

CramSession

The Original Study Guide



Over
4 Million
Downloaded

CompTIA

Linux+

CompTIA Linux+

XKO-001 XKO-001 XKO-001 XKO-001 XKO-001

Written by **Subject
Matter Experts**

Your **Trusted
Study Resource** for
Technical Certification

The **Most Popular
Study Guide** on the web

Table of Contents

Introduction	4
Planning the Implementation	5
The Parts.....	5
Determining Requirements	5
Kernel Versioning.....	6
Benefits of Using Linux	6
Benefits of Using Linux	7
Installation	7
Source Location	7
Boot Disk.....	7
Installation Options	7
<i>Workstation/Server/Custom</i>	7
<i>Partitioning</i>	8
<i>Filesystems</i>	8
<i>Security Options</i>	9
X-Windows	9
Boot Loader	9
Kernel.....	9
Configuration	10
Network Settings.....	10
<i>LAN</i>	10
<i>Dial</i>	10
X-Windows.....	11
Internet Services	11
<i>Inetd</i>	11
<i>FTP</i>	11
<i>Web Server</i>	12
<i>NFS</i>	12
<i>POP</i>	12
<i>Rlogin</i>	12
<i>SMB</i>	13
<i>Sendmail</i>	13
<i>Telnet</i>	14
<i>TFTP</i>	14
Printers and Other Hardware	14

Other Configuration Files 15

LILO 16

Modules..... 16

Modules..... 17

Administration 17

Users and Groups..... 17

The Root User..... 18

Files..... 19

The Filesystem..... 20

Locations 20

Directories and Files..... 20

Managing Services..... 21

Print Queues 22

Connecting to Other Machines 22

Shell Scripting 22

System Maintenance 23

Storage..... 23

Creating Filesystems..... 23

Fixing Filesystems..... 24

Mounting..... 24

Scheduling Tasks..... 24

Patches 25

Processes 26

Logs 27

Backups 27

TAR 27

Dump/Restore 28

Security 28

Physical..... 28

Passwords..... 28

System..... 29

Troubleshooting 30

Documentation 30

Tools 30

Diagnosis and Repair..... 31

Boot..... 31

Resource..... 32

Dependencies 32

Software/Configuration..... 33

Hardware..... 33

File Systems..... 33

Backups..... 33

Identify, Install, and Maintain System Hardware..... **34**

Resources..... 34

IRQ 34

Memory..... 35

CPU/Motherboard 35

Hard Drives 35

SCSI 35

IDE..... 36

Network..... 36

Peripheral Devices..... 38

Introduction

The CompTIA Linux+ exam is geared towards people with 6 months of Linux work experience.

Breakdown of sections is as follows (from CompTIA website)

Domain	% Examination
Planning the Implementation	4.00%
Installation	12.00%
Configuration	15.00%
Administration	18.00%
System Maintenance	14.00%
Troubleshooting	18.00%
Identify, Install, and Maintain System Hardware	19.00%

Within this study guide, examples will be shown in the courier font. Sometimes they will be all alone, such as

```
cat /etc/passwd
```

Other times, they may be prefaced by a \$ or a #, as in

```
# cat /etc/passwd
```

When written as such, the \$ and # are not to be typed: they indicate that the command is being run as a normal user (\$) or the super user, root (#). The command may appear with a comment after it to save space:

```
$ cat /etc/passwd # write the password file to the console
```

The # after the command denotes that everything following is a comment, and is not to be typed in. When the example contains both commands and other text, it is a copy of a session, so only the commands (beginning with \$ or # are to be typed in).

```
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
...
$
```

Here, the "cat /etc/passwd" command was typed in, and the output is shown. Since the command will also generate more information than is needed for the example, the ellipsis (...) indicate that some output was omitted.

Items in **bold** are important terms to remember.

UNIX has a built in system called the **man pages**. The man command is used to pull up information on a program (so `man man` will teach you about man). Sometimes multiple pages will be written for the same name, but with different uses. The command will usually be written as `command(X)` where X is a number or letter denoting the **section**. Pull up the man page with `man X command`.

Planning the Implementation

The Parts

The term Linux actually refers to the Kernel, which is the operating system itself. Software that runs on top of the Kernel is called an **application**, and includes items like web servers and mail clients. Since collecting all the appropriate software needed to make a Linux installation useful (shells, administration tools, etc.) is a time consuming task, **Distributions** are made that package the Linux kernel, and relevant software and applications. Examples of distributions are RedHat, Debian, Mandrake, and Corel Linux.

Determining Requirements

Linux owes its versatility to the wide availability of software that runs on it. Understanding what software to use to solve what problem is key to maximizing the utility of Linux.

- ❖ Web server - Apache (<http://www.apache.org>) is the most popular web server
- ❖ Web proxy - To better control web usage and to allow for caching of frequently accessed pages, Squid (<http://www.squid-cache.org>) is used.
- ❖ File Sharing - Linux can be made to look like an NT server with respect to file and print sharing. Samba (<http://www.samba.org>) is the software that does this.
- ❖ Email - Linux excels at handling email. Sendmail (<http://www.sendmail.org>) is the most widely used Mail Transfer Agent (MTA). Qmail (<http://www.qmail.org>) and PostFix (<http://www.postfix.org>) are alternatives. Regardless of your choice, a basic knowledge of Sendmail is required both for the exam, and for everyday work.
- ❖ DNS - The Domain Name Service provides mappings between names and IP addresses, along with distributing network information (i.e., mail servers). BIND (<http://www.isc.org/products/BIND/>) is the most widely used name server.
- ❖ X-Windows - XFree86 (<http://www.xfree86.org>) is the GUI system common to most Linux installations, though there are some commercial X versions that support more esoteric hardware. It provides the framework to build graphical applications. More information on the use and functionality of X is provided later.

