

```
!excuse karen
```

Files

/etc/utmp Contains a list of logged−in users.
/dev/tty?? Contains a list of workstation names.
/tmp/*.cnf Contains a list of user transcript files.
/tmp/*.mls Contains the transcript mailing list.

Related Information

The **talk** command, **write** command.

conflict Command

Purpose

Searches for alias and password conflicts.

Syntax

conflict [−**mail***User*] [−**search***Directory* ...] [*File* ...]

Description

The **conflict** command finds invalid mail drops and alias conflicts. The **conflict** command is not started by the user. The **conflict** command is called by the **cron** daemon and other programs used for system accounting. However, root user authority and the full path name of the command, **/usr/lib/mh/conflict**, are required to invoke the program.

The **conflict** command searches specified mail drop directories for mailbox files with names that do not correspond to valid users in the **/etc/passwd** file. In addition, the program searches alias files specified by the *File* parameter for duplicate names that do not resolve to the same address. By default, the **conflict** command searches the **/etc/mh/MailAliases** file.

The **conflict** command also searches entries in the group file (**/etc/group**) for invalid user names and users who do not have a valid group number.

Command output is to the monitor unless you specify the −**mail** flag. The −**mail** flag sends the command output to the specified user.

Flags

−**help**	Lists the command syntax, available switches (toggles), and version information. **Note:** For Message Handler (MH), the name of this flag must be fully spelled out.
−**mail** *User*	Sends the results of the **conflict** command to the user specified by the *User* variable.
−**search** *Directory*	Searches the directory indicated by the *Directory* variable for mailboxes that are not valid. You can specify any number of −**search** flags. The default mailbox directory is **/var/spool/mail**.

Files

/etc/mh/MailAliases	Contains the default mail alias file.
/etc/passwd	Contains a list of users.
/etc/group	Contains a list of groups.

/var/spool/$USER	The mail drop for the user **$USER.**
/$HOME/.mh_profile	Contains the MH user profile.
/etc/mh/mtstailor	Contains MH command definitions.

Related Information

The **ali** command, **whom** command.

The **mh_alias** file format.

Mail Overview in *AIX Version 4.3 System User's Guide: Communications and Networks*.

Peek, Jerry. *MH and xmh: E−mail for Users and Programmers.* Sebastopol, CA: O'Reilly & Associates, 1992.

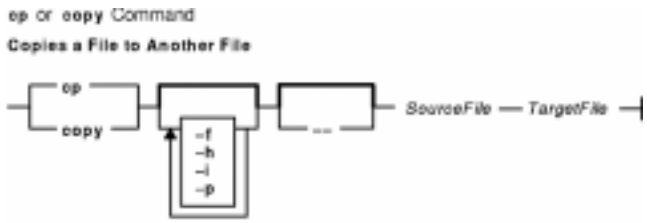
cp or copy Command

Purpose

Copies files.

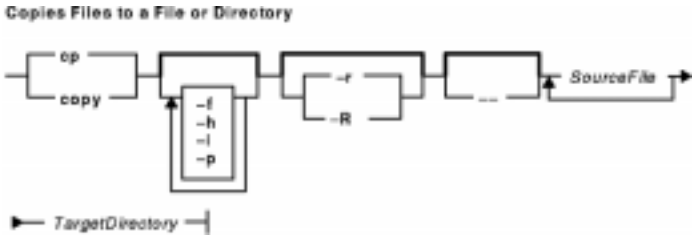
Syntax

To Copy a File to another File



{ **cp** | **copy** } [**−f**] [**−h**] [**−i**] [**−p**][**−−**] *SourceFile TargetFile*

To Copy a File to a Directory



{ **cp** | **copy** } [**−f**] [**−h**] [**−i**] [**−p**] [**−r** | **−R**] [**−−**] *SourceFile ... TargetDirectory*

To Copy a Directory to a Directory

{ **cp** | **copy** } [**−f**] [**−h**] [**−i**] [**−p**] [**−−**] { **−r** | **−R** } *SourceDirectory ... TargetDirectory*

Description

The **cp** command copies the source file specified by the *SourceFile* parameter to the destination file specified by the *TargetFile* parameter. Also, the **cp** command can copy the source files specified by the *SourceFile* parameter (or directories named by the *SourceDirectory* parameter) to the directory specified by the *TargetDirectory* parameter.

> **Note:** If one of the source parameters is a directory, you need to specify one of the **−r** or **−R** flags.

If any directories are created by the **cp** command during the copying process, the newly created directory will

have the same mode as the corresponding source directory.

You can also copy special device files. The preferred option for accomplishing this is the **−R** flag. Specifying **−R** causes the special files to be re−created under the new path name. Specifying the **−r** flag causes the **cp** command to attempt to copy the special file to a regular file.

Flags

−f Specifies removal of the target file if it cannot be opened for write operations. The removal precedes any copying performed by the **cp** command.

−h Forces the **cp** command to copy symbolic links. The default is to follow symbolic links, that is, to copy files to which symbolic links point.

−i Prompts you with the name of a file to be overwritten. This occurs if the *TargetDirectory* or *TargetFile* parameter contains a file with the same name as a file specified in the *SourceFile* or *SourceDirectory* parameter. If you enter `y` or the locale's equivalent of `y`, the **cp** command continues. Any other answer prevents the **cp** command from overwriting the file.

−p Duplicates the following characteristics of each SourceFile/SourceDirectory in the corresponding *TargetFile* and/or *TargetDirectory*:

- The time of the last data modification and the time of the last access. If this duplication fails for any reason, the **cp** command will write a diagnostic message to standard error.
- The user ID and group ID. If this duplication fails for any reason, the **cp** command may write a diagnostic message to standard error.
- The file permission bits and the S_ISUID and S_ISGID bits. If this duplication fails for any reason, the **cp** command will write a diagnostic message to standard error.

If the user ID or group ID cannot be duplicated, the file permission bits S_ISUID and S_ISGID will be cleared.

The target file will not be deleted if these characteristics cannot be preserved.

Access control lists (ACLs) associated with the *SourceFile* are also preserved. See "Managing Protected Resources with Access Control" in *AIX Version 4.3 System User's Guide: Operating System and Devices* to learn more about ACLs.

−r Copies file hierarchies under the file or directory specified by the *SourceFile* or *SourceDirectory* parameter (recursive copy). The **−r** flag processes special files in the same manner as regular files.

−R Copies file hierarchies under the regular files and directories from the directory specified by the *SourceFile* or *SourceDirectory* parameter to the directory specified by the *TargetDirectory* parameter. Special file types, such as first−in, first−out (FIFO) files and block and character device files, are re−created instead of copied. Symbolic links are followed unless the **−h** flag is specified. (The **−R** flag is preferred to the **−r** flag.)

−− Indicates that parameters following the **−−** (dash, dash) flag are to be interpreted as file names. This null flag allows the specification of file names that start with a − (minus sign).

Exit Status

This command returns the following exit values:

0 All files were copied successfully.

>0 An error occurred.

Examples

1. To make a copy of a file in the current directory, enter:

   ```
   cp prog.c prog.bak
   ```

   This copies `prog.c` to `prog.bak`. If the `prog.bak` file does not already exist, the **cp** command creates it. If it does exist, the **cp** command replaces it with a copy of the `prog.c` file.

2. To copy a file in your current directory into another directory, enter:

   ```
   cp jones /home/nick/clients
   ```

   This copies the `jones` file to `/home/nick/clients/jones`.

3. To copy a file to a new file and preserve the modification date, time, and access control list associated with the source file, enter:

   ```
   cp −p smith smith.jr
   ```

   This copies the `smith` file to the `smith.jr` file. Instead of creating the file with the current date and time stamp, the system gives the `smith.jr` file the same date and time as the `smith` file. The `smith.jr` file also inherits the `smith` file's access control protection.

4. To copy all the files in a directory to a new directory, enter:

   ```
   cp /home/janet/clients/* /home/nick/customers
   ```

   This copies only the files in the `clients` directory to the `customers` directory.

5. To copy a directory, including all its files and subdirectories, to another directory, enter:

   ```
   cp −R /home/nick/clients /home/nick/customers
   ```

   This copies the `clients` directory, including all its files, subdirectories, and the files in those subdirectories, to the `customers/clients` directory.

6. To copy a specific set of files to another directory, enter:

   ```
   cp jones lewis smith /home/nick/clients
   ```

   This copies the `jones`, `lewis`, and `smith` files in your current working directory to the `/home/nick/clients` directory.

7. To use pattern−matching characters to copy files, enter:

   ```
   cp programs/*.c .
   ```

This copies the files in the `programs` directory that end with `.c` to the current directory, signified by the single `.` (dot). You must type a space between the `c` and the final dot.

Files

/usr/bin/cp Contains the **cp** command.
/usr/bin/copy Contains the **copy** command.

Related Information

The **cpio** command, **link** command, **ln** command, **mv** command, **unlink** command.

File Systems and Directories Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

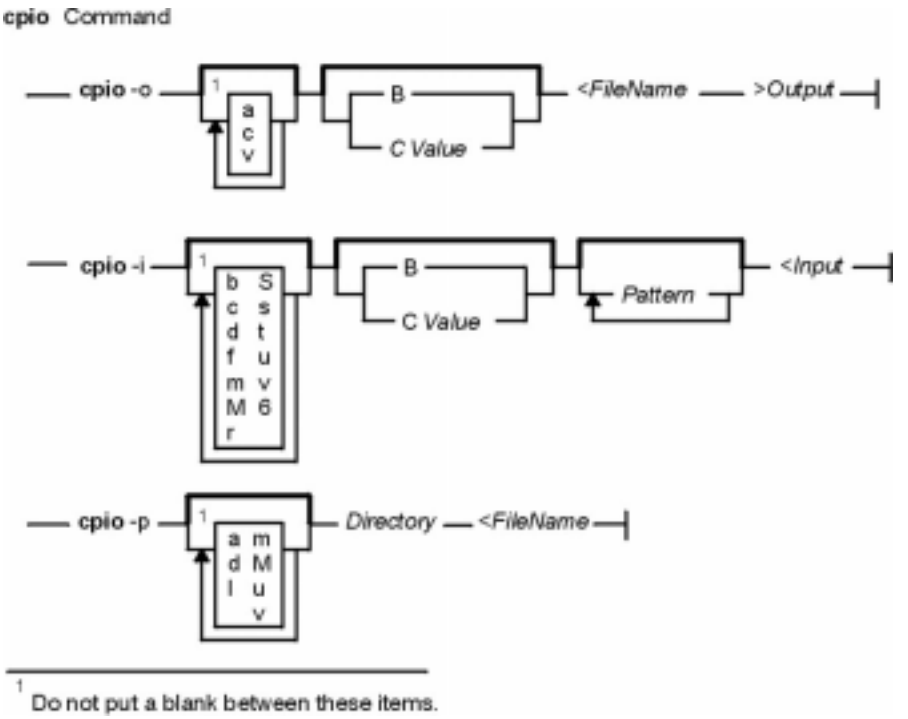
National Language Support Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

cpio Command

Purpose

Copies files into and out of archive storage and directories.

Syntax



cpio −o [**a**] [**c**] [**v**] [**B** | **C***Value*] <*FileName*>*Output*

cpio −i [**b**] [**c**] [**d**] [**f**] [**m**] [**M**] [**r**] [**s**] [**t**] [**u**] [**v**] [**S**] [**6**] [**B** | **C***Value*] [*Pattern...*] <*Input*

cpio −p [**a**] [**d**] [**l**] [**m**] [**M**] [**u**] [**v**] *Directory*<*FileName*

Description

Attention: 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 is reached.

Note:

1. The **cpio** command is not enabled for files greater than 2 Gig in size due to limitations imposed by XPG/4 and POSIX.2 standards.
2. **cpio** does not preserve the sparse nature of any file that is sparsely allocated. Any file that was originally sparse before the restoration will have all space allocated within the filesystem for the size of the file.

cpio −o Command

The **cpio −o** command reads file path names from standard input and copies these files to standard output, along with path names and status information. Avoid giving the **cpio** command path names made up of many uniquely linked files, as it may not have enough memory to keep track of them and would lose linking information.

cpio −i Command

The **cpio −i** command reads from standard input an archive file created by the **cpio −o** command and copies from it the files with names that match the *Pattern* parameter. These files are copied into the current directory tree. You can list more than one *Pattern* parameter, using the file name notation described in the **ksh** command. Note that in this application the special characters * (asterisk), ? (question mark), and [...] (brackets and ellipses) match the / (slash) in path names, in addition to their use as described in the **ksh** command**.** The default for the *Pattern* parameter is an * (asterisk), selecting all files in the Input. In an expression such as [a−z], the minus sign means *through* according to the current collating sequence.

A collating sequence can define equivalence classes for use in character ranges.

cpio −p Command

The **cpio −p** command reads file path names from standard input and copies these files into the directory named by the *Directory* parameter. The specified directory must already exist. If these path names include directory names that do not already exist, you must use the **d** flag to cause the specified directory to be created.

> **Note:** You can copy special files only if you have root user authority.

Parameters

Directory	Specifies the directory.
<FileName	Specifies a list of file names for the **cpio** command to use as input.
>Output	Specifies the output device such as a diskette or file. For more information on using tape devices see the **rmt** special file.
<Input	Specifies the input device (where *Input* is the *Output* file created by the **cpio −o** command). For more information on using tape devices, see the **rmt** special file.
Pattern	Specifies the pattern (as described in the **ksh** command) to be used with the command. The default for the *Pattern* parameter is an * (asterisk), selecting all the files in the *Input*.

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 the access times of the source files to their previous times.

b Swaps both bytes and halfwords.

> **Note:** If there is an odd number of bytes or halfwords in the file being processed, data can be lost.

B Performs block input and output using 512 bytes to a record.

> **Note:** When using the **B** or **C** options to extract or create a tape archive, the blocking factor must be a multiple of the physical block size for that tape device.
>
> When using the **B** or **C** options to extract an archive from tape, the blocking factor should not be larger than the size of the archive as it exists on the tape.

The **B** flag and the **C** flag are mutually exclusive. If you list both, the **cpio** command uses the last one it encounters in the flag list.

c Reads and writes header information in ASCII character form. If a **cpio** archive was created using the **c** flag, it must be extracted with **c** flag.

C *Value* Performs block input and output using the *Value* parameter times 512 bytes to a record. For instance, a **−C2** flag changes the block input and output sizes to 1024 bytes to a record.

d Creates directories as needed.

f Copies all files except those matching the *Pattern* parameter.

l Links files rather than copying them, whenever possible. This flag can only be used with the **cpio −p** command.

m Retains previous file modification time. This flag does not work when copying directories.

M Retains previous file modification time even when directories are copied.

r Renames files interactively. If you do not want to change the file name, enter a single period or press the <Enter> key. In the latter case, the **cpio** command does not copy the file.

s Swaps bytes. This flag is used only with the **cpio −i** command.

Note: If there is 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 is an odd number of halfwords in the file being processed, data can be lost.

t Creates a table of contents. This operation 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 with the **t** flag, the output looks similar to that of the **ls −l** command.

6 Processes an old file (for example, one written in UNIX Sixth Edition format). This flag is usable only with the **cpio −i** command.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To copy files onto diskette, enter:

   ```
   cpio −ov <filenames >/dev/rfd0
   ```

   This copies the files with path names listed in the `filenames` file 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.

   Note: Files with uid's and gid's greater than 65535 cannot be archived using the **cpio** command. In such instances, the user should use backup and restore.

2. To copy files in the current directory onto diskette, enter:

   ```
   ls *.c | cpio −ov >/dev/rfd0
   ```

   This copies all the files in the current directory whose names end with `.c`

3. To copy the current directory and all subdirectories onto diskette, enter:

```
find . −print | cpio −ov >/dev/rfd0
```

This saves the directory tree that starts with the current directory (`.`) and includes all of its subdirectories and files. Do this faster by entering:

```
find . −cpio /dev/rfd0 −print
```

The `−print` entry 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, enter:

   ```
   cpio −itv </dev/rfd0
   ```

   This displays the table of contents of the data previously saved onto the `/dev/rfd0` file in the **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, enter:

   ```
   cpio −idmv </dev/rfd0
   ```

   This copies the files previously saved onto the `/dev/rfd0` file by the **cpio** command back into the file system (specify the **−i** flag). The **d** flag allows the **cpio** command to create the appropriate directories if a directory tree is saved. The **m** flag maintains the last modification time in effect when the files are 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, enter:

   ```
   cpio −i "*.c" "*.o" </dev/rfd0
   ```

   This copies the files that end with `.c` or `.o` from diskette. Note that the patterns `"*.c"` and `"*.o"` must be enclosed in quotation marks to prevent the shell from treating the * (asterisk) 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, enter:

   ```
   cpio −ir </dev/rfd0
   ```

   The **−r** flag causes the **cpio** command to ask you whether to rename each file before copying it from diskette. For example, the message:

   ```
   Rename <prog.c>
   ```

   asks whether to give the file saved as `prog.c` a new name as it is copied. To rename the file, type the new name and press the Enter key. To keep the same name, you must enter the name again. To avoid copying the file at all, press the Enter key.

8. To copy a directory and all of its subdirectories, enter:

   ```
   mkdir /home/jim/newdir
   find . −print | cpio −pdl /home/jim/newdir
   ```

   This duplicates the current directory tree, including the current directory and all of its subdirectories and files. The duplicate is placed in the new `/home/jim/newdir` directory. The **l** flag causes the **cpio** command to link files instead of copying them, when possible.

Note: The performance of **cpio** to the 9348 Magnetic Tape Unit Model 12 can be improved by changing the default block size. To change the block size, enter the following at the command line:

```
chdev -l <device_name> -a block_size=32k
```

Files

/usr/bin/cpio Contains the **cpio** command.

Related Information

The **find** command, **ksh** command, **ln** command, **ls** command, and **li** command.

The **cpio** file format, **rmt** special file.

The Backup Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* provides information on different methods of backing up, restoring process, different types of backup media, and guidelines for backup policies.

The Directory Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* explains working with directories and path names.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains file system types, management, structure, and maintenance.

The Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* provides information on working with files.

cplv Command

Purpose

Copies the contents of a logical volume to a new logical volume.

Syntax

To Copy to a New Logical Volume

cplv [**−v** *VolumeGroup*] [**−y** *NewLogicalVolume* | −Y *Prefix*] *SourceLogicalVolume*

To Copy to an Existing Logical Volume

cplv −e *DestinationLogicalVolume* [**−f**] *SourceLogicalVolume*

Description

> **Attention:** Do not copy from a larger logical volume containing data to a smaller one. Doing so results in a corrupted file system because some data (including the superblock) is not copied.
>
> **Attention:** This command will fail if the **cplv** creates a new logical volume and the volume group is varied on in concurrent mode.

The **cplv** command **copies** the contents of *SourceLogicalVolume* to a new or existing *DestinationLogicalVolume*. The *SourceLogicalVolume* parameter can be a logical volume name or a logical volume ID. The **cplv** command creates a new logical volume with a system−generated name by using the default syntax. The system−generated name is displayed.

> **Notes:**
>
> 1. If you are copying a striped logical volume and the destination logical volume does not exist, an identical copy, including the striped block size and striping width of the source logical volume is created and then the data is copied.
> 2. If you are copying a striped logical volume and you have created the destination logical volume, with the **mklv** command using a different stripe block size and striping width, or the destination is not a striped logical volume, the new characteristics are maintained, and the data is copied from the source logical volume.
> 3. To use this command, you must either have root user authority or be a member of the **system** group.

You can use the Web−based System Manager Volumes application (**wsm lvm** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit cplv** fast path to run this command.

Flags

−e Specifies that the *DestinationLogicalVolume* exists and that a new logical volume should not be created. If the *DestinationLogicalVolume* is smaller than the *SourceLogicalVolume*, the extra logical partitions are not copied. When you use

	this flag, any data already in the *DestinationLogicalVolume* is destroyed. For this reason, user confirmation is required, unless the **−f** flag is added. The *Type* characteristic of the *DestinationLogicalVolume* must be **copy** to prevent inadvertently overwriting data. To change the *Type* characteristic, use the **chlv** command.
−f	Copies to an existing logical volume without requesting user confirmation.
−v *VolumeGroup*	Specifies the volume group where the new logical volume resides. If this is not specified, the new logical volume resides in the same volume group as the *SourceLogicalVolume*.
−y*NewLogicalVolume*	Specifies the name to use, in place of a system−generated name, for the new logical volume. Logical volume names must be unique systemwide names, and can range from 1 to 15 characters.
−Y *Prefix*	Specifies a prefix to use in building a system−generated name for the new logical volume. The prefix must be less than or equal to 13 characters. A name cannot begin with a prefix already defined in the PdDv class in the Device Configuration Database for other devices, or a name already used by another device.

Examples

1. To copy the contents of logical volume `fslv03` to a new logical volume, enter:

   ```
   cplv fslv03
   ```

   The new logical volume is created, placed in the same volume group as `fslv03`, and named by the system.

2. To copy the contents of logical volume `fslv03` to a new logical volume in volume group `vg02`, enter:

   ```
   cplv −v vg02
   fslv03
   ```

   The new logical volume is created, named, and added to volume group `vg02`.

3. To copy the contents of logical volume `lv02` to a smaller, existing logical volume, `lvtes`t, without requiring user confirmation, enter:

   ```
   cplv −e lvtest −f lv02
   ```

Files

/usr/sbin Directory where the **cplv** command resides.

Related Information

The **chlv** command, **migratepv** command, **mklv** command.

The Logical Volume Storage Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the Logical Volume Manager, physical volumes, logical volumes, volume groups, organization, ensuring data integrity, and understanding the allocation characteristics.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains the structure, main menus, and tasks that are done with

SMIT.

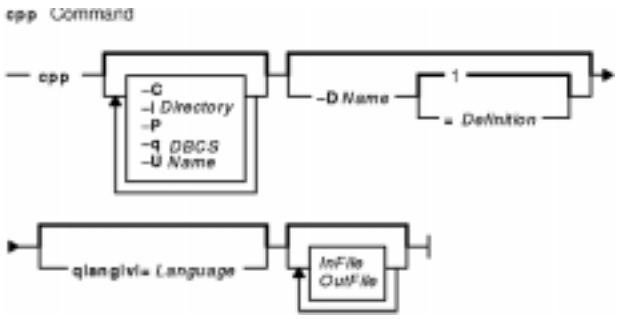
AIX HACMP/6000 Concepts and Facilities.

cpp Command

Purpose

Performs file inclusion and macro substitution on C language source files.

Syntax



/usr/ccs/lib/cpp [**−C**] [**−P**] [**−q***DBCS*] [**−I***Directory*] [**−U***Name*] [**−D***Name* [*=Defin ition*]] [**−qlanglvl=***Language*] [*InFile*] [*OutFile*]

Description

The **cpp** command performs file inclusion and macro substitution on C language source files. It reads *InFile* and writes to *OutFile* (standard input and standard output by default).

The **cpp** command is designed to conform to the preprocessing directives and instructions for the C language as defined by the document "Draft American National Standard for Information Systems − Programming Language C" (X3J11/88−159).

The **cpp** program recognizes the following special names:

__LINE__	The current line number.
__DATE__	The date of translation of the source file.
__TIME__	The time of translation of the source file.
__STDC__	Indicates a conforming implementation.
__FILE__	The current file name.
__STR__	Indicates the compiler will generate inline code for certain string functions (as defined in **/usr/include/string.h**).
__MATH__	Indicates the compiler will generate inline code for certain math functions (as defined in **/usr/include/math.h**).
__ANSI__	Indicates **langlvl** is set equal to ANSI.
__SAA__	Indicates **langlvl** is set equal to SAA.
__SAA_L2__	Indicates **langlvl** is set equal to SAAL2.
__EXTENDED__	Indicates **langlvl** is set equal to extended.
__TIMESTAMP__	Indicates the date and time when the source file was last modified.

All **cpp** directive lines must begin with a **#** (pound sign). These directives are:

#define *NameTokenString*	Replaces subsequent instances of *Name* with *TokenString*.
#define *Name***(***Argument,...,Argument***)** *TokenString*	Replaces subsequent instances of the sequence *Name* (*Argument, . . . ,Argument*) with *TokenString*, where each occurrence of an *Argument* in *TokenString* is replaced by 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***"** or **#include <***File***>**	Includes at this point the contents of *File*, which **cpp** then processes. If you enclose *File* in " " (double quotation marks) the **cpp** command searches first in the directory of *InFile*, second in directories named with the **−I** flag, and last in directories on a standard list. If you use the <*File*> notation, the **cpp** command searches for *File* only in the standard directories. It does not search the directory in which *InFile* resides.
#line *Number* ["*File*"]	Causes the implementation to behave as if the following sequence of source lines begins with a source line that has a line number as specified by *Number*. If *File* is supplied, the presumed name of the file is changed to be *File*.
#error *TokenString*	Produces a diagnostic message that includes *TokenString*.
#pragma *TokenString*	An implementation−defined instruction to the compiler.
#endif	Ends a section of lines begun by a test directive (**#if**, **#ifdef**, or **#ifndef**). Each test directive must have a matching **#endif**.
#ifdef *Name*	Places the subsequent lines in the output only if: *Name* has been defined by a previous **#define** OR *Name* has been defined by the **−D** flag, OR *Name* is a special name recognized by the **cpp** command, AND *Name* has not been undefined by an intervening **#undef**, OR *Name* has not been undefined with the **−U** flag.
#ifndef *Name*	Places the subsequent lines in the output only if:

Name has never been defined by a previous **#define**,

AND

Name is not a special name recognized by the **cpp** command,

OR

Name has been defined by a previous **#define** but it has been undefined by an intervening **#undef**,

OR

Name is a special name recognized by the **cpp** command, but it has been undefined with the **−U** flag.

#if *Expression*

Places subsequent lines in the output only if *Expression* evaluates to nonzero. All the binary nonassignment C operators, the ?: operator, and the unary −, !, and − operators are legal in *Expression*. 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 *Expression* in these two forms:

defined *(Name) or* ***defined****Name*

This allows the utility of **#ifdef** and **#ifndef** in a **#if** directive. Only these operators, integer constants, and names that are known by **cpp** should be used in *Expression*. The **sizeof** operator is not available.

#elif *Expression*

Places subsequent lines in the output only if the expression in the preceding **#if** or **#elif** directive evaluates to false or is undefined, and this *Expression* evaluates to true.

#else

Places subsequent lines in the output only if the expression in the preceding **#if** or **#elif** directive evaluates to false or is undefined (and hence the lines following the **#if** and preceding the **#else** have been ignored).

Each test directive's condition is checked in order. If it evaluates to false (0), the group that it controls is skipped. Directives are processed only through the name that determines the directive in order to keep track of the level of nested conditionals; the rest of the directives' preprocessing tokens are ignored, as are the other preprocessing tokens in the group. Only the first group whose control condition evaluates to true (nonzero) is processed. If none of the conditions evaluates to true, and there is a **#else** directive, the group controlled by the **#else** is processed; lacking a **#else** directive, all the groups until the **#endif** are skipped.

Flags

−C	Copies C language comments from the source file to the output file. If you omit this flag, the **cpp** command removes all C language comments except those found on a **cpp** directive line.
−D*Name*[=*Definition*]	Defines *Name* as in a **#define** directive. The default *Definition* is **1**.
−I*Directory*	Looks first in *Directory*, 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 **#include**.
−P	Preprocesses input without producing line control information for the next pass of the C compiler.
−q*DBCS*	Specifies double−byte character set mode.
−U*Name*	Removes any initial definition of *Name*, where *Name* is a symbol predefined by the preprocessor (except for the four preprocessor mode indicators: **__ANSI__**, **__EXTENDED__**, **__SAA__**, and **__SAA_L2__**). This flag is not recognized in ANSI mode.
−qlanglvl=*Language*	Selects a language level for processing. *Language* can be ANSI, SAA, SAAL2, or extended. The default is extended. **Note:** When *Language* is extended, _NO_PROTO is not automatically defined. Such definition can be done using the **−D** option in the **/etc/xlc.cfg** file.

Examples

1. To display the text that the preprocessor sends to the C compiler, enter:

   ```
   /usr/ccs/lib/cpp pgm.c
   ```

   This preprocesses `pgm.c` and displays the resulting text at the work station. 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, enter:

   ```
   /usr/ccs/lib/cpp −P −C pgm.c pgm.i
   ```

   This preprocesses `pgm.c` and stores the result in `pgm.i`. It omits line numbering information intended for the C compiler (`−P`), and includes program comments (`−C`).

3. To predefine macro identifiers, enter:

   ```
   /usr/ccs/lib/cpp −DBUFFERSIZE=512 −DDEBUG
   pgm.c
   pgm.i
   ```

   This defines `BUFFERSIZE` with the value `512` and `DEBUG` with the value `1` before preprocessing.

4. To use **#include** files located in nonstandard directories, enter:

   ```
   /usr/ccs/lib/cpp −I/home/jim/include
   pgm.c
   ```

   This looks in the current directory for quoted **#include** files, then in `/home/jim/include`, and then in the standard directories. It looks in `/home/jim/include` for angle−bracketed **#include** files (< >) and then in the standard directories.

5. To preprocess with the ANSI definition, enter:

```
/usr/ccs/lib/cpp −qlanglvl=ansi pgm.c
```

Files

/usr/include Standard directory for **#include** files.

Related Information

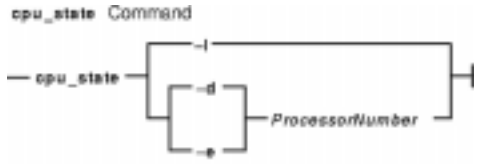
The **m4** command.

cpu_state Command

Purpose

Controls and lists which processors will be active when the system is next started.

Syntax



cpu_state **−l** | { **−d** | **−e** } *ProcessorNumber*

Description

The **cpu_state** command controls and lists which processors on a multiprocessor system will be active when the system is next started. The **−d** or **−e** flags respectively disable or enable the processor identified by the *ProcessorNumber* parameter. The **−l** flag displays a report with the following fields:

Name	The ODM processor name, shown in the form `proc`*x*, where *x* is the physical processor number.
Cpu	The logical processor number.
Status	The processor state for the next boot.
Location	The ODM processor location code. This code is shown in the form *AA−BB−CC−DD*. *AA* is always 00 (to indicate the main unit). *BB* is 0*x*, where *x* is P, Q, R, or S, to indicate the first, second, third or fourth processor card. *CC* is always 00, and *DD* is 00 or 01 to indicate the processor position on the cpu card.

The physical processor numbers used in ODM names are based on the location of processors within the system (cpu card and card position number). Logical numbers are assigned to processors by numbering all *currently* enabled processors starting with physical processor 0 (zero). Thus, in a system with 2 enabled processors out of a possible 4, the logical numbers 0 (zero) and 1 (one) are used.

The **Status** field does not display the current processor state, but rather the state to be used for the next boot. This means that if an enabled processor is disabled using the **−d** flag, the **−l** flag will display the processor as disabled, but having a logical number. When the system is rebooted, the processor will not be used and its status field will remain disabled (until it is changed) and it will have no logical number. If the processor status is unknown, the **Status** field contains `no reply`. This status means either that the processor has a hardware problem detected by a power−on test, or that no reply was received.

Notes:

1. The **cpu_state** command changes do not take effect until the system is restarted.
2. The **cpu_state** command can only be used by a **root** user, and is intended for tasks such as system maintenance, performance measurement, and testing.
3. The **cpu_state** command works only on multiprocessor systems with Micro Channel I/O. For IBM systems, this includes the IBM 7012 Model G Series, the IBM 7013 Model J Series, and the IBM 7015 Model R Series.

The **lsdev** command can be used on any multiprocessor system to query information about processors. Note that the conventions stated above for the logical processor number and location code are not common across all multiprocessor systems. The **lsdev** command for this is as follows:

```
lsdev −C −c processor −S Available
```

Flags

−d Disables the specified processor.

−e Enables the specified processor.

−l Lists the status of all processors.

Examples

1. To list the status of the processors in the system, enter:

   ```
   cpu_state −l
   ```

   On a four processor system with all processors running, this produces a listing similar to the following:

   ```
   Name    Cpu  Status     Location
   proc0   0    Enabled    00−0P−00−00
   proc1   1    Enabled    00−0P−00−01
   proc2   2    Enabled    00−0Q−00−00
   proc3   3    Enabled    00−0Q−00−01
   ```

2. To prevent processor number one from running when the system is restarted, enter:

   ```
   cpu_state −d 1
   ```

   For the system shown in the previous example, the command **cpu_state −l** would then produce a listing similar to the following:

   ```
   Name    Cpu  Status     Location
   proc0   0    Enabled    00−0P−00−00
   proc1   1    Disabled   00−0P−00−01
   proc2   2    Enabled    00−0Q−00−00
   proc3   3    Enabled    00−0Q−00−01
   ```

   If the system is rebooted, the disabled state takes effect and processor one will no longer have a logical processor number. The command **cpu_state −l** would then produce a listing similar to the following:

   ```
   Name    Cpu  Status     Location
   proc0   0    Enabled    00−0P−00−00
   proc1   −    Disabled   00−0P−00−01
   proc2   1    Enabled    00−0Q−00−00
   proc3   2    Enabled    00−0Q−00−01
   ```

3. To re−enable processor number one after it has been marked disabled, enter:

   ```
   cpu_state −e 1
   ```

   Using the second listing in the previous example (booting after disabling processor one). the command **cpu_state −l** would then produce a listing similar to the following:

```
Name    Cpu  Status     Location
proc0   0    Enabled    00-0P-00-00
proc1   -    Enabled    00-0P-00-01
proc2   1    Enabled    00-0Q-00-00
proc3   2    Enabled    00-0Q-00-01
```

Implementation Specifics

This command is part of Base Operating System (BOS) Runtime.

File

/usr/sbin/cpu_state Contains the **cpu_state** command.

Related Information

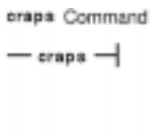
Starting the System in *AIX Version 4.3 System Management Concepts: Operating System and Devices*, Stopping the System in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

craps Command

Purpose

Starts the craps game.

Syntax



craps

Description

The **craps** command starts the craps game similar to ones played in Las Vegas. The **craps** command simulates the roller while you place bets. You can bet with the roller by making a positive bet or you can bet with the house by making a negative bet.

You begin the game with a two thousand dollar bankroll. When the program prompts with `bet?`, you can bet all or part of your bankroll. You can not bet more than your current bankroll. The roller throws the dice. The payoff odds are one−to−one.

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. On subsequent rolls, the point wins for the roller; 7 wins for the house; and any other number rolls again. For example:

```
Your bankroll is $2000
bet? 100
5       3
The point is 8
        6       6
        4       1
        2       1
        2       5
You lose your bet of $100
Your bankroll is $1900
```

In this example, the player has a bankroll of two thousand dollars and bets one hundred dollars. The first roll was 8. This became the point because neither you nor the house wins on a first roll of 8. Subsequent rolls were: 12, 5, 3, and 7. The house wins on a roll of 7 when the roller is trying to match the point. The player lost the bet of one hundred dollars. After displaying the new bankroll, the game will prompt `bet?` and the game will continue.

If you lose your bankroll, the game prompts with `marker?`, offering to lend you an additional two thousand dollars. Accept the loan by responding `Y` (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 two thousand dollars, the game asks `Repay marker?`. If you want to repay part or all of your loan, enter `Y` (yes). If you have more than one marker, the **craps** command prompts `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 twenty thousand dollars), the game tells you so and exits. If you accumulate a bankroll of more than fifty thousand dollars while holding markers, the money owed is repaid automatically.

A bankroll of more than one hundred thousand dollars breaks the bank, and the game prompts `New game?` To quit the game, press the Interrupt (Ctrl−C) or End Of File (Ctrl−D) key sequence; the game indicates whether you won, lost, or broke even, and exits.

Files

/usr/games Location of the system's games.

Related Information

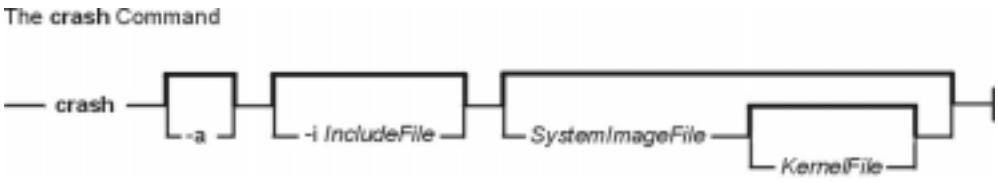
The **arithmetic** command, **back** command, **bj** command, **fish** command, **fortune** command, **hangman** command, **moo** command, **number** command, **quiz** command, **ttt** command, **turnoff** command, **turnon** command, **wump** command.

crash Command

Purpose

Displays system images for examining a dump.

Syntax



crash [**−a**] [**−i***IncludeFile*] [*SystemImageFile* [*KernelFile*]]

Description

The **crash** command is an interactive utility for examining an operating system image, or the running kernel. The **crash** command facility interprets and formats control structures in the system and certain miscellaneous functions for examining a dump.

The *SystemImageFile* parameter specifies the file that contains the system image. The default for the *SystemImageFile* parameter is the **/dev/mem** file. The *KernelFile* parameter contains the kernel symbol definitions. The default for the *KernelFile* parameter is the **/unix** file.

You can run the **crash** command with no arguments to examine an active system. If you specify a system image file, the **crash** command assumes it is a system dump file and sets the default thread to the thread running at the time of the crash.

Notes:

1. When using the **crash** command a kernel file must be available.
2. Stack tracing of the current process on a running system does not work.
3. When **crash** is run on a running system, you will see a message about the possibility that **crash** may cause a system crash and/or data corruption, and there will be error log entries noting each time **crash** is started and stopped. These are done because of the risk posed by certain **crash** subcommands. Those subcommands are shown here with the messages, labeled **Attention**. If you limit yourself to subcommands without these messages, you can avoid problems.
4. If the dump file has been compressed, the **crash** command will be unable to read it. If you are attempting to run **crash** directly against a dump device that contains a compressed dump, you will have to use the **savecore** command to copy the dump to a file first. After running the **savecore** command, uncompress the file and run **crash** against that file. If you have a compressed regular file that contains your dump, it must be uncompressed with the **uncompress** command before **crash** can read the dump. See the **sysdumpdev** command for more information on compressed dumps. This section applies to only AIX Version 4.3.2 and later versions.

The **crash** command recognizes the following aliases in subcommand format specifications:

Format	Aliases	Format	Aliases
byte	**b**	**instruction**	**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**
i−node	**ino, i**		

Note: The **instruction** format disassembles instructions at a specified address. The **crash** command attempts to disassemble instructions to a PowerPC or POWER architecture instruction, depending upon the machine architecture on which the system dump was taken. If the instruction is not in the appropriate architecture, the **crash** command attempts to disassemble it to an instruction from the other architecture. This behavior can be modified with the **set idarch** subcommand. For more information, see "Interpreting an Assembler Listing" in *AIX Assembler Language Reference*.

Symbols displayed by crash

Several **crash** subcommands (such as **trace**, **ds**, ...) display symbol names and offsets corresponding to numeric addresses. All symbols from *KernelFile* display with the symbol name, a + sign, and the offset, for example:

```
.thread_terminate+3e0
vmmerrlog + 0x00000018
```

Text symbols from kernel extensions are shown in the form .[*KernelExtensionName*:*FunctionName*]+*Offset*, but only if the traceback tables for the kernel extension can be found in *SystemImageFile*. *Offset* is the hexadecimal offset (in bytes) from the beginning of the function. In the following example, the address being shown is `0x44c` bytes past the beginning of the `svc_run` routine in the `nfs_ext` kernel extension.

```
.[nfs.ext:svc_run]+44c
```

Data symbols from kernel extensions cannot be displayed by **crash**, so the addresses of data items within kernel extensions display in the form .[*KernelExtensionName*]+*Offset*, where *Offset* is the offset of the address from the beginning of the kernel extension. This is also used for text symbols when traceback tables are not available in *SystemImageFile*. In the following example, the address being shown is `0x4c` bytes past the beginning of the `nfs.ext` kernel extension:

```
.[nfs.ext] + 0x0000004c
```

Addresses in crash

Many of the commands in **crash** take addresses or kernel symbol names as parameters. Addresses are always specified in hexadecimal, and can usually be specified in one of the following forms:

addr An 8 digit hexadecimal number, taken to be as an effective address within the context of the current process and thread, or (in some cases) the context of the thread specified on a previous **cm** command. **addr** can be prefixed with the characters `0x`.

segid:offset segid is the segment ID for a virtual memory segment. The maximum size is 6 hex digits. offset is the offset (in bytes) from the beginning of that segment. The maximum size is 7 hex digits.

r:realaddr **r** is the literal character "**r**". **realaddr** is a real memory address. This form can only be used

when running **crash** against a system dump, and it only will display dump data areas that were dumped by real address instead of virtual address. **readaddr** can be up to 12 hexadecimal digits. To enhance readability, you may include underscores ("_") anywhere within these values.

Examples:

```
18340050
2314:55300
r:14_3370_0560 (same as r:1433700560)
```

Command−line Editing

The **crash** command provides command line editing features similar to those provided by the Korn shell. **vi** mode provides vi−like editing features, while **emacs** mode gives you controls similar to emacs. You can turn these features on by using the **crash** subcommand **set edit**. So, to turn on vi−style command−line editing, you would type the subcommand `set edit vi`.

Output Redirection

The **crash** command provides a subset of Korn shell input/output redirection. Specifically, the following operators are provided:

| (pipe symbol)

Pipes all output of the command before the symbol to the input of the command after the symbol. Both standard output and error output are affected, which is different than standard shell behavior.

> filename

Writes the output of the command before the **>** to filename. Both standard and error output are written to the file.

>> filename

Adds the output of the command before the **>>** to the end of filename. Both standard and error output are written to the file.

Subcommands

The **crash** command recognizes several subcommands. The **crash** command presents a > (greater−than sign) prompt when it is ready to interpret subcommands entered from the workstation. The general subcommand format for the **crash** command is:

Subcommand [*Flags*] [*StructuresToBeDisplayed*]

When allowed, the *Flag* parameters 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 operations with addresses assume hexadecimal notation.

Since the **crash** command only deals with kernel threads, the word *thread* when used alone will be used to mean *kernel thread* in the **crash** documentation that follows. The default thread for several subcommands is the current thread (the thread currently running). On a multiprocessor system, you can use the **cpu** subcommand to change the current processor: the default thread becomes the running thread on the selected processor.

The parameters *ProcessTableEntry* and *ThreadTableEntry* are used in many subcommands to indicate a

process or thread respectively. These parameters are simply numbers for table entry indexes which can be displayed using the **proc** and **thread** subcommands.

Most of the subcommands recognized by the **crash** command have aliases (abbreviated forms that give the same result). The **crash** command recognizes the following subcommands:

alter [[**−c**] [**−s**] [**−w**] [**−l**] [**L** | **ll**] *addrval*

Recognized by the **poke** subcommand alias. Put the value *val* at given address *addr*.

Attention: You should use this command with extreme caution. Errors in the use of this command may cause system failures.

−c
Indicates that *val* is a char.(1 byte) This is the default.

−s
Indicates that *val* is a short. (2 bytes)

−w
Indicates that *val* is a word. (4 bytes)

−l
Indicates that *val* is a long. (4 bytes)

−L | ***ll***
Indicates that *val* is a long long. (8 bytes)

alter [[**−c**] [**−s**] [**−w**] [**−l**]] [**L** | **ll**] *addrorigval=val*

Recognized by the **poke** subcommand alias. Put value at given address only if *origval* is already stored at that address.

Attention: You should use this command with extreme caution. Errors in the use of this command may cause system failures.

Note: The flags indicate the *val* is a **c**har, **s**hort, **w**ord, **l**ong, or **L** | **ll** for storing 64−bit information.

alter [[**−c**] [**−s**] [**−w**] [**−l**]] [**L** | **ll**] *addr*

Recognized by the **poke** subcommand alias. Start altering memory at the given address, interactively,

Attention: You should use this command with extreme caution. Errors in the use of this command may cause system failures.

Note: The flags indicate the *val* is a **c**har, **s**hort, **w**ord, **l**ong, or **L** | **ll** for storing 64−bit information.

buf [*BufferHeaderNumber*] . . .

Recognized by the **bufhdr** and **hdr** subcommand aliases. Displays the system buffer headers. The **buff.**# file is created where the **#** variable file extension is the *BufferHeaderNumber*. Binary data is written to the **buff.**# file.

buffer [*Format*] [*BufferHeaderNumber*] . . .

Recognized by the **b** subcommand alias. Displays the data in a system buffer according to the *Format* parameter. When specifying a buffer header number, the buffer associated with that buffer header is displayed. If you do not provide a *Format* parameter, the previous *Format* is used. Valid parameters are the **decimal**, **octal**, **hex**, **character**, **byte**, **directory**, **i−node**, and **write** formats. The **write** format creates a file in the current directory containing the buffer data.

calc*expr*
Recognized by the **@** subcommand alias. Expression can be any expression supported by the **bc** command, although mixed case is supported. Values are in hex.

callout
Recognized by the **c**, **call**, **calls**, **time**, **timeout**,**tout**, and **trb** subcommand aliases. Displays all entries on the active **trblist** (rough equivalent of the cutoutable).

cm [**ldron** | **ldroff**] [**vmmon** | **vmmoff**] [*ThreadTableEntry SegmentNumber*]
Changes the segment map of the **crash** command internal pointers for any process thread segment not paged out if you specify the process−thread−slot number and segment number. This allows the **od** subcommand to display data relative to the beginning of the segment desired for threads other than the current thread. Specification of **vmmon** or **vmmoff** allows selection of whether effective addresses in the range 0x70000000 through 0xaffffffff are to be interpreted by the **od** subcommand as kernel or VMM data references. Similarly, selection of **ldron** or **ldroff** allows selection of whether effective addresses in segment 11 (0xbxxxxxxx) and segment 13 (0xdxxxxxxx) are to be interpreted by the **od** subcommand as references to loader data. When entering the cm subcommand without any parameters, it resets the map of internal pointers.

conv*num*
Recognized by the **%** subcommand alias. If *num* is hex, convert to decimal, otherwise convert to hex. Bases are guessed by leading **0x** for hex or *base***#** for bases other than decimal.

conv [−] [**bdxo**] *num*
Recognized by the **%** subcommand alias. Convert *num* to specified base **b**inary, **d**ecimal, **h**ex, or **o**ctal. Bases are guessed by leading **0x** for hex or *base***#** for bases other than decimal.

conv [−] [**bdxo**] [**bdxo**] *num*
Recognized by the **%** subcommand alias. Convert *num* from base specified by the first flag to the base specified by the second flag, **b**inary, **d**ecimal, **h**ex, or **o**ctal.

cpu [*ProcessorNumber*]
If no argument is given, the **cpu** subcommand displays the number of the currently selected processor. Initially, the selected processor is the processor that caused the system crash (when running crash against a dump), or processor 0 (when running against a running system). If the *ProcessorNumber* argument is given, the **cpu** subcommand selects the specified processor as the current processor. By extension, this selects the current thread (the running thread on the selected processor). Processor numbering starts from zero.

dblock*Address*
Recognized by the **dblk** subcommand alias. The **dblock** subcommand displays the allocated streams data block headers. The *Address* parameter is required. If the *Address* is not supplied, this subcommand prints an error message stating that the address is required. See **/usr/include/sys/stream.h** header file for the **datab** structure definition. The `freep` and `db_size` descriptions are not included in **/usr/include/sys/stream.h**. These structure members and their descriptions are:

`freep`
Address of the free pointer.

`db_size`
Size of the data block.

There is no checking performed on the address passed in as the required

parameter. The **dblock** subcommand will accept any address. It is up to the user to be sure that a valid address is specified.

To determine a valid address, you will need to run the **mblock** subcommand. From the output of the **mblock** subcommand, select a non−zero data block address from under the column heading, DATABLOCK.

decode*instr*
Decode the given instruction word. *instr* is specified as a hex value.

devsw
Show device switch table.

devsw*major*
Show the device switch structure for device with the given major number.

dlock[*ThreadIdentifier* | **−p** [*ProcessorNumber*]]

Displays deadlock analysis information about all types of locks (simple, complex, and lockl). The **dlock** subcommand searches for deadlocks from a given start point. If *ThreadIdentifier* is given, the corresponding thread is the start point. If **−p** is given without a *ProcessorNumber*, the start point is the running thread on the current processor. If **−p***ProcessorNumber* is given, the running kernel thread on the specified processor is the start point. If no arguments are given, **dlock** searches for deadlocks among all threads on all processors.

The first output line gives information about the starting thread, including the lock which is blocking the kernel thread, and a stack trace showing the function calls which led to the blocking lock request. Each subsequent line shows the lock held by the blocked thread from the previous line, and identifies the thread or interrupt handler which is blocked by those locks. If the information required for a full analysis is not available (paged out), an abbreviated display is shown; in this case, examine the stack trace to locate the locking operations which are causing the deadlock. The display stops when a lock is encountered for a second time, or no blocking lock is found for the current thread.

dmodsw
Displays the streams drivers switch table. The information printed is contained in an internal structure. The members of this internal structure and their descriptions are:

`address`
Address of **dmodsw**.

`d_next`
Pointer to the next driver in the list.

`d_prev`
Pointer to the previous driver in the list.

`d_name`
Name of the driver.

`d_flags`
Flags specified at configuration time.

`d_sqh`
Pointer to synch queue for driver−level synchronization.

`d_str`
Pointer to streamtab associated with the driver.

`d_sq_level`
Synchronization level specified at configuration time.

`d_refcnt`
Number of open or pushed count.

`d_major`
Major number of a driver.

The `flags` structure member, if set, is based on one of the following values:

#define	Value	Description
F_MODSW_OLD_OPEN	0x1	Supports old−style (V.3)open/close parameters
F_MODSW_QSAFETY	0x2	Module requires safe timeout/bufcall callbacks
F_MODSW_MPSAFE	0x4	Non MP Safe drivers need funneling

The synchronization level codes are described in the **/usr/include/sys/strconf.h** header file.

ds [*DataAddress*] . . . Finds the data symbols closest to the given addresses.

du [*ThreadTableEntry*] Displays a combined hex and ASCII dump of the specified thread's **uthread** structure and of the user structure of the process which owns the thread. If the data is not available (paged out), a message is displayed. The default is the current thread.

dump Displays the name of each component for which there is data present. After you select a component name from the displayed list, the **crash** program loads and runs the associated formatting routine contained in the**/usr/lib/ras/dmprtns** directory. If there is more than one data area for the selected component, the formatting routine displays a list of the data areas and allows you to select one. The **crash** command then displays the selected data area. You may enter the **quit** subcommand to return to the previously displayed list and make another selection or enter the **quit** subcommand a second time to leave the **dump** subcommand loop.

errpt [*count*] Displays messages in the error log. *Count* is the number of messages to print that have already been read by the **errdemon** process. (The default is 3 messages.) **errpt** always prints all messages that have not yet been read by the **errdemon** process.

file [*FileTableEntry*] . . . Recognized by the **files** and **f** subcommand aliases. Displays the file table. Unless specific file entries are requested, only those with a nonzero reference count are displayed.

find [**−u**] [**−s**] [**−p** *slot*] [**−c***context*] [**−a***alignment*] *pattern*

Recognized by the **x** subcommand alias. Search user−space for a given *pattern*. The default is to search the GPR save areas in the mstsave areas which are both on the Current Save Area Chain (CSA) and in each uthread area for every thread.

−u
: Search all process private segments, (Stack, Uarea,...)

−s
: Search all process private segment from the current stack pointer.

−c*context*
: Number of bytes of context to print on a match.

−a*alignment*
: Byte alignment for *pattern*. The default is 4.

−p*slot*
: Search only specified process. The default is to search all processes.

Attention: Using this command on a running system may cause system crashes.

Rules for *pattern*

pattern is a search pattern of any arbitrary length that contains either a hexadecimal number or a string. To specify a hexadecimal pattern, just type the hex digits. 'Don't care' digits can be represented with the character `x`. To specify a string pattern, enclose the pattern in double quotes. 'Don't care' characters can be represented with the sequence `\x`

Examples:

```
> find −k 02x4
00110a28: 02140008                                |....|
00110af0: 02640004                                |.d..|
00110c80: 02e40004                                |....|
003f0ed8: 02242ff8                                |.$/.|
...

> find −k "b\xt"
00012534: 6269745f                                |bit_|
00012618: 6269745f                                |bit_|
0001264c: 6269745f                                |bit_|
00021cb0: 62797465                                |byte|
00021d60: 62797465                                |byte|
...

> find −k "i_ena" 0 250000
001ceaa8: 695f656e 61626c65                       |i_enable|
```

find −k [**−c***context*] [**−a***alignment*] *pattern* [*start*[*end*]]

Recognized by the **x** subcommand alias. Search the kernel segments. The default range is the whole of each kernel segment.

−c*context* Number of bytes of context to print on a match.

−a*alignment* Byte alignment for *pattern*. The default is 4.

find −b*branch_addr* [*start_addr*[*end_addr*]]

Recognized by the **x** subcommand alias. Search for a branch to the given address. The default range is the whole of each kernel segment.

find −m [**−a***addr*] [**−t***type*] [**−c**] [**−i**] [*start*[*end*]]

Recognized by the **x** subcommand alias. Search the things that look like mbufs. The default search range is the network memory heap.

−a Search for mbufs that point to this cluster address.

−t*type* Only search for this *type* of mbuf.

−c Only search for clusters.

−i Ignore length sanity checks.

find −v [**−f**] *wordval* [*start*[*end*]]

Recognized by the **x** subcommand alias. Search for the first word not matching the given value. The default is to search the kernel segments.

−f Force scan to continue when a region not in the dump is scanned.

find −U*seg_id* Recognized by the **x** subcommand alias. Search for processes whose segment registers contain the given segment ID.

fmodsw Displays the streams modules switch table. The information printed is contained in an internal structure. The members of this internal structure and their definitions are:

`address`	Address of **fmodsw**.
`d_next`	Pointer to the next module in the list.
`d_prev`	Pointer to the previous module in the list.
`d_name`	Name of the module.
`d_flags`	Flags specified at configuration time.
`d_sqh`	Pointer to synch queue for module−level synchronization.
`d_str`	Pointer to streamtab associated with the module.
`d_sq_level`	Synchronization level specified at configuration time.
`d_refcnt`	Number of open or pushed count.
`d_major`	−1.

The `flags` structure member, if set, is based on one of the following values:

#define	Value	Description
F_MODSW_OLD_OPEN	0x1	Supports old−style (V.3)open/close parameters
F_MODSW_QSAFETY	0x2	Module requires safe timeout/bufcall callbacks
F_MODSW_MPSAFE	0x4	Non MP Safe drivers need funneling

The synchronization level codes are described in the **/usr/include/sys/strconf.h** header file.

fs [*ThreadTableEntry*]

Traces a kernel stack for the thread specified by the thread slot number. Displays the called subroutines with a hex dump of the stack frame for the subroutine that contains the parameters passed to the subroutine. The default thread is the thread currently running. This subcommand will not work on the current thread or a running system because it uses stack tracing; however, it does work on a dump image.

help Recognized by the **?** subcommand alias. Print a list of a commands with short descriptions.

help*cmd...* Recognized by the **?** subcommand alias. Print a long description of the specified command.

help */regex...* Recognized by the **?** subcommand alias. Print those long descriptions that match the specified extended, case−insensitive regular expression.

hide*symbol...* Hide the specified symbol from the **crash** commands that convert addresses to symbols and offsets. The main reason for this ability is to hide symbols that may show up in the middle of a function. This occurs in assembly routines. See the **unhide** subcommand.

hide Show all hidden symbols. See the **unhide** subcommand.

id[***...**] *addr* [*units*]

id[***...**] *symname* [*units*] Instruction Decode. Equivalent to **od** with the **instruction** format. The *units* are measured in number of instructions.

inode [−] [<MAJ> <MIN> <INUMB>] . . .

Recognized by the **ino** and **i** subcommand aliases. Displays the i−node table and the i−node data block addresses. A specific i−node can be

displayed by specifying the major and minor device number of the device where the i−node resides and the i−node number. The i−node will only be displayed if it is currently on the system hash list.

kfp[*FramePointer*] . . .
Recognized by the **fp** and **rl** subcommand aliases. If the **kfp** subcommand is entered without parameters, it displays to the screen the last kernel frame pointer address that was set using **kfp**. If a frame pointer address is provided, then it sets the kernel frame pointer to the new address. This subcommand is used conjunction with the **−r** flag on the **trace** subcommand.

knlist[*Symbol*] . . .
Displays the addresses of all the symbol names given. If the symbol is not found, a no−match message is displayed to the screen. This subcommand reads the kernel export list for either the running system or a dump image.

le [**−l32** | **−l64** | **−p***proc_slot* | **−a**] [[*address* | *name*]...]

Displays load list entries. The default is to display load list entries starting at the kernel load anchor. If an *address* is specified, without the **−a** flag, only load list entries which include the *address* within the text or data area display. If a *name* is specified all load list entries which have a *name* that includes the input string display. If an attempt is made to display a paged−out loader entry the subcommand displays an error message. The following flags control the load list entry chain that is searched/displayed:

−l32
Uses the 32−bit shared library load list anchor.

−l64
Uses the 64−bit shared library load list anchor.

−p*proc_slot*
Uses the load list anchor contained in the indicated processes user area.

−a
Displays a single load list entry at a specified address (an address must be specified with this flag)

link*addrnum* [*next_offset* [*end_val*]

Recognized by the **ll** subcommand alias. Follows linked list starting at *addr*. Print *num* words for each entry. *next_offset* is the offset in words of the next pointer, the default is 0. *end_val* is the value of the next pointer that terminates the list, the default is 0.

linkblk
Recognized by the **lblk** subcommand alias. The **linkblk** subcommand displays the streams linkblk table. See **/usr/include/sys/stream.h** header file for the **linkblk** structure definition. If there are no **linkblk** structures found on the system, the **linkblk** subcommand prints a message stating that no structures are found.

lock
Recognized by the **locks** subcommand alias. Print status on global kernel locks as well as threads waiting on events and locks.

lock [**−clsq**] *addr*|*symbol*...

Recognized by the **locks** subcommand alias. Print lock at *addr*, or address specified by *symbol*. The default format is that of a Simple_lock.

−c Print as a Complex_lock

−l Print as a lock_t

−s Print as a Simple_lock. This is the default.

−q Suppress instrumentation data.

Status Bits:

I	INTERLOCK
W	WAITING
WW	WANT_WRITE (Complex_lock)
RD	READ_MODE(Complex_lock)
L	LOCKBIT (Simple_lock)
S	INSTR_ON (Simple_lock)

mblock*Address*

Recognized by the **mblk** subcommand alias. The **mblock** subcommand displays the allocated streams message block headers. The *Address* parameter is required. If *Address* is not supplied, the subcommand prints an error message stating that the address is required. See **/usr/include/sys/stream.h** header file for the **msgb** structure definitions.

There is very little checking performed on the address passed in as the required parameter. The **mblock** subcommand accepts any address that falls on a 128−byte boundary. It is up to the user to be sure that a valid address is specified.

To determine a valid address, run the **queue** subcommand. From the output of the **queue** subcommand, you will need to select a non−zero address in the head of the message queue, under the column heading HEAD, for either a read queue or a write queue.

mbuf *Address*

Displays system **mbuf** structures at the specified address.

mst [**−f**] [*Address*] . . .

Displays the mstsave portion of the uthread structure at the addresses specified (see the **uthread.h** and **mstsave.h** header files in **/usr/include/sys**). If you do not specify an address, it displays all of the mstsave entries on the CSA chain except the first. If you specify the **−f** flag the first mstsave area on the CSA chain displays.

ndb

Displays network kernel data structures either for a running system or a system dump. The **ndb** subcommand, short for network debugger, supports the following options:

?	Provides first−level help information.
help	Provides additional help information.
tcb [*Addr*]	Shows TCBs. The default is **HEAD TCB.**
udb [*Addr*]	Shows UDBs. The default is **HEAD UDB.**
sockets*Addr*	Shows sockets at the given address.
mbuf [*Addr*]	Shows the **mbuf** at the specified address.
ifnet [*Addr*]	Shows the **ifnet** structures at the specified address.
quit	Stops the running option.
xit	Exits the **ndb** submenu.

netm

Displays the most recent net_malloc_police record.

Note: Requires that the Memory Overlay Detection System (MODS) be enabled or that the network option net_malloc_police be turned on. For additional information on the net_malloc_police option, see the **no** command.

netm −a

Displays all records, starting with the most recent.

Note: Requires that the MODS be enabled or that the network option net_malloc_police be turned on. For additional information on the net_malloc_police option, see the **no** command.

netm*addr* Displays records whose address or caller fields match the given address.

Note: Requires that the MODS be enabled or that the network option net_malloc_police be turned on. For additional information on the net_malloc_police option, see the **no** command.

netstat Equivalent to the command line version of the **netstat** command.

nm[*Symbol*] . . . Displays symbol value and type as found in *KernelFile*.

od [**ldr:**][**vmm:**] [* . . .] [*SymbolName* | *Address*] [*Count*] [*Format*]

The **od** subcommand dumps the number of data values specified by *Count* starting at *SymbolName* or *Address* according to *Format*. Possible formats are octal, longoct, decimal, longdec, character, hex, instruction, and byte. The default is hex.

Note: If you use the *Format*, you must also use *Count*. If the *SymbolName* or *Address*> is proceeded by an asterisk, then the symbol or address is dereferenced before displaying the data. Additionally, the strings **ldr:** and **vmm:** may be used to indicate that addresses are to be considered loader or VMM addresses, just as if the **cm ldron** and/or **cm vmmon** subcommand had been issued.

ppd [*ProcessorNumber* *]

Displays per−processor data area (PPDA) structures for the specified processor. If no processor is specified, the current processor selected by the **cpu** subcommand is used. If the asterisk argument is given, the PPDA of every enabled processor is displayed.

prall Equivalent to **crash −a** from the command line.

print [*Type*] *Address* Recognized by the **pr**, **str**, or **struct** subcommand aliases. Does **dbx**−style printing of structures. The **−i** option must be given on the command line to use this feature. *Type* is the name of the structure to be displayed.

print −d*type* Recognized by the **pr**, **str**, or **struct** subcommand aliases. Sets the default type for subsequent print commands to *type*.

print −l*offset*|*name* [**−e***end_val*] [*type*] *address*

Recognized by the **pr**, **str**, or **struct** subcommand aliases. Displays a linked list, starting at *address* using *offset* or structure member *name* as the location of the next pointer. Stop when next value of equals *end_val*. The default *end_val* is 0.

proc [−] [**−r**] [*ProcessTableEntry*] . . .

Recognized by the **ps** and **p** subcommand aliases. Displays the process table, including the thread count (the number of threads in the process) and state of each process. (See the **/usr/include/sys/proc.h** file for this structure definition.) The **−r** flag displays only runnable processes. The − (minus) flag displays a longer listing of the process table.

qrun Displays the list of scheduled streams queues. If there are no queues found for scheduling, the **qrun** subcommand prints a message stating that there are no queues scheduled for service.

queue [*Address*] Recognized by the **que** subcommand alias. The **queue** subcommand displays the STREAMS queue. If the optional parameter, *Address*, is not supplied, **crash** will display information for all write queues available. Refer to the **/usr/include/sys/stream.h** header file for the queue structure definition.

If you wish to see the information stored for a read queue, issue the **queue** subcommand with the read queue address specified as the *Address* parameter.

When you issue the **queue** subcommand with the *Address* parameter, the column headings do not distinguish between the read queue and the write queue. One queue address will be displayed under the column heading, QUEUE and the other queue in the pair will be displayed under the column heading, OTHERQ. The write queue will have a numerically higher address than the read queue.

quit Recognized by the **q** subcommand alias. Exits from the **crash** command.

search[**−sn**] *name* Search the symbols table for *name*.

−s Prints symbols matching *name* in the **nm** format. Also prints the symbol table entry for the last symbol found.

−n Prevents the search from examining kernel extensions..

search[**−n**] *addr* Search for the symbol with the largest value less than or equal to *addr*.

−n Prevents the search from examining kernel extensions..

segst64 [**−p***pslot* | **−t***tslot*] [**−l***limit* [**−s***segflag*[:*value*][, *segflag*[:*value*]]...] [**−n** [*start_esid* [*end_esid*]]

Recognized by the **adspace**, **as**, and **sr** subcommand aliases. Displays segstate information for a 64−bit process. The segstate for the current process displays unless the **−p** or **−t** flags are specified. All of the segstate entries display unless limited by the **−l** flag or the starting esid, *start_esid* and possible ending esid, *end_esid*. Specifying the **−s** flag limits the display to only those segstate entries matching the given *segflag*s, matching pattern types, as well as their corresponding values. The **−l** flag limits the display to a maximum number of entries. The **−n** flag also prints the segnodes for the displayed data. Segnode entries are not included in the count when limiting the data with **−l**.

−p pslot	Specifies the process slot number.
−t tslot	Specifies the thread slot number.
−s*segflag*:*[value]*	Limites the display to the segstate entries matching that *segflag* and *value*.
−l limit	Specifies the number of entries to print.
−n	Prints the uadnodes for the displayed data.

select Recognized by the **sel** subcommand alias. Displays all select control blocks.

select p*proc_slot* Recognized by the **sel** subcommand alias. Displays select control blocks for process in specified slot.

Note: The **p** flag is not prefixed with a **−**.

select*dev_idunique_id*

Recognized by the **sel** subcommand alias. Displays select control blocks matching the specified device and unique IDs

set Display **crash** variables and values.

set allhex [**no**] Causes **crash** to use only hex values for both input and output as opposed to a mixture of hex and decimal. Specify **no** to turn this option off.

set edit [**emacs**|**gmacs**|**none**|**vi**]

Sets command line editing mode.

set fpregs [**yes**|**no**|**auto**]

Specify whether or not floating−point registers should be displayed. If

auto is used, the *fpeu* variable in the mstsave area determines when to display the registers.

set idarch [**ppc**|**pwr**|**auto**]

Set instruction decode architecture. **auto** detects the architecture from the system.

set logfile [*filename*]

Set logfile to given name, or turn off logging if no name is given.

set loglevel [**0**|**1**|**2**] Set logging granularity to:

0
coarse−only commands will be logged.

1
medium−commands and output to terminal will be logged.

2
fine−commands and all outputs will be logged, including redirected.

set prtype [*type*] Set the default print type. This is equivalent to **print −d***type*.

set quiet [**no**] Suppress error messages concerning missing or swapped out threads and processes. Specify **no** to turn off this option.

socket[−] . . . Recognized by the **sock** subcommand alias. Displays the system socket structures. If the − (minus sign) flag is used, the socket buffers will also be displayed.

sr64 [**−p***pslot* | **−t***tslot*] [**−l***limit* [**−n** [*start_esid* [*end_esid*]]

Recognized by the **segst** and **seg** subcommand aliases. Displays the effective segment IDs (esid) and their corresponding segvals for a 64−bit process. If you do not specify the **−p** or **−t** flags, **sr64** uses the current process. Otherwise, it uses *pslot* as the process slot number for the desired process, or *tslot* as the thread slot number of a thread contained within the desired process. It lists all entries in the adspace unless a starting esid, *start_esid* and possible ending esid, *end_esid* is given. Also, it stops listing if the number of entries specified by the **−l** flag have printed. Since adspace_t holds 16 entries, each line consists of an esid, its corresponding value, and the 3 subsequent values following it in the adspace_t. The **−n** flag also prints the uadnodes for the displayed data. uadnode entries are not included in the count when limiting the data with **−l**.

−p*pslot* Specifies the process slot number.

−t*tslot* Specifies the thread slot number.

−l*limit* Specifies the number of entries to print.

−n Prints the uadnodes for the displayed data.

stack[*ThreadTableEntry*] . . .

Recognized by the **stk**, **s**, **kernel**, and **k** subcommand aliases. Displays a dump of the kernel stack of the thread identified by *ThreadTableEntry*. If you do not specify an entry, information about the last running kernel thread is displayed. You cannot trace the stack of the current kernel thread on a running system.

stat Displays statistics found in the dump. These statistics include the panic message (if a panic occurred), time of crash, system name, and whether the MODS (here called **xmalloc** debug) is enabled.

status [*ProcessorNumber*] Displays a description of the kernel thread scheduled on the designated processor. If no processor is specified, the **status** subcommand displays information for all processors. The information displayed includes the processor number, thread identifier, thread table slot, process identifier,

process table slot, and process name.

stream

Displays the stream head table. The information printed is contained in an internal structure. The members of this internal structure are:

`address`	Address of stream head
`wq`	Address of streams write queue
`dev`	Associated device number of the stream
`read_error`	Read error on the stream
`write_error`	Write error on the stream
`flags`	Stream head flag values
`push_cnt`	Number of modules pushed on the stream
`wroff`	Write offset to prepend M_DATA
`ioc_id`	ID of outstanding M_IOCTL request
`pollq`	List of active polls
`sigsq`	List of active M_SETSIGs

The `flags` structure member, if set, is based on combinations of the following values:

#define	Value	Description
F_STH_READ_ERROR	0x0001	M_ERROR with read error received, fail all read calls
F_STH_WRITE_ERROR	0x0002	M_ERROR with write error received, fail all writes
F_STH_HANGUP	0x0004	M_HANGUP received, no more data
F_STH_NDELON	0x0008	do TTY semantics for ONDELAY handling
F_STH_ISATTY	0x0010	this stream acts as a terminal
#define	**Value**	**Description**
F_STH_MREADON	0x0020	generate M_READ messages
F_STH_TOSTOP	0x0040	disallow background writes (for job control)
F_STH_PIPE	0x0080	stream is one end of a pipe or fifo
F_STH_WPIPE	0x0100	stream is the "write" side of a pipe
F_STH_FIFO	0x0200	stream is a fifo
F_STH_LINKED	0x040	stream has one or more lower streams linked
F_STH_CTTY	0x0800	stream controlling tty
F_STH_CLOSED	0x4000	stream has been closed, and should be freed

F_STH_CLOSING	0x8000	actively on the way down

symptom[**−e**] Displays the symptom string for a dump. It is not valid on a running system. The optional **−e** option will create an error log entry containing the symptom string, and is normally only used by the system and not entered manually. The symptom string can be used to identify duplicate problems.

tcb [*ThreadTableEntry*] . . . Displays the mstsave portion of the user structures of the named threads (see the **user.h** and **mstsave.h** header files). If you do not specify an entry, information about the last running thread displays. This subcommand replaces the **pcb** subcommand.

thread [**−**] [**−r**] [**−p***ProcessTableEntry* | **−a***Address* | *ThreadTableEntry*]

Recognized by the **th** subcommand alias. Displays the contents of the thread table. The **−** (minus) flag displays a longer listing of the thread table. The **−r** flag displays only runnable threads. The **−p** flag displays only those threads which belong to the process identified by *ProcessTableEntry*. The **−a** flag displays the thread structure at *Address*. If *ThreadTableEntry* is given, only the corresponding thread is displayed.

trace [**−r** | **−m** [**−f**]] [**−k** | **−s**] [**−r**] [*ThreadTableEntry*] . . .

Recognized by the **t** subcommand alias. Displays a kernel stack trace of the thread identified by *ThreadTableEntry*. If you do not specify a thread table entry, information about the current thread is displayed. When using the **−k** flag, the stack frame addresses indicate the stack frame containing the link register value pointing to the function that is displayed. The **−m** flag causes **trace** to display the traceback associated with each mstsave area on the Current Save Area Chain (CSA), except the first. To see a traceback from the first mstsave area, specify the **−f** flag. When either the **−m** or **−k** flags are used, trace may also show the LR (link register) and top stack frame pointer. These are not part of the stack trace and are therefore marked with an asterisk. The **−r** flag will cause **trace** to use the kernel frame pointer set up by the **kfp** subcommand as its starting address instead of the frame pointer found in the *SystemImageFile*. The **trace** subcommand will stop and an error will be reported if an invalid frame pointer is encountered. The **−s** flag displays saved register information for each stack frame. With no flags, **trace** prints address information.

ts[*TextAddress*] . . . Finds the text symbols closest to the given addresses.

tty[**d**] [**l**] [**e**] [*Name* | *Major* [*Minor*]]

Displays the tty structures. If no parameters are specified, a short list of all open terminals is displayed. Selected terminals can be displayed by specifying the terminal name, such as **tty1**, or a major number with optional minor number. The flags modify the displayed information: the **d** flag displays driver information; the **l** flag displays line discipline information; and the **e** flag displays information for every module or driver present in the stream for the selected lines.

unhide*symbol...* Unhide the specified symbol. See the **hide** subcommand.

unhide Unhide all hidden symbols. See the **hide** subcommand.

user [**−s**] [*ThreadTableEntry*] . . .

Recognized by the **uarea**, **u_area**, and **u** subcommand aliases. Displays the uthread structure and the associated user structure of the thread identified by *ThreadTableEntry*. (See the**/usr/include/sys/user.h** file for the user structure definition.) If you do not specify the entry, the information about the last running thread displays. The **−s** flag limits the

	output to segment register information.
var	Recognized by the **tunables**, **tunable**, **tune**, and **v** subcommand aliases. Displays the tunable system parameters.
vfs [−] [*VfsSlotNumber*]	Recognized by the **mount**, **mnt**, and **m** subcommand aliases. The vfs uses the specified *VfsSlotNumbe*r to display an entry in the vfs table. Use the − flag to display the v−nodes associated with the vfs. The default displays the entire vfs table. See the **sys/vfs.h** header file for structure definitions. See examples 8 and 9 for samples of the **vfs** subcommand output.
vnode [*VNodeAddress*]	Displays data at the specified v−node address as a v−node. The address must be specified in hexadecimal notation. The default is to display all v−node structures. See the **sys/vfs.h** header file for structure definitions. See example 10 for a sample of the **vnode** subcommand output.
which*symbol* | *addr*	Recognized by the **wf** subcommand alias. Displays the name of the kernel source file containing *symbol* or *addr*.
xmalloc	Recognized by the **xm** and **malloc** subcommand aliases. Prints information concerning the allocation and usage of kernel memory (the **pinned_heap** and the **kernel_heap**).
xmalloc [*addr*]	Recognized by the **xm** and **malloc** subcommand aliases. Prints **xmalloc** information about *addr*. If *addr* is not specified **crash** attempts to find the addresses involved in the system crash caused by the MODS.
xmalloc −s [*addr*]	Recognized by the **xm** and **malloc** subcommand aliases. Prints debug **xmalloc** allocation records associated with *addr*. **Note:** The **−s** flag requires that the memory overlay detection system (MODS) has been turned on.
xmalloc −h [*addr*]	Recognized by the **xm** and **malloc** subcommand aliases. Prints MODS **xmalloc** free list records associated with *addr*. **Note:** The **−h** flag requires that the memory overlay detection system (MODS) has been turned on.
xmalloc [**−l**] **−f**	Recognized by the **xm** and **malloc** subcommand aliases. Prints allocation records on free list from earliest−freed to latest−freed. The **−l** flag prints a long listing. **Note:** The **−f** flag requires that the memory overlay detection system (MODS) has been turned on.
xmalloc [**−l**] **−a**	Recognized by the **xm** and **malloc** subcommand aliases. Prints the allocation record table. The **−l** flag prints a long listing. **Note:** The **−a** flag requires that the memory overlay detection system (MODS) has been turned on.
xmalloc [**−l**] **−p***pageno*	Recognized by the **xm** and **malloc** subcommand aliases. Prints page descriptor information for page *pageno*. The **−l** flag prints additional information.
xmalloc −d [*addr*]	Recognized by the **xm** and **malloc** subcommand aliases. Prints debug **xmalloc** allocation record hash chain associated with the record hash value for *addr*.
xmalloc−v	Recognized by the **xm** and **malloc** subcommand aliases. Verify allocation trailers of allocated records, and free fill patterns of freed records.

	Note: The **−v** flag requires that the memory overlay detection system (MODS) has been turned on.
!	Runs shell commands.
?	Displays summary of **crash** commands.

Flags

−a Prints an assortment of data structures without user interaction to standard output.

−i Reads this include file and to allow the structures declared in this file to be used with the **print** subcommand. You can supply several options on the command line with this flag.

Examples

1. To invoke the **crash** command with the default system image and kernel image files, enter:

   ```
   crash
   ```

   The **crash** command returns a **>** prompt and waits for you to enter a subcommand.

2. To invoke the **crash** command with the system image file and the UNIX kernel file, enter:

   ```
   crash sysimage /unix
   ```

   The **crash** program returns a **>** prompt and waits for you to enter a subcommand.

   The following examples show examples of the usage of **crash** subcommands:

3. To write buffer 0 to the **buff.0** file in binary format, enter:

   ```
   buffer write 0
   ```

   The **crash** command returns a **>** prompt and waits for you to enter a subcommand.

4. To set the segment map to thread slot number 1 and segment 2, then display ten words starting from address 2ff3b400, using segment 2 of thread slot number 1, enter:

   ```
   >cm 1 2
   t1,2>od 2ff3b400 10
   ```

5. To display the user structure from the unix kernel **sys/user.h**, enter:

   ```
   crash −i sys/user.h <print user *u
   ```

6. To display deadlocks anywhere on the system, enter:

   ```
   >dlock
   ```

7. To display a deadlock involving a given thread 00d3f, enter:

   ```
   >dlock 00d3f
   ```

   The output is similar to:

   ```
   Deadlock from tid 00d3f.  This tid waits for the first line lock,
   owned by Owner−Id that waits for the next line lock, and so on...
        LOCK NAME    |   ADDRESS  |  OWNER−ID | WAITING FUNCTION
              lockC1 | 0x001f79e0 |  Tid 113d | .lock_write_ppc
                      called from :  .times + 0000020c
   ```

```
Dump data incomplete.Only 0 bytes found out of 4.
                  called from :  .file + 0000000b
            lockC2 | 0x001f79e8 | Tid d3f | .lock_write_ppc
                     called from :  .times + 000001c8
Dump data incomplete.Only 0 bytes found out of 4.
                  called from :  .file + 0000000b
```

8. To display the third entry in the vfs table, enter:

   ```
   > vfs 3
   ```

   The output is similar to:

   ```
   VFS ADDRESS TYPE OBJECT STUB NUM FLAGS PATHS
     3 1a62494  jfs 1a6d47c 1a6d650 5 D /dev/hd1 mounted over /u
           flags:  C=disconnected D=device I=remote P=removable
                   R=readonly     S=shutdown U=unmounted Y=dummy
   ```

9. To display the v−node table data associated with the third entry in the vfs table, enter:

   ```
   > vfs − 3
   ```

   The output is similar to:

   ```
   VFS ADDRESS TYPE  OBJECT    STUB NUM FLAGS PATHS
     3 1a62494  jfs 1a6d47c 1a6d650   5 D   /dev/hd1 mounted over /u
   ADDRESS VFS MVFS VNTYPE FSTYPE COUNT ISLOT    INODE FLAGS
   1a6e0ac   3    −   vreg    jfs     1    −   18f82c0
   1a6e218   3    −   vreg    jfs     1    −   18f8770
   1a6e24c   3    −   vreg    jfs     1    −   18f8590
   1a6e17c   3    −   vdir    jfs     3    −   18f7f00
   1a6dea4   3    −   vreg    jfs     2    −   18f65b0
   1a6dfa8   3    −   vdir    jfs     5    −   18f6100
   1a6d47c   3    −   vdir    jfs     1    −   18ea580 vfs_root
   ```

10. To display the v−node data for the address `1a6e078`, enter

    ```
    > vnode 1a6e078
    ADDRESS VFS MVFS VNTYPE FSTYPE COUNT ISLOT    DATAPTR FLAGS
    1a6e078   0    −   vreg    jfs     4     −    18f6790
     Total VNODES printed 1
    ```

Files

/usr/sbin/crash	Contains the **crash** command.
/dev/mem	Default system image file
/unix	Default kernel file
/usr/include/sys/*.h	Header files for table and structure information.

Related Information

The **ksh** command, **mount** command, **pstat** command, **ps** command, **savecore** command, **stty** command, and **sysdumpdev** command.

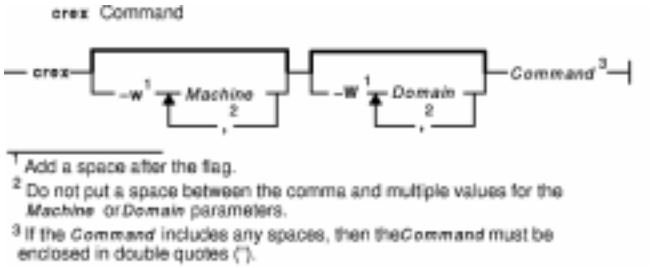
Memory Overlay Detection System (MODS) in the *AIX Version 4.3 Kernel Extensions and Device Support Programming Concepts*.

crex Command

Purpose

Starts the concurrent daemon for the Distributed System Management Interface Tool (DSMIT).

Syntax



crex [**−w***Machine* [**,***Machine*] ...] [**−W***Domain* [**,***Domain*] ...] *Command*

> **Note:** Add a space after the **−w** and **−W** flags. Do not put a space between the comma and multiple values for the *Machine* or *Domain* parameters. If the *Command* includes any spaces, then the *Command* must be enclosed in double quotes (").

Description

The **crex** command allows you to bypass the DSMIT interface by entering commands at the command line. The command is sent to all the machines specified in concurrent mode. To send commands to machines specified in sequential mode, see the **srex** command.

> **Note:** If the command contains any reserved shell characters such as a single quote ('), ampersand (&), or pipe symbol (|), use double quotes to enclose the command. If the command contains double quotes ("), use single quotes to enclose the command.

To use the **crex** command, you must have your system configured correctly as a DSMIT server. To learn more about DSMIT and configuring your system, see "Distributed System Management Interface Tool (DSMIT) Overview," in *Distributed SMIT 2.2 for AIX: Guide and Reference*.

> **Note:** Do not use the **crex** command with interactive commands such as **passwd** or full−screen commands such as **diag**. To run interactive commands, use the **srex** command.

Propagating Commands Through Multiple Managing Machines

The **crex** command can be used for a fast fan−out of commands. For example, suppose that A, B, and C are DSMIT managing machines and that B and C are connected by a slow communication line. A domain of managed machines (Domain_B) is on a LAN together with machine B, and a domain of managed machines (Domain_C) is on a LAN together with machine C. To use managing machine A to execute a non−interactive command on the managed machines in Domain_B and Domain_C, follow these steps:

1. Root must be a registered DSMIT administrator on machines B and C.
2. Enter the following commands on A to establish root's DSMIT administrator credentials on B and C:

```
srex −w B,C dsmit−login
```

3. Send a command from A to B and C. In this example, we remove the user `davis` from all of the managed machines in `Domain_B` and `Domain_C`:

```
crex −w B "crex −W Domain_B rmuser davis"
crex −w C "crex −W Domain_C rmuser davis"
```

> **Note:** Alternatively, you could run `crex −w B,C "crex −W Domain_B rmuser davis","crex −W Domain_C rmuser davis"`

Flags

−w*Machine* Specifies the machines to be in the working collective.

−W*Domain* Specifies the domains to be in the working collective.

Security

Access Control: You must be a registered DSMIT administrator to run this command.

Examples

1. To gather all the network information for a domain and save it in a file, enter:

```
crex −W Domain "netstat −rn" > network_info
```

   The network data is saved in **network_info** within the current directory.

2. To see if a guest account exists on a specific machine or on a machine within a domain, enter:

```
crex −w Machine −W Domain 'lsuser −a id ALL | grep −e "host" −e "guest"'
```

3. To remove user `adam` from machine A and user `davis` from machine B, enter:

```
crex −w Machine_A,Machine_B "rmuser adam","rmuser davis"
```

DSMIT matches the first command with the first machine, the second command with the second machine, and so on.

Files

/usr/share/DSMIT/domains	Contains the list of domains used by DSMIT.
/usr/share/DSMIT/dsmitos	Contains the list of operating systems of DSMIT clients.
/usr/share/DSMIT/hosts	Contains the list of machines with DSMIT installed that can run commands built by the DSMIT server.
/usr/share/DSMIT/security/v5srvtab	Stores the local machine's unique DSMIT principal key.
/usr/share/DSMIT/security/admin.cfg	Stores the DSMIT administrator's keys
/usr/share/DSMIT/security/managing.cfg	Stores intermediate keys used by the managing systems.
/usr/share/DSMIT/security/managed.cfg	Stores the managed machine's DSMIT principal keys.
/usr/share/DSMIT/security/dsmit.ptr	Stores the name of the DSMIT configuration file server.

Related Information

The **srex** command, **dsmit** command.

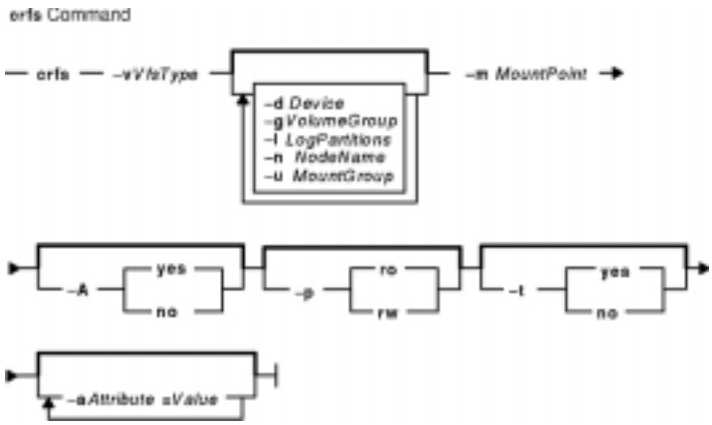
Distributed System Management Interface Tool (DSMIT) Overview in the *Distributed SMIT 2.2 for AIX: Guide and Reference*.

crfs Command

Purpose

Adds a file system.

Syntax



crfs **−v** *VfsType* { **−g***VolumeGroup* | **−d***Device* } [**−l** *LogPartitions*] **−m***MountPoint* [**−n** *NodeName*] [**−u***MountGroup*] [**−A** { **yes** | **no** }] [**−p** {**ro** | **rw** }] [**−a** *Attribute*=*Value* ...] [**−t** { **yes** | **no** }]

Description

The **crfs** command creates a file system on a logical volume within a previously created volume group. A new logical volume is created for the file system unless the name of an existing logical volume is specified using the **−d**. An entry for the file system is put into the **/etc/filesystems** file.

> **Note:** The file system is created with the **setgid** (set group ID) bit enabled. This determines the default group permissions. All directories created under the new file system will have the same default group permissions.

You can use the Web−based System Manager File Systems application (**wsm fs** fast path) to run this command. You could also use the System Management Interface Tool (SMIT) **smit crfs** fast path to run this command.

Flags

−a *Attribute*=*Value* Specifies a virtual file system−dependent attribute/value pair. To specify more than one attribute/value pair, provide multiple **−a** *Attribute*=*Value* parameters (see example).

The following attribute/value pairs are specific to the Journaled File System (JFS):

−a ag= ***{ 8 | 16 | 32 | 64 }***
Specifies the allocation group size in megabytes. An allocation group is a grouping of inodes and disk blocks similar to BSD cylinder groups. The default ag value is 8. This attribute only applies to AIX Version 4.2 or later.

−a bf= ***{ true | false }***
Specifies a large file enabled file system. See "Understanding Large File Enabled File Systems" for more information. If you do not need a large file

enabled file system, set this option to false; this is the default. Specifying **bf=true** requires a fragment size of 4096 and **compress=no**. This attribute only applies to AIX Version 4.2 or later.

−a compress= ***{ no | LZ }***
Specifies data compression. If you do not want data to be compressed, set this option to **no**. The default compress value is **no**. Selecting compression requires a fragment size of 2048 or less.

−a frag= ***{ 512 | 1024 | 2048 | 4096 }***
Specifies the JFS fragment size in bytes. A file system fragment is the smallest unit of disk storage that can be allocated to a file. The default fragment size is 4096 bytes.

−a logname=*LVName*
Specifies the log logical volume name. The specified logical volume will be the logging device for the new JFS. The *LVName* logical volume must already exist. The default action is to use an existing logging device in the target volume group.

−a nbpi= ***{ 512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768 | 65536 | 131072 }***
Specifies the number of bytes per i−node (nbpi). The nbpi affects the total number of i−nodes on the file system. The **nbpi** value is inversely proportional to the number of i−nodes on the file system. The default **nbpi** value is 4096 bytes. The values 32768, 65536, and 131072 only apply to AIX Version 4.2 or later.

−a size=*Value*
Specifies the size of the JFS in 512−byte blocks. If the specified size is not evenly divisible by the physical partition size, it is rounded up to the closest number that is evenly divisible. This attribute is required when creating a JFS file system. See "Understanding JFS Size Limitations" for more information.

The maximum size of a JFS file system is a function of its fragment size and the nbpi value. These values yield the following size restrictions:

nbpi	**Fragment size in bytes**	**Maximum size in 512−byte blocks**
512	512, 1024, 2048, 4096	16777216
1024	512, 1024, 2048, 4096	33554432
2048	512, 1024, 2048, 4096	67108864
4096	512, 1024, 2048, 4096	134217728
8192	512, 1024, 2048, 4096	268435456
16384	512	268435456
16384	1024, 2048, 4096	536870912
32768	512	268435456
32768	1024	536870912
32768	2048, 4096	1073741824
65536, 131072	512	268435456
65536, 131072	1024	536870912
65536, 131072	2048	1073741824
65536, 131072	4096	2147483648

AIX Version 4.1 is limited to NBPI values from 512 to 16384.
In AIX Version 4.3, you can have NBPI values from 512 to 128K, with corresponding maximum file system sizes.

The volume group in which the file system resides defines a maximum logical volume size and also limits the file system size.

Notes:

1. Only JFS file systems created with the default **ag**, **bf**, **compress**, **frag**, and **nbpi** values and a size of less than 2 gigabytes are recognized on an AIX Version 3.2 system. Furthermore, file systems created with an **ag** value greater than 8 are not recognized on an AIX Version 4.1 system much less an AIX Version 3.2 system.
2. The **ag**, **bf**, **compress**, **frag**, and **nbpi** attributes are set at file system creation and cannot be changed after the file system is successfully created. The **size** attribute defines the minimum file system size, and you cannot decrease it once the file system is created.
3. The root filesystem (/) cannot be compressed.
4. Some **nbpi** values and allocation group sizes are mutually exclusive. See "Understanding JFS Size Limitations" for information.

−A Specifies whether the file system is mounted at each system restart:

yes
File system is automatically mounted at system restart.

no
File system is not mounted at system restart.

−d *Device* Specifies the device name of a device or logical volume on which to make the file system. This is used to create a file system on an already existing logical volume.

−g *VolumeGroup* Specifies an existing volume group on which to make the file system. A volume group is a collection of one or more physical volumes.

−l*LogPartitions* Specifies the size of the log logical volume, expressed as a number of logical partitions. This flag applies only to JFS file systems that do not already have a log device.

−m *MountPoint* Specifies the mount point, which is the directory where the file system will be made available.

Note: If you specify a relative path name, it is converted to an absolute path name before being inserted into the **/etc/filesystems** file.

−n*NodeName* Specifies the remote host name where the file system resides. This flag is only valid with remote virtual file systems such as the Network File System (NFS).

−p Sets the permissions for the file system.

ro
Read−only permissions

rw
Read−write permissions

−t Specifies whether the file system is to be processed by the accounting subsystem:

yes
Accounting is enabled on the file system.

no
Accounting is not enabled on the file system (default value).

−u *MountGroup* Specifies the mount group.

−v *VfsType* Specifies the virtual file system type.

Examples

To make a JFS on the `rootvg` volume group with nondefault fragment size and nondefault nbpi, enter:

```
crfs −v jfs −g
rootvg −m /test −a \
size=32768 −a frag=512 −a nbpi=1024
```

This command creates the `/test` file system on the `rootvg` volume group with a fragment size of 512 bytes, a number of bytes per i−node (nbpi) ratio of 1024, and an initial size of 16MB (512 * 32768).

Files

/etc/filesystems Lists the known file systems and defines their characteristics.

Related Information

The **chfs** command, **mkfs** command, **mklv** command.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains file system types, management, structure, and maintenance.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

The System Management Interface Tool (SMIT): Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains SMIT structure, main menus, and tasks.

Understanding Journaled File System Size Limitations in *AIX Version 4.3 System Management Concepts: Operating System and Devices*.

cron Daemon

Purpose

Runs commands automatically.

Syntax

cron

Description

The **cron** daemon runs shell commands at specified dates and times. The following event types are scheduled by the **cron** daemon:

- **crontab** command events
- **at** command events
- **batch** command events
- **sync** subroutine events
- **ksh** command events
- **csh** command events

The way these events are handled is specified by the **/var/adm/cron/queuedefs** file.

Regularly scheduled commands can be specified according to instructions contained in the **crontab** files. You can submit your **crontab** file with the **crontab** command. Use the **at** command to submit commands that are to be run only once. Because the **cron** daemon never exits, it should be run only once.

The **cron** daemon examines **crontab** files and **at** command files only when the **cron** daemon is initialized. When you make changes to the **crontab** files using the **crontab** command, a message indicating the change is sent to the **cron** daemon. This eliminates the overhead of checking for new or changed files at regularly scheduled intervals.

When the **TZ** environment variable is changed, either with the **chtz** command, a Web−based System Manager application, or through SMIT, the **cron** daemon must be restarted. This enables the **cron** daemon to use the correct timezone and summer time change information for the new **TZ** environment variable.

The **cron** daemon creates a log of its activities in the **/var/adm/cron/log** file.

Security

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the **cron** daemon will generate the following audit record (event) every time the command is executed:

Event	Information
CRON_Start	Lists the name of each job, whether the job was initiated by an **at** or **cron** command, and the

time the job started.

CRON_Finish Lists the user's name, process ID of the job, and the time the processing was completed.

See "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more details about how to properly select and group audit events, and how to configure audit event data collection.

Files

/var/adm/cron/FIFO	A named pipe that sends messages to the **cron** daemon when new jobs are submitted with the **crontab** or **at** commands.
/var/adm/cron	Specifies the main **cron** daemon directory.
/var/adm/cron/log	Specifies the accounting information.
/var/adm/cron/queuedefs	Specifies the **cron** daemon events file.
/var/spool/cron	Specifies the spool area.
/usr	Indicates directory kept open by the **cron** daemon.
/usr/bin	Indicates directory kept open by the **cron** daemon.
/usr/lib	Indicates directory kept open by the **cron** daemon.
/etc	Indicates directory kept open by the **cron** daemon.
/tmp	Indicates directory kept open by the **cron** daemon.

Related Information

The **at** command, **auditpr** command, **batch** command, **crontab** command, **csh** command, **ksh** command, **rc** command.

The **sync** subroutine.

The Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains more about audits and audit events.

Setting up and running Web−based System Manager in *AIX Version 4.3 System Management Guide: Operating System and Devices*.

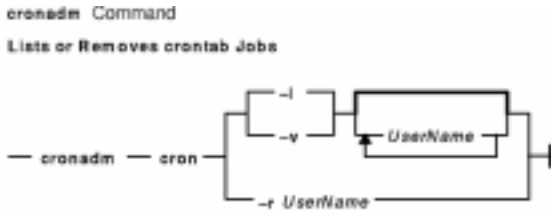
cronadm Command

Purpose

Lists or removes **crontab** or **at** jobs.

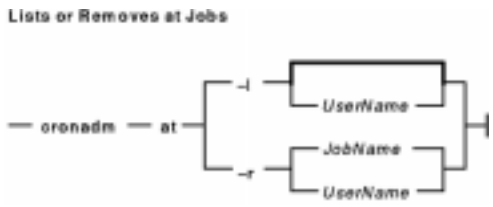
Syntax

To List or Remove crontab Jobs



cronadm cron { { **−l** | **−v** } [*UserName*] ... | **−r** *UserName* }

To List or Remove at Jobs



cronadm at {**−l** [*UserName*] | **−r** { *UserName* | *JobName* } }

Description

The **cronadm** command is used by a root user to list or remove all users **crontab** or **at** jobs.

The **cron** jobs are listed and removed by the *UserName* parameter. One or more *UserNames* can be specified. To list all **cron** jobs, do not specify a user. The **at** jobs are listed by *UserName* and can be removed either by the *UserName* parameter or by the *JobName* parameter.

The name of a **crontab** job file is the name of the user who submitted the **crontab** job and the name of the file in the **/var/spool/cron/crontabs** directory. The name of an **at** job is the name of the user who submitted the **at** job concatenated with a code for the time the **at** job was submitted.

Flags

cronadm cron

−l Lists all **crontab** files. If the *UserName* parameter is specified, only the designated **crontab** files are listed.

−r Removes **crontab** files. The *UserName* parameter should be specified, to remove the designated **crontab** file.

−v Lists the status of all **crontab** jobs. If the *UserName* parameter is specified, only the designated **crontab** files are listed verbosely.

cronadm at

−l Lists the **at** jobs for the user specified by the *UserName* parameter.
−r Removes the **at** job specified by either the *UserName* or *JobName* parameter.

Security

Access Control: Used only by a user with root authority.

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the cronadm command will generate the following audit record (event) every time the command is executed:

Event	Information
AT_JobRemove	Lists whether a **crontab** or **at** job was removed and when.

See "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more details about how to properly select and group audit events, and how to configure audit event data collection.

Examples

1. To list all **crontab** jobs, enter:

   ```
   cronadm cron −l
   ```

2. To list all **at** jobs currently queued for user `bob`, enter:

   ```
   cronadm at −l bob
   ```

Files

/usr/bin/cronadm Contains the **cronadm** command.

Related Information

The **at** command, **auditpr** command, **crontab** command.

The **cron** daemon.

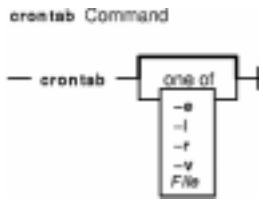
The Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains more about audits and audit events.

crontab Command

Purpose

Submits, edits, lists, or removes cron jobs.

Syntax



crontab [**−e** | **−l** | **−r** | **−v** | *File*]

Description

The **crontab** command submits, edits, lists, or removes cron jobs. A cron job is a command run by the **cron** daemon at regularly scheduled intervals. To submit a cron job, specify the **crontab** command with the **−e** flag. The **crontab** command invokes an editing session that allows you to create a crontab file. You create entries for each cron job in this file. Each entry must be in a form acceptable to the **cron** daemon. For information on creating entries, see "The crontab File Entry Format" .

When you finish creating entries and exit the file, the **crontab** command copies it into the **/var/spool/cron/crontabs** directory and places it in a file named for your current user name. If a file with your name already exists in the **crontabs** directory, the **crontab** command overwrites it.

Alternatively, you can create a crontab file by specifying the *File* parameter. If the file exists, it must be in the format the **cron** daemon expects. If the file doesn't exist, the **crontab** command invokes the editor. If the **EDITOR** environment variable exists, the command invokes the editor it specifies. Otherwise, the **crontab** command uses the **vi** editor.

To list the contents of your crontab file, specify the **crontab** command with the **−l** command. To remove an existing file, use the **−r** flag.

The cron Daemon

The **cron** daemon runs commands according to the crontab file entries. Unless you redirect the output of a cron job to standard output or error, the **cron** daemon mails you any command output or errors. If you specify a cron job incorrectly in your crontab file, the **cron** daemon does not run the job.

The **cron** daemon examines crontab files only when the **cron** daemon is initialized. When you make changes to your crontab file using the **crontab** command, a message indicating the change is sent to the **cron** daemon. This eliminates the overhead of checking for new or changed files at regularly scheduled intervals.

Controls on Using the crontab Command

The **/var/adm/cron/cron.allow** and **/var/adm/cron/cron.deny** files control which users can use the **crontab** command. A root user can create, edit, or delete these files. Entries in these files are user login names with one name to a line. If your login ID is associated with more than one login name, the

crontab command uses the first login name that is in the **/etc/passwd** file, regardless of which login name you might actually be using.

The following is an example of an **cron.allow** file:

```
root
nick
dee
sarah
```

If the **cron.allow** file exists, only users whose login names appear in it can use the **crontab** command. The root user's log name must appear in the **cron.allow** file if the file exists. A system administrator can explicitly stop a user from using the **crontab** command by listing the user's login name in the **cron.deny** file. If only the **cron.deny** file exists, any user whose name does not appear in the file can use the **crontab** command.

A user cannot use the **crontab** command if one of the following is true:

- The **cron.allow** file and the **cron.deny** file do not exist (allows root user only).
- The **cron.allow** file exists but the user's login name is not listed in it.
- The **cron.deny** file exists and the user's login name is listed in it.

If neither the **cron.allow** nor the **cron.deny** file exists, only someone with root user authority can submit a job with the **crontab** command.

The crontab File Entry Format

A crontab file contains entries for each cron job. Entries are separated by newline characters. Each crontab file entry contains six fields separated by spaces or tabs in the following form:

```
minute  hour  day_of_month  month  weekday  command
```

These fields accept the following values:

minute	0 through 59
hour	0 through 23
day_of_month	1 through 31
month	1 through 12
weekday	0 through 6 for Sunday through Saturday
command	a shell command

You must specify a value for each field. Except for the *command* field, these fields can contain the following:

- A number in the specified range. To run a command in May, specify 5 in the **month** field.
- Two numbers separated by a dash to indicate an inclusive range. To run a **cron** job on Tuesday through Friday, place 2−5 in the **weekday** field.
- A list of numbers separated by commas. To run a command on the first and last day of January, you would specify 1,31 in the **day_of_month** field.
- An ***** (asterisk), meaning all allowed values. To run a job every hour, specify an asterisk in the hour field.
  Note: Any character preceeded by a backslash (including the %) causes that character to be treated literally. The specification of days may be made by 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
  ```

would run command on the first and fifteenth days of each month, as well as every Monday. To specify days by only one field, the other field should contain an ***** .

Specifying Commands

The **cron** daemon 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** daemon treats everything that precedes it as the command invocation and makes all that follows it available to standard input, unless you escape the percent sign (\%). Blank lines and lines whose first non−blank character is the number sign (#) will be ignored.

> **Note:** The shell runs only the first line of the command field. All other lines are made available to the command as standard input.

The **cron** daemon starts a subshell from your **HOME** directory. If you schedule a command to run when you are not logged in and you want commands in your **.profile** file to run, the command must explictly read your **.profile** file.

The **cron** daemon supplies a default environment for every shell, defining **HOME,LOGNAME,SHELL** (=**/usr/bin/sh**), and **PATH** (=**/usr/bin**).

Flags

−e Edits a copy of your crontab file or starts an editing session if you don't already have a crontab file. When editing is complete, the entry is installed as your crontab file. The editing session is started using the editor specified by the **EDITOR** environment variable. The default editor is **vi**.

−l Lists your crontab file.

−r Removes your crontab file from the **crontab** directory.

−v Lists the status of your cron jobs.

Security

Auditing Events: If the auditing subsystem has been properly configured and is enabled, the **crontab** command generates the following audit record (event) every time the command is run:

Event	Information
CRON_JobRemove	Lists which users removed a **cron** job and when.
CRON_JobAdd	Lists which users added a **cron** job and when.

See "Setting up Auditing" in *AIX Version 4.3 System Management Guide: Operating System and Devices* for more details about how to properly select and group audit events, and how to configure audit event data collection.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To copy a file called `mycronjobs` into the **/var/admn/cron/crontabs** directory, enter the following:

```
crontab mycronjobs
```

2. To write the time to the console every hour on the hour, enter:

```
0 * * * * echo The hour is `date` .
>/dev/console
```

3. To run the **calendar** command at 6:30 a.m. every Monday, Wednesday, and Friday, enter:

```
30 6 * * 1,3,5 /usr/bin/calendar
```

4. To run the **calendar** command every day of the year at 6:30, enter the following:

```
30 6 * * * /usr/bin/calendar
```

5. To run a script called `maintenance` every day at midnight in August, enter the following:

```
0 0 * 8 * /u/harry/bin/maintenance
```

6. To define text for the standard input to a command, enter:

```
0 16 * 12 5 /usr/sbin/wall%HAPPY HOLIDAY!%Remember to
turn in your time card.
```

   The text following the **%** (percent sign) defines the standard input to the **wall** command as:

```
HAPPY HOLIDAY!

Remember to turn in your time card.
```

Files

/var/adm/cron/FIFO	A named pipe that sends messages to the **cron** daemon when new jobs are submitted with the **crontab** or **at** command.
/var/spool/cron/crontabs	Specifies the crontab spool area.
/var/adm/cron/cron.allow	Specifies a list of users allowed access to the **crontab** command.
/var/adm/cron/cron.deny	Specifies a list of users denied access to the **crontab** command.

Related Information

The **auditpr** command, **sh** command, the **wall** command.

The **cron** daemon.

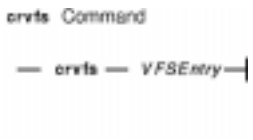
The Auditing Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains more about audits and audit events.

crvfs Command

Purpose

Creates entries in the **/etc/vfs** file.

Syntax



crvfs*VFSEntry*

Description

The **crvfs** command adds **/etc/vfs** file entries by specifying fields within the *VFSEntry* parameter. The *VFSEntry* parameter is composed of the following entries: *VFSName:VFSNumber:MountHelper:FileSystemHelper*.

Any of the entries in the *VFSEntry* parameter can be the **NULL** value, with the exception of the *VFSName* entry. If all the arguments are satisfactory, and the *VFSName* entry given on the command line does not already exist, a new entry is created in the **/etc/vfs** file.

Parameters

VFSEntry Specifies a string in the following format: *VFSName***:***VFSNumber***:***MountHelper***:***FileSystemHelper*

VFSName
: Specifies the name of a virtual file system type.

VFSNumber
: Specifies the virtual file system type's internal number as known by the kernel.

MountHelper
: Specifies the name of the backend used to mount a file system of this type.

FileSystemHelper
: Specifies the name of the backend used by certain file system specific commands to perform operations on a file system of this type.

Examples

To create a new vfs entry called `newvfs`, enter:

```
crvfs "newvfs:4:none:/etc/helpers/newvfshelper"
```

This creates the `newvfs` entry.

Files

/etc/vfs Contains descriptions of virtual file system types.

Related Information

The **chvfs** command, **lsvfs** command, **mount** command, **rmvfs** command.

The File Systems Overview for System Management in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains file system types, management, structure, and maintenance.

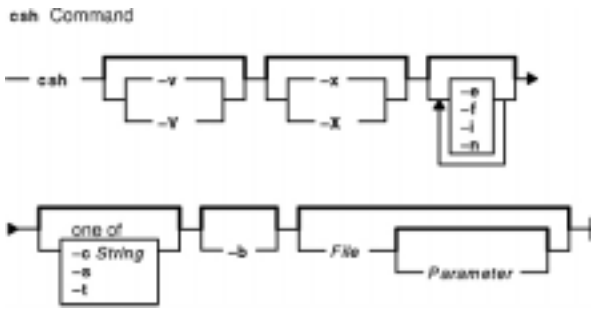
The Mounting Overview in *AIX Version 4.3 System Management Concepts: Operating System and Devices* explains mounting files and directories, mount points, and automatic mounts.

csh Command

Purpose

Invokes the C shell.

Syntax



csh [**−v** | **−V**] [**−x** | **−X**] [**−e**] [**−f**] [**−i**] [**−n**] [**−c** *String* | **−s** | **−t**] [**−b**]
[*File* [*Parameter*]]

Description

The C shell is an interactive command interpreter and a command programming language that uses syntax similar to the C programming language. The shell carries out commands either interactively from a terminal keyboard or from a file. The **csh** command invokes the C shell.

When you invoke the **csh** command, it begins by looking in your home directory and executing commands from the **.cshrc** file (used to store customized user information) if it exists. If the **csh** command runs as a login shell, it executes commands from your **.cshrc** and **.login** files.

After the shell processes flag arguments, if neither the **−i**, **−c**, **−s**, nor **−t** flag is specified and the *File* [*Parameter*] is specified, then the shell executes the script file identified by the *File* [*Parameter*], including any parameters specified. The script file specified must have read permission; the shell ignores any **setuid** and **setgid** settings.

> **Note:** You should not specify a script file if you use the **csh** command with either the **−c** or **−s** flag.

If you specify a script file, the command opens the file and saves the script file name for possible resubstitution by `$0` (dollar sign, zero). The script will then be carried out by **csh**. Remaining parameters initialize the **argv** variable.

> **Notes:**
>
> 1. If C shell is already running, the **.cshrc** file can be read again by typing `source Pathname`, where the *Pathname* parameter is the path to the **.cshrc** file.
> 2. To avoid problems with remote operations, the **.cshrc** file should not contain any functions that echo output unless they test for the **$prompt** variable, which signifies that the shell is interactive. Otherwise, whenever a remote system uses the **exec** command on a command sent by the local system, both the command and the

shell are carried out. For example, `exec csh rcp −t Filename` executes the **.cshrc** file and treats the echoed output as the expected response. An **if** clause can be used to check for the **$prompt** variable.

Flags

If the first argument to a shell is a − (minus sign), that shell is a login shell. The C shell flags are interpreted as follows:

−b Forces a break from option processing, causing any further shell arguments to be treated as non−option arguments. This flag can be used to pass options to a shell script without confusion or possible subterfuge. The shell cannot run a script whose real and effective user and group IDs differ without this flag.

−c Reads commands from the following single argument, which must be present. Any remaining arguments are placed in the **argv** variable.

−e Exits if any invoked command ends abnormally or yields a nonzero exit status.

−f Starts the C shell without searching for or running commands from the **.cshrc** file in your home directory.

−i Prompts for its top−level input (an interactive shell), even if input does not appear to be coming from a workstation. Shells are interactive without this flag if their input and output are attached to workstations.

−n Parses commands but does not run them. This flag aids you in syntactic checking of shell procedures.

−s Takes command input from standard input.

−t Reads and processes a single line of input. You can use a \ (backslash) to escape the new−line character at the end of the current line and continue onto another line.

−V Sets the **verbose** shell variable before the **.cshrc** file runs.

−v Sets the **verbose** shell variable, so that command input is echoed after history substitution.

−X Sets the **echo** shell variable even before the **.cshrc** file runs.

−x Sets the **echo** shell variable, so that commands are echoed after all substitutions and immediately before they run.

Files

$HOME/.cshrc	Read at the beginning of execution by each shell. The **.cshrc** file is user−defined.
$HOME/.login	Read by the login shell after the **.cshrc** file at login.
$HOME/.logout	Read by the login shell at logoff.
/usr/bin/sh	Contains the path to the default shell.
/tmp/sh*	Contains the temporary file for <<.
/etc/passwd	Contains the source of home directories for the ~*File* parameter.

Related Information

The **bsh** command, **chuser** command, **ksh** command, **sh** command.

The **environment** file.

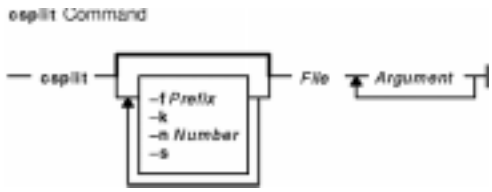
C Shell in *AIX Version 4.3 System User's Guide: Operating System and Devices*.

csplit Command

Purpose

Splits files by context.

Syntax



csplit [**−f** *Prefix*] [**−k**] [**−n** *Number*] [**−s**] *File Argument ...*

Description

The **csplit** command copies the specified file and separates the copy into segments. The original input file, which remains unaltered, must be a text file.

The **csplit** command writes the segments to files **xx00** . . . **xx99**, depending on how many times the *Argument* parameter is specified (99 is the maximum). By default, the *Argument* parameter expects a line number. The following rules apply when you specify multiple line numbers:

- File **xx00** contains the lines from the beginning of the original file up to, but not including, the line number specified in the first *Argument* parameter.
- File **xx01** contains lines beginning with the number specified by the first *Argument* parameter up to, but not including, the line referenced by the second *Argument* parameter. Each line number specified as an argument marks the beginning of a new file.
- File **xx***nn* (the last file created) contains lines beginning with the number specified by the last *Argument* parameter through the end of the file.

For example, if the original file had 108 lines and you entered:

```
csplit orginal.txt 11 72 98
```

the **csplit** command would create four files: the **xx00** file would contain lines 1−10, the **xx01** file would contain lines 11−71, the **xx02** file would contain lines 72−97, the **xx03** file would contain lines 98−108.

The *Argument* parameter can also contain the following symbols and pattern strings:

/Pattern/	Creates a file that contains the segment from the current line up to, but not including, the line containing the specified pattern. The line containing the pattern becomes the current line.
%*Pattern***%**	Makes the line containing the specified pattern the current line, but does not create a file for the segment.
+Number	Moves forward the specified number of lines from the line matched by the preceding pattern. For example, `/Page/+5` searches for `Page`, then advances 5 lines.
−Number	Moves backward the specified number of lines from the line matched by the preceding pattern. For example, `/Page/−5` searches for `Page`, then backs up 5 lines.

{*Number***}** Repeats the preceding option the specified number of times. This number can follow any pattern or line number. If it follows a pattern, the **csplit** command reuses that pattern the specified number of times. If it follows a line number, the **csplit** command splits the file from that point for the number of lines specified by the line number.

Put quotation marks around all patterns that contain spaces or other characters special to the shell. Patterns may not contain embedded new−line characters. In an expression such as `[a−z]`, the − (minus sign) means *through*, according to the current collating sequence. A collating sequence may define *equivalence classes* for use in character ranges.

Flags

−f*Prefix* Specifies the prefix to be used for the created file segments. The default value for this variable is **xx**.

−k Leaves created file segments intact in the event of an error.

−n*Number* Changes the number of decimal places used in the created file names. The default is two decimal places, or **xx00** . . . **xx99**. If you specify the **−n 4** flag, for example, new files are named **xx0000** . . . **xx0099**.

−s Suppresses the display of character counts.

Exit Status

This command returns the following exit values:

0 Successful completion.

>0 An error occurred.

Examples

1. To split the text of `book` into a separate file for each chapter, enter:

   ```
   csplit book "/^ Chapter *[k.0−9]k./" {9}
   ```

   This creates 10 files, **xx00** through **xx09**. The **xx00** file contains the front matter that comes before the first chapter. Files **xx01** through **xx09** contain individual chapters. Each chapter begins with a line that contains only the word `Chapter` and the chapter number.

2. To specify the prefix `chap` for the files created from `book`, enter:

   ```
   csplit −f chap book "/^ Chapter *[k.0−9]k./" {9}
   ```

This splits `book` into files named **chap00** through **chap09**.

Files

/usr/bin/csplit Contains the **csplit** command.

Related Information

The **ed** command, **regcmp** command, **split** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* introduces you to files and the way you can work with them.

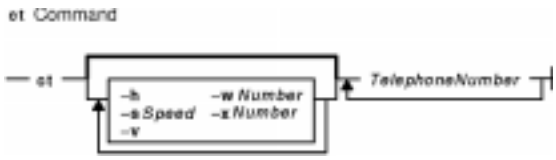
Shells Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes what shells are, the different types of shells, and how shells affect the way commands are interpreted.

ct Command

Purpose

Dials an attached terminal and issues a login process.

Syntax



ct [**−h**] [**−s***Speed*] [**−v**] [**−w***Number*] [**−x***Number*] *TelephoneNumber ...*

Description

The **ct** command is a Basic Networking Utilities (BNU) command that enables a user on a remote terminal, such as an 3161, to communicate with a workstation 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 workstation.

A user on the local system issues the **ct** command with the appropriate telephone number to call the modem attached to the remote terminal. When the connection is established, the **ct** command issues a login prompt that is displayed on the remote terminal. The user on the remote terminal enters a login name at the prompt and opens a new shell. The user at the remote terminal then proceeds to work on the workstation just like a local user.

The **ct** command is useful in the following situations:

- A user working off−site needs to communicate with a local system under strictly supervised conditions, and the local user does not want to disclose the workstation's phone number. Because the local system contacts the remote terminal, the remote user does not need to know the telephone number of the local system. Additionally, the local user issuing the **ct** command can monitor the work of the remote user.
- The cost of the connection should be charged either to the local site or to a specific account on the calling workstation. 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 **ct** command with the telephone number of the remote terminal, but without 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 the modem attached to the remote terminal.

If there are no free lines, the **ct** command displays a message to that effect and asks if the local user wants to wait for one. If the reply is no, the **ct** command hangs up. If the local user wants to wait for a free line, the **ct** command prompts for the number of minutes to wait. The **ct** command continues to dial the remote system at one−minute intervals until the connection is established or until the specified amount of time has elapsed.

In order to establish a **ct** connection, the remote user contacts the local user with a regular telephone call and asks the local user to issue the **ct** command. However, if such connections occur regularly at your site, your system administrator may prefer to set up BNU in such a way that a specified local system automatically issues the **ct** command to one or more specified terminals at certain designated times.

Notes:

1. Before issuing the **ct** command, be certain that the remote terminal is attached to a modem that can answer the telephone.
2. If the user issuing the **ct** command does not have root authority, the port used for the connection must be a shared or delayed port. Otherwise, the remote login will fail. For more information on shared and delayed ports, see the **pshare** and **pdelay** commands. In addition, for the **ct** command to succeed on a shared or delayed port, the user invoking the command must be a member of the UNIX−to−UNIX copy program (uucp) user group.

The **ct** command is not as flexible as the BNU **cu** command. For example, the user cannot issue commands on the local system while connected to a remote system through the **ct** command. However, the **ct** command does have two features not available with the **cu** command:

- The user can instruct the **ct** command to continue dialing the specified telephone 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 to instruct the **ct** command to continue dialing each modem until a connection is established over one of the lines.

If the local user specifies alternate dialing paths by entering more than one number on the command line, the **ct** command tries each line listed in the BNU **Devices** file(s) (by default, the **/etc/uucp/Devices** file) until it finds an available line with appropriate attributes or runs out of entries. If there are no free lines, the **ct** command 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 one−minute intervals until the specified time is exceeded. The local user can override this prompt by specifying a time with the **−w***Number* flag when entering the command.

After the user logs off, the **ct** command prompts the user on the remote terminal with a reconnect option; the system can either display a new login prompt or drop the line.

Flags

−h	Prevents the **ct** command from hanging up the current line to answer an incoming call.
−s*Speed*	Specifies the rate at which data is transmitted. The default is 1200 baud.
−v	Allows the **ct** command to send a running narrative to standard error output.
−w*Number*	Specifies the maximum number of minutes that the **ct** command is to wait for a line. The command then dials the remote modem at one−minute intervals until the connection is established or until the specified time has elapsed.
−x*Number*	Starts debugging, which displays detailed information about the command's execution on standard error output on the local system. The *Number* variable specifies the debugging level, and is a single digit from 0 to 9. The recommended debugging level is 9.
TelephoneNumber	Specifies the telephone number of the modem attached to the remote terminal. The *TelephoneNumber* variable can include the digits 0 through 9, − (minus signs) representing delays, = (equal signs) representing secondary dial tones, * (asterisks), and # (pound signs). The telephone number can contain a maximum of 31 characters.

Examples

1. To dial a modem attached to a remote terminal with an internal telephone number, enter:

   ```
   ct 41589
   ```

   The internal telephone number of `4−1589` is dialed. The − (hyphen) is optional. The system responds:

   ```
   Allocated dialer at 1200 baud
   ```

```
Confirm hang_up? (y to hang_up)
```

2. To dial a modem attached to a remote terminal with a local telephone number, enter:

   ```
   ct -w3 9=5553017
   ```

   The **ct** command dials the local telephone number of `555-3017`, where dialing `9` is required to reach an outside dial tone. A three-minute wait is specified as the maximum number of minutes that the **ct** command is to wait for a line.

3. To dial a modem attached to a remote terminal with a long-distance telephone number, enter:

   ```
   ct -w5 9=12345557003
   ```

The command dials the long-distance telephone number of `1-234-555-7003`, where `9` is required to reach an outside dial tone. A five-minute wait is specified as the maximum number of minutes that the **ct** command is to wait for a line.

Files

/usr/bin/ct	Contains the **ct** command.
/etc/uucp/Devices	Lists information about available devices.
/etc/uucp/Dialcodes	Contains dialing code abbreviations.
/etc/uucp/Dialers	Defines modem dialers.
/etc/uucp/Systems	Lists accessible remote systems.
/etc/uucp/Sysfiles	Specifies alternate files to be used as **Systems**, **Devices**, and **Dialers** files.

Related Information

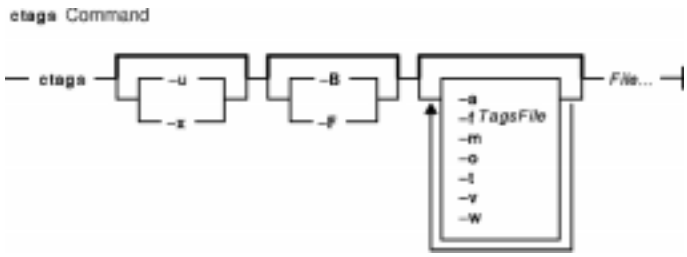
The **cu** command, **pdelay** command, **pshare** command, **tip** command.

ctags Command

Purpose

Makes a file of tags to help locate objects in source files.

Syntax



ctags [**−u** | **−x**] [**−B** | **−F**] [**−a**] [**−m**] [**−o**] [**−t**] [**−v**] [**−w**] [**−f** *TagsFile*] *File*. ..

Description

The **ctags** command creates a tags file for use with the ex and vi editors from the specified C, Pascal, FORTRAN, yacc, lex, and LISP source files. The tags file consists of locators of programming language specific objects (such as functions and type definitions) within the source files. A locator consists of the object name, the file in which it is defined, and either a basic regular expression or a line number that can be used in searching for the object definition. Specifiers are given in separate fields on the line, separated by spaces or tabs. Using the tags file, ex and vi can quickly find these object definitions.

The following file name suffixes are supported by the **ctags** command:

.c Treated as C−language source code and searched for C routine and macro definitions.

.h Treated as C−language source code and searched for C routine and macro definitions.

.f Treated as FORTRAN−language source code.

.l Treated as LISP−language source code if its first nonspace character is **[** (open bracket), **(** (open parenthesis), or **;** (semicolon). Treated as lex−language source code otherwise.

File names ending with any other suffixes are first examined to see if they contain any Pascal or FORTRAN routine definitions. If not, they are processed again as C−language source code. Files without a **.** (dot) suffix are processed as C−language source code.

The **main** tag 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 use of **ctags** practical in directories with more than one program.

Notes:

1. Recognition of the keywords **function**, an address specification for the **subroutine**, and **procedure** in FORTRAN and Pascal code ignores block structure. The **ctags** command may yield inadequate results if any two Pascal procedures have the same name, even though they are in different blocks.
2. The **ctags** command does not recognize **#if** and **#ifdef** statements.
3. If both the **−B** and **−F** options are specified, the last one specified will take

precedence.
4. The **−x** option takes precedence over any options (**−a**, **−u**, or **−f**) that would otherwise create a tags file.
5. When the **−v** option is specified, the **−x** option is implied.
6. The output of the **ctags** command is always sorted by object identifier.

Flags

−a Appends to the tags file. After appending, **ctags** sorts the tags file.

−B Causes **ctags** to use backward searching patterns (?. . .?).

−F Causes **ctags** to use forward searching patterns (/. . ./). This is the default searching pattern.

−f*TagsFile* Creates a tags file with the name specified by *TagsFile* instead of the default **tags** file.

−m Causes **ctags** to not create tags for macro definitions.

−o Causes **ctags** to generate line numbers for typedefs instead of a basic regular expression which is used in searching for the object definition.

−t Creates tags for typedefs. This flag is on by default due to standards conformance.

−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 flag may slow the processing of the command. (It is usually faster to simply rebuild the tags file.)

−v Produces an index of the form expected by the **vgrind** command on the standard output. This listing contains the function name, file name, and page number (assuming 64−line pages).

−w Causes **ctags** to suppress diagnostic warning messages.

−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 provides a simple, readable, function index. If you specify this flag, the **ctags** command does not build, update, or append a tags file, but writes to standard output.

Examples

1. To write the output of the **ctags** command to standard output for the C−language source files, **x.c**, **y.c**, and **z.c**, enter:

   ```
   ctags −x x.c y.c z.c
   ```

2. To create a tags file named **foo_tags** for all the C−language source files within the current directory, enter:

   ```
   ctags −f foo_tags *
   ```

3. To add additional tags,including type definitions, to the **foo_tags** tags file for the C−language source file **zip.c**, enter:

   ```
   ctags −utf foo_tags zip.c
   ```

Exit Status

The following exit values are returned:

0 Successful completion.

>0 An error occurred.

Files

tags Output tags file.

Related Information

The **ex** command, **lex** command, **vgrind** command, **vi** command, **yacc** command.

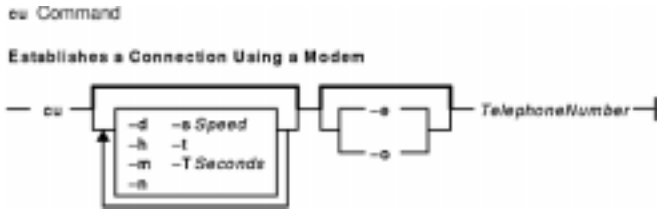
cu Command

Purpose

Connects directly or indirectly to another system.

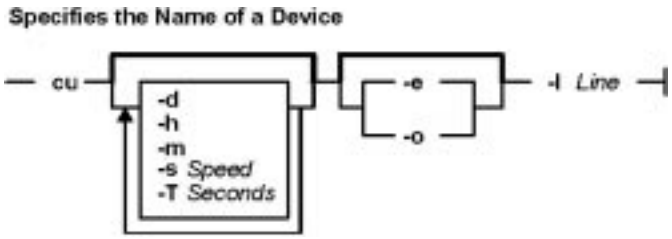
Syntax

To Establish a Connection Using a Modem



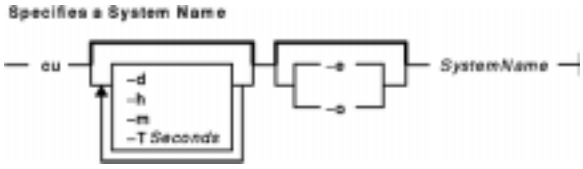
cu [**−d**] [**−h**] [**−m**] [**−T***Seconds*] [**−n**] [**−s***Speed*] [**−t**] [**−e** | **−o**] *TelephoneNumber*

To Specify the Name of a Device for a Connection



cu [**−d**] [**−h**] [**−m**] [**−T***Seconds*] [**−s***Speed*] [**−e** | **−o**] **−l***Line*

To Specify a System Name for a Connection



cu [**−d**] [**−h**] [**−m**] [**−T***Seconds*] [**−e** | **−o**] *SystemName*

Description

The **cu** command is a Basic Networking Utilities (BNU) command that connects one system to a terminal connected to either a UNIX or non−UNIX system. The connection can be established over a hardwired line or over a telephone line using 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.

After issuing the **cu** command from the local system, the user must press the Enter key and then log in to the remote system. After making the connection, 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. Internally, the program accomplishes this by initiating an output diversion to a file on the local system when a line from the remote system begins with ~> (tilde, greater than). The trailing ~> marks the end of the diversion. 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.

The **cu** command can be used to connect multiple systems, and commands can then be executed on any of the connected systems. For example, the user can 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 in and issuing the command. Commands can be executed on system X by prefixing the command with a single tilde (**~***Command*) and on system Y by prefixing the command with two tildes (**~~***Command*). 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 −n** command (to display the node name) on systems Z, X, and Y as follows:

```
$ uname −n
Z
$ ~!uname −n
X
$ ~~!uname −n
Y
```

Notes:

1. The **cu** command does not do integrity checking on data it transfers.
2. Data fields with special **cu** characters may not be transmitted properly.
3. The exit code is 0 for normal exit, otherwise, −1.

In addition to issuing regular commands on the remote system, the user can issue special **cu** command subcommands, which are preceded by a **~** (tilde). Use these subcommands to issue commands on the local system and to perform tasks such as transferring files between two UNIX systems. As soon as the user enters the **~!**, **~$**, **~%**, **~l**, or **~t** subcommand, the system displays the name of the local computer in a format similar to the following:

```
~[SystemName]/%
```

The user then enters the subcommand to be executed on the local computer.

Flags

−d	Prints diagnostic traces.
−e	Designates that even parity is to be generated for data sent to the remote system.
−h	Emulates local echo, supporting calls to other systems that expect terminals to be set to half−duplex mode.
−l*Line*	Specifies the name of a device to be used as the line of communication between the local and the remote system. This 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

the *Line* is taken from the **Devices** file(s) (by default, the **/etc/uucp/Devices** file).

When the **−l** and **−s** flags are used together, the **cu** command searches the **Devices** file(s) 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 hardwired asynchronous line (for example, **/dev/tty2**), in which case the *TelephoneNumber* parameter is not required. If the specified device is associated with a modem, a telephone number must be provided. Using this flag with the *SystemName* parameter rather than with *TelephoneNumber* parameter does not give the desired result.

Under ordinary circumstances, the user should not have to specify the transmission speed or a line or device. The defaults set when BNU is installed should be sufficient.

−m Instructs the **cu** command to ignore modem control signal data carrier detect (DCD).

−n For added security, prompts the user to provide the telephone number to be dialed, rather than taking it from the command line.

−o Designates that odd parity is to be generated for data sent to the remote system.

−s*Speed* Specifies the rate at which data is transmitted to the remote system (300, 1200, 2400, 4800, 9600, or 19200 baud).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 BNU **Devices** file(s) (by default, the **/etc/uucp/Devices** file). Most modems operate at 300, 1200, or 2400 baud, while most hardwired lines are set to 1200 baud or higher. When transferring data such as a file between a local and a remote system, a speed of 300 baud may occasionally be needed. The lower baud rate results in less interference on the line.

−t Used to dial an ASCII terminal that has been set to autoanswer. Appropriate mapping of carriage−return to carriage−return line feed pairs is set.

−T*Seconds* Specifies the maximum number of seconds to wait before timing out. The default is 45 seconds.

Note: You can also enter `WAIT=n` before any send string in the **Dialers** file. Where `n` is the number of seconds to wait before timing out.

Parameters

SystemName The name of the remote system, recognized by BNU, with which a connection is established. A system name can be used rather than a telephone number; in that case, the **cu** command obtains an appropriate hardwired line or telephone number from the BNU **Systems** file(s) (by default, the **/etc/uucp/Systems** file). System names must be ASCII characters only.

Note: Do not use the *SystemName* flag with the **−l** flag and the **−s** flag. If you do, the **cu** command connects to the first available line for the requested system name, ignoring the specified line and speed.

TelephoneNumber The telephone number used to establish a remote connection using a modem. This entry can be either a local or a long−distance telephone number.

Subcommands

The **cu** command transmit process interprets lines beginning with a **~** (tilde) in the following ways:

~! Returns the user to an interactive shell on the local system. Toggle between the local and remote systems using **~!** (remote to local) and Ctrl−D (local to remote).

~%break Transmits a break sequence to the remote system. The break can also be specified as **~%b**.

~%cd *DirectoryName* Changes the directory on the local system from the current directory to the directory

	specified by the *DirectoryName* variable.
~%debug	Toggles the **−debug** flag on or off; this can also be specified as **~%d**.
~%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.
~%put *From* [*To*]	Copies the *From* file on the local system to the *To* file on the remote system. If the *To* variable 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. Only ASCII files can be transferred using this subcommand. The use of the **~%put** subcommand requires the **stty** command and the **cat** command 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 in the transmitted data. There is an artificial slowing of transmission by the **cu** command during the **~%put** operation so that loss of data is unlikely.
~%take *From* [*To*]	Copies the *From* file on the remote system to the *To* file on the local system. If the *To* variable 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. Only ASCII files can be transferred using this subcommand. The use of the **~%take** subcommand requires the **echo** command and the **cat** command 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.
~.	Logs the user off the remote computer and then terminates the remote connection. Usually the connection terminates when you log off the remote computer. However, with some types of interconnection hardware, it may be necessary to use a **~.** to terminate the conversation after the normal logoff sequence has been used.
~!*Command*	Executes, on the local system, the command denoted by the *Command* variable.
~$*Command*	Runs, on the local system, the command denoted by the *Command* variable, then sends the command's output to the remote system for execution.
~l	Prints the values of the **TERMIO** structure variables for the remote communication line. This is useful for debugging.
~t	Prints the values of the **TERMIO** structure variables for the user's terminal. This is useful for debugging.
~~*String*	Sends the string denoted by the *String* variable to the remote system.

Examples

The following are examples of connecting to a remote system.

1. To connect to a remote system, enter:

   ```
   cu venus
   ```

   In this example, you are connected to the remote system `venus`. System `venus` must be listed in one of the local **Systems** files (by default, the **/etc/uucp/Systems** file or one of the **Systems** files listed for the **cu** command in the **/etc/uucp/Sysfiles** file).

2. To dial a remote system and set the baud rate, enter:

   ```
   cu −s1200 9=12015558391
   ```

   In this example, you dial a remote system whose telephone number is `1−201−555−8391`, where

dialing `9` is required to get an outside dial tone. The baud rate is set to `1200`.

3. To log in to a system connected by a hardwired line asynchronous line, enter:

   ```
   cu -l /dev/tty2
   ```

   The **cu** command contacts the system connected to the `tty2` device.

4. To dial a remote system with a specified line and a specific speed, enter:

   ```
   cu -s 1200 -l tty3
   ```

   The command contacts the system connected to the `tty3` device, using a speed of `1200` baud.

5. To dial a remote system using a specific line associated with a modem, enter:

   ```
   cu -l cul4 9=12015558391
   ```

In this example, you dial a remote system whose telephone number is `1-201-555-8391`, where dialing `9` is required to get an outside dial tone. The **cu** command uses the modem connected to the `cul4` device.

1. To display the contents of a file after logging in to the remote system, enter:

   ```
   ~!pg /usr/msg/memos/file10
   ```

   The **~!** subcommand executes the **pg** command on the local system, displaying the contents of the `file10` file in the `/usr/msg/memos` directory on the local system.

2. To copy a file from the local system to the remote system without changing the name of the file, enter:

   ```
   ~%put /home/amy/file
   ```

   The `/home/amy/file` file is copied from the local system to the remote system without changing the name of the file.

3. To copy a file from the local system to the remote system and change the file name, enter:

   ```
   ~%put /home/amy/file /home/amy/tmpfile
   ```

   The `/home/amy/file` file is copied from the local system to the remote system and the file name changed to `/home/amy/tmpfile`.

4. To copy a file from the remote system to the local system without changing the name of the file, enter:

   ```
   ~%take /home/jeanne/test1
   ```

   The `/home/jeanne/test1` file is copied from the remote system to the local system without changing the name of the file.

5. To copy a file from the remote system to the local system and change the file name, enter:

   ```
   ~%take /home/jeanne/test1 /usr/dev/jeanne/tmptest
   ```

In this example, the `/home/jeanne/test1` file is copied from the remote system to the local system and the file name changed to `/usr/dev/jeanne/tmptest`.

Files

/etc/locks	Prevents multiple use of device.
/usr/bin/cu	Specifies the path name of the **cu** command.
/bin/cu	Specifies a symbolic link to the **/usr/bin/cu** command.
/etc/uucp/Devices	Contains information about available links.
/etc/uucp/Dialcodes	Contains dialing code abbreviations.
/etc/uucp/Dialers	Controls initial handshaking on a link.
/etc/uucp/Permissions	Contains access permission codes.
/etc/uucp/Systems	Lists accessible remote systems.
/etc/uucp/Sysfiles	Specifies alternate files to be used as **Systems**, **Devices**, and **Dialers** files.

Related Information

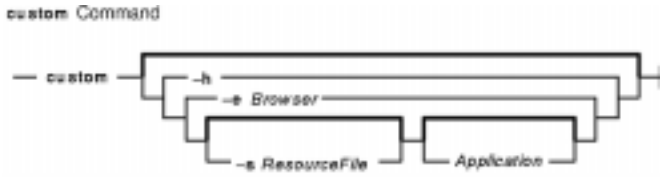
The **cat** command, **ct** command, **echo** command, **rmail** command, **stty** command, **uname** command, **uucp** command, **uuname** command, **uupick** command, **uustat** command, **uuto** command, **uux** command.

custom Command

Purpose

Enables users to customize X applications.

Syntax



custom [**−h** | **−e** *Browser* | [**−s** *ResourceFile*] [*Application*]]

Description

The **custom** command starts the customizing tool, which is used to customize various aspects of applications.

The customizing tool can change the look of an application. It provides a user−friendly way to add resource values to your **.Xdefaults** file. *Resources* are customizable items such as colors, fonts, and other attributes that allow you to customize resources of a client application. Each application has its own set of unique resources, which are listed in an **app−custom** file. The customizing tool describes the resources available for modification for an application and the possible resource values you can select.

Flags

−h	Provides command line help.
−e*Browser*	Calls one of the standalone browsers. Valid values for *Browser* are **color**, **font**, **cursor**, and **picture**.
−s*ResourceFile*	Specifies the resource file from which to load and save resource settings. If the **−s** flag is not specified, the default is to load the values from the resource database stored in the **RESOURCE_MANAGER** property on the X server. If this database does not exist, then **$HOME/.Xdefaults** is loaded.

Most standard X Toolkit command−line options are understood by the **custom** command. See the following table that lists the standard command−line options:

Option	Resource	Value	Sets
−bg	*background	Next argument	Background color
−background	*background	Next argument	Background color
−bd (1)	*borderColor	Next argument	Border color
−bordercolor (1)	*borderColor	Next argument	Color of border
−bw	.borderWidth	Next argument	Width of border in pixels
−borderWidth	.borderWidth	Next argument	Width of border in pixels

−display	.display	Next argument	Server to use
−fn (2)	*font	Next argument	Font name
−font (2)	*font	Next argument	Font name
−fg	*foreground	Next argument	Foreground color
−foreground	*foreground	Next argument	Foreground color
−geometry	.geometry	Next argument	Size and position
−iconic	.iconic	On	Start as an icon
−name	.name	Next argument	Name of application
−reverse	*reverseVideo	On	Reverse video
−rv	*reverseVideo	On	Reverse video
+rv	*reverseVideo	Off	No Reverse video
−selection− Timeout	.selection− Timeout	Next argument	Selection timeout
−synchronous	*synchronous	On	Synchronous debug mode
+synchronous	*synchronous	Off	Synchronous debug mode
−title	.title	Next argument	Title of application
−xrm	value of argument	Next argument	Depends on argument
−xnllanguage	.xnlLanguage	Next argument	Locale

1 These options often have no visible effect on AIXwindows applications if the AIXwindows Window Manager (mwm) is running.

2 Motif applications do not generally respond to these options.

> **Note:** Resources beginning with an* (asterisk) set the resource of every widget in the application to the same value. Resources that begin with a . (period) set the resources of only the application's top−level Shell widget.

Parameters

Application Specifies the name or class of the application to customize.

Examples

1. To start the customizing tool and use prompts to choose the application to customize, enter the following:

   ```
   custom
   ```

2. To start the customizing tool to modify the **app−defaults** file of the **xcalc** application, enter the following:

   ```
   custom −s
   /usr/lib/X11/app−defaults/XCalc xcalc
   ```

Resources

The customizing tool has the following application resources:

listOfApps This resource is used to display the application names on the starting dialog. The application name and corresponding **app−custom** file must be listed in pairs with the following syntax:

```
Application:app−custom
[,Application:app−custom]...
```

For example:

```
Custom.listOfApps:
xclock:XClock,custom:Custom
```

You can specify a maximum of 100 applications.

colorEditor*rgbtxtPath This resource specifies the full path name of the **rgb.txt** file that the X server uses to define named colors. The default value is **/usr/lib/X11/rgb.txt**, which is correct for an X server running on a display that is directly attached to your system unless you are using an Xstation with AIX Xstation Manager/6000 Version 1.3, then you need to set the value of this resource to **/usr/lpp/x_st_mgr/bin/rgb.txt**.

windowSearchDepth The customizing tool must determine the top−level shell window of the application. It starts with the root window and conducts a recursive search to a depth of three windows by default. This default can be changed using the **windowSearchDepth** resource.

timeout The Instant Changes button is grayed out until communication with the application is established. The amount of time to wait for the application to contact the customizing tool is controlled by the **Custom*timeout** resource.

resourceFile The resource file is where your resource changes are saved. The default is **$HOME/.Xdefaults**. The **−s** flag allows the user to override this value.

appCustomPath This resource specifies where the customizing tool is to look for the **app−custom** file. The **appCustomPath** string consists of a series of possible file names separated by colons. Within each name, the following values can be substituted:

%N
: Name of the **app−custom** file (usually the same as the class name of the application)

%T
: **"app−custom"**

%L
: Locale in which **custom** is running.

%l
: Language part of the locale.

%t
: Territory part of the locale.

%c
: Codeset part of the locale.

%:
: A : (colon).

%%
: A % (percent sign).

$envvar
: Value of the named environment variable.

${envvar}
: Value of the named environment variable.

$$
: A $ (dollar sign).

The default value of **appCustomPath** is as follows:

```
$HOME/%L/%T/%N:\
$HOME/%T/%N:\
/usr/lib/X11/%L/%T/%N:\
/usr/lib/X11/%T/%N
```

topEditHighlight, **bottomEditHighlight**, **foregroundEditHighlight**, **backgroundEditHighlight**

The Browser button is highlighted when a browser is called and unhighlighted when a browser is canceled. These resources set the highlight color for the top shadow, bottom shadow, foreground, and background of the Browser button.

pictureEditor*editor

You can edit the bitmap or pixmap by pressing the Edit Picture button on the Pictures browser window. The editor is a separate application that exists on your system. It is called on your behalf. The **Custom*pictureEditor*editor** resource determines which editor commands to choose from. This resource accepts a list of commands separated by \n's (backslash 'n's). The first command that identifies an existing program that the user has permission to execute is used. The file name in the Chosen Picture text field is passed as a parameter to the editor when it is invoked. The default setting for this resource is:

```
Custom*pictureEditor*editor:
/usr/dt/bin/dticon  -f \n
/usr/lib/X11/bitmap
```

> **Note:** The default editor, **/usr/dt/bin/dticon** only exists if the Common Desktop Environment (CDE) is installed. It edits both bitmaps (monochrome images) and pixmaps (color images). The **dticon** command accepts bitmaps stored in either the X Pixmap Version 2 Enhanced (XPM2) format which was used by the X Desktop (**xdt**) application shipped in AIXwindows Version 1.2.5, or X Pixmap Version 3 (XPM3) − a new XPG3 compliant format used by CDE. However, it requires pixmap images be stored in the XPM3 format. CDE has documented tools that can convert pixmaps from the XPM2 to the XPM3 format.

The **/usr/bin/X11/bitmap** command is an unsupported sample program that accepts bitmaps in either the XPM2 or XPM3 formats. It does not support pixmap editing. Be sure that the Bitmap app−defaults file has been installed in the **/usr/lib/X11/app−defaults** directory before invoking the **bitmap** command. If not, issue the following command in the **/usr/lpp/X11/Xamples/programs/bitmap** directory:

```
xmkmf;
make install
```

The following object names (and their class names) can be used to customize this tool:

```
custom (Custom)
      startupDialog_popup (XmDialogShell)
                startupDialog (XmSelectionBox)
      helpDialog_popup (XmDialogShell)
                helpDialog (XmForm)
      saveDialog_popup (XmDialogShell)
                saveDialog (XmSelectionBox)
      colorEditor_popup (XmDialogShell)
                colorEditor (XibmColorEditor)
      fontEditor_popup (XmDialogShell)
                fontEditor (XibmFontEditor)
      pictureEditor_popup (XmDialogShell)
                pictureEditor (XibmPictureEditor)
      cursorEditor_popup (XmDialogShell)
                cursorEditor (XibmCursorEditor)
```

```
selectmanyEditor_popup (XmDialogShell)
          selectmanyEditor (XibmSelectManyEditor)
filenameEditor_popup (XmDialogShell)
          filenameEditor (XmFileSelectionBox)
mainWindow (XmMainWindow)
          menubar (XmRowColumn)
          form (XmForm)
                  appClassLabel (XmLabel)
                  appClass (XmLabel)
                  groupMenuLabel (XmLabel)
                  groupMenu (XmRowColumn)
                  scrolledGroup (XmScrolledWindow)
                          scrolledGroupForm (XmForm)
                                 (XmLabelGadget)
                                 TypeField (XmTextField)
                                 TypeButton (XmPushButton)
```

where *Type* can be one of the color, font, picture, cursor, selectmany, filename, selectone, string, or number data type values.

Exit Status

This command returns the following exit values:

0 Indicates successful completion.

>0 Indicates an error occurred.

Files

/usr/bin/X11	Is the path from which you run the **custom** command once the custom package is installed.
/usr/lib/X11/app−custom	Contains information about resources for individual applications.
/usr/lib/X11/*locale***/app−custom**	Contains information about resources for individual applications that is translated for specific locales.
/usr/lib/X11/app−defaults/Custom	Contains default settings for the Customizing Tool.
/usr/lib/X11/*locale***/app−defaults/Custom**	Contains default settings for the Customizing Tool in locales that require special settings.

Related Information

How to Start the Customizing Tool in *AIX Version 4.3 AIXwindows Programming Guide*.

cut Command

Purpose

Writes out selected bytes, characters, or fields from each line of a file.

Syntax

cut { **−b** *List* [**−n**] | **−c** *List* | **−f** *List* [**−s**] [**−d** *Character*] } [*File* ...]

Description

The **cut** command cuts bytes, characters, or fields from each line of a file and writes these bytes, characters, or fields to standard output. If you do not specify the *File* parameter, the **cut** command reads standard input.

You must specify either the **−b**, **−c**, or **−f** flag. The *List* parameter is a comma−separated, blank−separated, or hyphen−separated list of integer numbers (in increasing order). The hyphen separator indicates ranges. The following entries are some example *List* parameters which could refer to bytes, characters, or fields:

```
1,4,7
1−3,8
−5,10
3−
```

where `−5` is a short form for the first through fifth and `3−` is a short form for the third through last.

If using the **cut** command on fields, the length of the fields specified by the *List* parameter can vary from field to field and line to line. The position of the field delimiter character, such as a tab character, determines the length of a field.

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

−b*List*	Specifies byte positions. These byte positions ignore multibyte character boundaries unless the **−n** flag is also specified.
−c*List*	Specifies character positions. For example, if you specify `−c 1−72,` the **cut** command writes out the first 72 characters in each line of the file.
−d*Character*	Uses the character specified by the *Character* variable as the field delimiter when you specify the **−f** flag. You must put quotation marks around characters with special meaning to the shell, such as the space character.
−f*List*	Specifies a list of fields assumed to be separated in the file by a delimiter character, which is by default the tab character. For example, if you specify `−f 1,7,` 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.

−n Suppresses splitting of multibyte characters. Use only with the **−b** flag. If the last byte of a character falls within the range denoted by the *List* variable of the **−b** flag, the character is written; otherwise, the character is excluded.

−s Suppresses lines that do not contain delimiter characters. Use only with the **−f** flag.

Exit Status

This command returns the following exit values:

0 All input files were output successfully.

>0 An error occurred.

Examples

1. To display several fields of each line of a file, enter:

   ```
   cut −f 1,5 −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 (`−f 1,5`) separated by colons (`−d :`).

   For example, if the **/etc/passwd** file looks like this:

   ```
   su:*:0:0:User with special privileges:/:/usr/bin/sh
   daemon:*:1:1::/etc:
   bin:*:2:2::/usr/bin:
   sys:*:3:3::/usr/src:
   adm:*:4:4:System Administrator:/var/adm:/usr/bin/sh
   pierre:*:200:200:Pierre Harper:/home/pierre:/usr/bin/sh
   joan:*:202:200:Joan Brown:/home/joan:/usr/bin/sh
   ```

   The **cut** command produces:

   ```
   su:User with special privileges
   daemon:
   bin:
   sys:
   adm:System Administrator
   pierre:Pierre Harper
   joan:Joan Brown
   ```

2. To display fields using a blank separated list, enter:

   ```
   cut −f "1 2 3" −d : /etc/passwd
   ```

   The **cut** command produces:

   ```
   su:*:0
   daemon:*:1
   bin:*:2
   sys:*:3
   adm:*:4
   pierre:*:200
   joan:*:202
   ```

Files

/usr/bin/cut Contains the **cut** command.

Related Information

The **grep** command, **paste** command, **sh** command.

Files Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes what files are and how they are stored by the operating system.

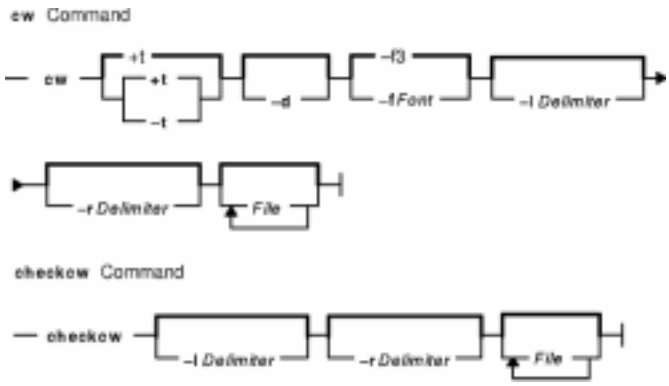
Input and Output Redirection Overview in *AIX Version 4.3 System User's Guide: Operating System and Devices* describes how to redirect input and output.

cw or checkcw Command

Purpose

Prepares constant−width text for the **troff** command.

Syntax



cw [**+t** | **−t**] [**−d**] [**−f** *Font*] [**−l** *Delimiter*] [**−r** *Delimiter*] [*File ...*]

checkcw [**−l** *Delimiter*] [**−r** *Delimiter*] [*File ...*]

Description

The **cw** command preprocesses any specified **troff** files containing English−language 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 sign) as one of the input file names. The **cw** command writes to standard output.

Because output resulting from this command resembles the output of line printers and workstations, use this command to typeset examples of programs and computer output for user manuals and programming text. The **cw** command produces distinctive output when used with the Times Roman font.

The CW font contains a nonstandard set of characters. Any text typeset with this font requires different character and interword spacing from that used for standard fonts. Therefore, you must use the **cw** command to preprocess documents that use the CW font.

The CW font contains the following 94 ASCII printing characters:

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
!$%&()`'*+@.,/:;=?[]|−_^~"<>{}#\
```

This font also contains 11 non−ASCII characters represented by 4−character **troff** strings (in some cases attaching these strings to nonstandard graphics).

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 the requests 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 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 *Option−list*	Concatenates all the options (delimited like **troff** macro options), with the odd−numbered options set in the CW font and the even−numbered options set in the prevailing font.
.PC *Option−list*	Acts the same as the **.CP** macro, except the even−numbered options are set in CW font and the odd−numbered options are set in the prevailing font.

The **.CW** and **.CN** requests should bracket text that is to be typeset as is, using the CW font. 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 4−character names listed previously. In particular, the **cw** command causes all **.** (periods) and **'** (apostrophes) at the beginning of lines, and all \ (backslashes) and ligatures (such as fi and ff), to be hidden from the **troff** command. 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. These requests are meant to enclose words or phrases that are 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 as well as the **.CW**/**.CN** pairs are properly balanced. It prints out all lines in the selection with the unmatched delimiters.

Notes:

1. The **.** (period) or \ (backslash)delimiter characters should not be used.
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** macro package and **mv** macro package contain definitions of **.CW** and **.CN** macros that are adequate for most users. If you define your own macros, make sure that the **.CW** macro starts 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 1 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 column inch.
6. Documents that contain CW text can also contain tables and equations. In this case, the order of preprocessing must be the **cw** command, **tbl** command, and **eqn** command. Usually, the tables do not contain CW text, although it is possible to have elements in the table set in the CW font. Ensure that the **cw** command does not modify the **tbl** command format information. Attempts to set equations in the CW font are usually unsuccessful.
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.
8. Some devices such as the IBM 3816 Pageprinter do not have a CW font. You receive

a `troff can't open /usr/lib/font/devNAME/CW.out` message for these devices. The **troff** command uses the font in font position 3 as the CW font.

Parameters

File Specifies **troff** English−language text files to be preprocessed by the **cw** command to produce constant−width characters in the output file.

File Specifies **troff** English−language text files to be preprocessed by the **checkcw** command to check right and left delimiters as well as **.CW** and **.CN** pair balance.

Flags

+t Turns the transparent mode on (this is the default).

−t Turns the transparent mode off.

−d Displays the current flag settings on the standard error output in the form of **troff** comment lines. This flag is meant for debugging.

−f*Font* Replaces the value of the *Font* variable with the **cw** command font (the default equals 3, which replaces the bold font). The **−f5** flag is commonly used for formatters that allow more than four simultaneous fonts.

Note: This flag is useful only on the command line.

−l*Delimiter* Sets the left delimiter as the 1− or 2−character string specified by the *Delimiter* variable. The left delimiter is undefined by default.

−r*Delimiter* Sets the right delimiter to that specified by the *Delimiter* variable. The right delimiter is undefined by default. The left and right delimiters can (but need not) be different.

Related Information

The **eqn** command, **mmt** command, **tbl** command, **troff** command.

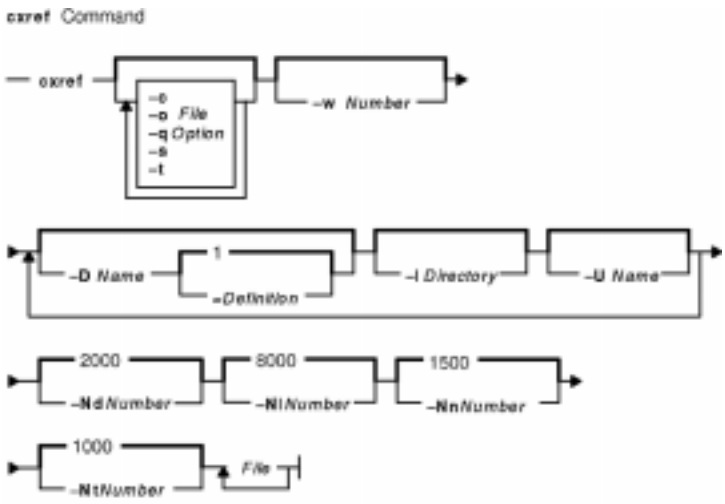
The **mm** macro package, **mv** macro package.

cxref Command

Purpose

Creates a C and C++ program cross−reference listing.

Syntax



cxref [**−c**] [**−o** F*ile*] [**−q***Option*] [**−s**] [**−t**] [**−w** *Number*] [[**−D** *Name* [=*Definition*]] [**−I** *Directory*] [**−U** *Name*]] ... [**−Nd***Number*] [**−Nl***Number*] [**−Nn***Number*] [**−Nt***Number*] *File* ...

Description

The **cxref** command analyzes C and C++ program *File*s and creates a cross−reference table, using 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). The formal parameters in a function definition are always listed; but if a function is only prototyped and not defined, the parameters are not listed. When a reference to a symbol is that symbol's declaration, an * (asterisk) precedes it.

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*Number*	Makes the listing *Number* columns wide, where *Number* is a decimal integer greater than or equal to 51. If *Number* is less than 51, the listing will be 80 columns wide.
−**Nd***Number*	Changes the dimension table size to *Number*. The default is 2000.
−**Nl***Number*	Changes the number of type nodes to *Number*. The default is 8000.
−**Nn***Number*	Changes the symbol table size to *Number*. The default is 1500.
−**Nt***Number*	Changes the number of tree nodes to *Number*. The default is 1000.

In addition, the **cxref** command recognizes the following flags of the **cpp** command (macro preprocessor):

−D *Name*[=*Definition*]	Defines *Name* as in a **#define** directive. The default definition is 1.
−I *Directory*	Looks first in directory, then looks in the directories on the standard list for **#include** files with names that do not begin with a slash (/) (see the **cpp** command).
−U *Name*	Removes any initial definition of *Name*, where *Name* is a reserved symbol predefined by the preprocessor.
−q*Option*	Pass **−q***Option* to the preprocessor. For example, `−qmbcs` sets multibyte mode specified by the current locale, and `−qidirfirst` modifies the search order for files included with the **#include file_name** directive.

Examples

To provide a combined cross−reference listing of `stdin1.c` and `stdin2.c`, making the output 80 columns wide, enter:

```
cxref −c −t stdin1.c stdin2.c > output
```

Files

/usr/ccs/lib/xpass Special version of C compiler first−pass.

/usr/ccs/bin/cxref Contains the **cxref** command.

Related Information

The **cpp** command.

Vos remarques sur ce document / Technical publication remark form

Titre / Title : Bull AIX Commands Reference Vol.1 ac to cxref

Nº Reférence / Reference Nº : 86 A2 38JX 02

Daté / Dated : April 2000

ERREURS DETECTEES / ERRORS IN PUBLICATION

AMELIORATIONS SUGGEREES / SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Vos remarques et suggestions seront examinées attentivement.
Si vous désirez une réponse écrite, veuillez indiquer ci-après votre adresse postale complète.

Your comments will be promptly investigated by qualified technical personnel and action will be taken as required.
If you require a written reply, please furnish your complete mailing address below.

NOM / NAME : _____ Date : _____

SOCIETE / COMPANY : _____

ADRESSE / ADDRESS : _____

_____

Remettez cet imprimé à un responsable BULL ou envoyez-le directement à :

Please give this technical publication remark form to your BULL representative or mail to:

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

Technical Publications Ordering Form

Bon de Commande de Documents Techniques

To order additional publications, please fill up a copy of this form and send it via mail to:
Pour commander des documents techniques, remplissez une copie de ce formulaire et envoyez-la à :

BULL ELECTRONICS ANGERS
CEDOC
ATTN / MME DUMOULIN
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

Managers / Gestionnaires :
Mrs. / Mme : **C. DUMOULIN** +33 (0) 2 41 73 76 65
Mr. / M : **L. CHERUBIN** +33 (0) 2 41 73 63 96

FAX : +33 (0) 2 41 73 60 19
E–Mail / Courrier Electronique : srv.Cedoc@franp.bull.fr

Or visit our web site at: / Ou visitez notre site web à:
`http://www–frec.bull.com` (PUBLICATIONS, Technical Literature, Ordering Form)

CEDOC Reference # Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté	**CEDOC Reference #** Nº Référence CEDOC	**Qty** Qté
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
__ __ ____ _ [__]		__ __ ____ _ [__]		__ __ ____ _ [__]	
[_ _] : **no revision number means latest revision** / pas de numéro de révision signifie révision la plus récente					

NOM / NAME : ____________________ Date : __________

SOCIETE / COMPANY : ____________________

ADRESSE / ADDRESS : ____________________

____

PHONE / TELEPHONE : ____________________ FAX : __________

E–MAIL : ____________________

For Bull Subsidiaries / Pour les Filiales Bull :
Identification: ____________________

For Bull Affiliated Customers / Pour les Clients Affiliés Bull :
Customer Code / Code Client : ____________________

For Bull Internal Customers / Pour les Clients Internes Bull :
Budgetary Section / Section Budgétaire : ____________________

For Others / Pour les Autres :
Please ask your Bull representative. / Merci de demander à votre contact Bull.

PLACE BAR CODE IN LOWER LEFT CORNER

BULL ELECTRONICS ANGERS
CEDOC
34 Rue du Nid de Pie – BP 428
49004 ANGERS CEDEX 01
FRANCE

ORDER REFERENCE
86 A2 38JX 02

Bull

Utiliser les marques de découpe pour obtenir les étiquettes.
Use the cut marks to get the labels.

AIX
AIX Commands
Reference Vol.1
ac to cxref

86 A2 38JX 02

AIX
AIX Commands
Reference Vol.1
ac to cxref

86 A2 38JX 02

AIX
AIX Commands
Reference Vol.1
ac to cxref

86 A2 38JX 02