Though there are always several software packages to solve the same problem, using the more popular one is usually the best choice because more support exists. Once you become familiar with the operation of the product, using the other packages will be made easier.

With the exception of some commercial products, the source code for all Linux applications is readily available. This increases the platforms that can run Linux, since most software just needs to be compiled to work. Currently, Linux runs on x86, Alpha, SPARC, PowerPC, and MIPS processors, among many others. Choosing the appropriate platform is usually a matter of sizing up the application, and determining the cost of various alternatives.

Most mainstream hardware is supported under Linux. Since the majority of device drivers are written by Linux users, the more popular the device the more likely there is a driver. Linux vendors tend to distribute a list of supported hardware:

- ❖ RedHat - <http://hardware.redhat.com/hcl/>
- ❖ X Windows - <http://www.xfree86.org/4.0.2/Status.html>
- ❖ Mandrake - <http://www.linux-mandrake.com/en/fhard.php3>
- ❖ Debian - <http://www.debian.org/doc/FAQ/ch-compat.html>

The various portions of software, including the kernel and distributions themselves, can be licensed under one of several methods:

- ❖ Commercial - like Microsoft Windows, you own the right to use the software. In the Linux world, this is rare.

- ❖ GPL (<http://www.gnu.org>)- The GNU Public License - The basis of the GPL is that the source code must be readily available to the user, at no charge (other than media and handling). Furthermore, any additions to, or inclusions of GPL'ed software falls, under the GPL.
- ❖ BSD - The BSD license is much like the GPL except that it does not place heavy restrictions on the redistribution of source or any modifications.
- ❖ Freeware - The author retains any source and is under no obligation to release it. There is no charge, though, for use of the software.

Software, where the source is freely available, is said to be **open sourced**. **Closed source** software is thus software where the source is not freely available.

Software can be distributed in source or binary form. Where it is in source form, instructions are usually given to compile it, or a **Makefile** (script file of computer instructions to build the binary version) is provided.

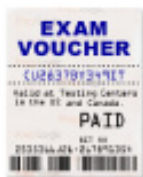
Binary distributions usually take one of three forms. A **tarball** is a compressed collection of software, much like a .zip file. The term takes its name from the utility, **tar**, used to bunch the files into one file, which is then compressed. This form of binary distribution is also used to distribute source. It has the limitation that it cannot carry any dependency information (i.e., you will need the ABC library to run), nor can it execute any instructions (such as creating users for you). The Slackware distribution uses this form, though it will look for the presence of certain scripts within the tarball to be executed. Furthermore, there is no easy way to go back and determine what package a file belongs to.

RPM stands for the RedHat Package Manager, and is the packaging of choice for many distributions. It carries dependency information, and keeps information on what files belong to what package. DEB is the package management system used by Debian distributions. It is much like RedHat, except that it takes care of dependencies more effectively. Use of either one of these makes upgrades easier, since files can be designated as configuration files to be left alone. Old libraries can then be removed, rather than the tarball method, where there would be multiple copies.

Kernel Versioning

The Linux kernel is constantly under development. To keep track of versions, a simple system has been put in place. Versions consist of three integers, separated by periods, i.e., 2.4.2. The first digit is the major number, while the second is the minor. The last is the patch level. Thus, 2.4.0 is the initial release of the 2.4 kernel, with the next upgrade being 2.4.1. The second digit is also important in that it determines whether or not the kernel is a development or stable kernel. If it is odd, it is in development. Both the stable and development kernels will have releases at various intervals. Once 2.4 (as of the time of this writing, it is the current stable stream) goes into a feature freeze, it will be branched out into 2.5. Though minor development (bug fixes, minor updates to drivers) will continue, major reconstruction will be going on in 2.5. Once 2.5 is at a release state, it will be released as 2.6.0 and the process will continue.

Take Your Exam for Less!



Discount Exam Vouchers from PrepLogic
Why pay retail price for the exam when you can
save up to **40%** with discount exam vouchers?

Buy Your Voucher Now

PrepLogic
Be Prepared. Be Confident. Get Certified.

Get this **Study Guide** and many more for **FREE** at

CramSession
www.cramsession.com

Benefits of Using Linux

Linux is a powerful, modular operating system that is free of the licensing restrictions that burdens many of its competitors. It can support multiple users, works on standard or enterprise hardware, and has a wide variety of software available, from desktop productivity software to Internet servers. The open nature of all the software ensures that bugs can be found and fixed by anyone with the right skills, rather than being constrained by a vendor's schedule.

Installation

Each distribution of Linux has a different installation procedure, though there are many commonalities.

Source Location

One critical item to determine is where you plan on storing the distribution itself. If only one computer is to be installed, it may be easiest to store it on a CD and install from there. However, if several computers are involved, a network-based installation may be preferred.

The first method is called a local installation, which usually employs a CD, though the image may already be on the hard drive (usually on a DOS partition). The second, network based installation, has four possible sources; namely, HTTP, FTP, SMB, and NFS (note that not all distributions support all of these methods). Your choice will depend on the server that is storing the image. SMB would be used if a Windows machine hosts the image, NFS if you are already in a UNIX shop, and FTP/HTTP if you decide to install from a remote site (i.e., your local distribution mirror).

Boot Disk

Common to all methods is the need for a boot disk (though the CD itself may be bootable). This disk boots into a stripped down version of Linux that is used to guide the user through the installation. If you cannot boot off the CD, or are doing a hard drive or network install, you will have to locate the images directory on the source. As all distributions are different, this may be difficult. With RedHat, this is in the root directory of the CD in a directory called images. There are multiple files, but `boot.img` and `bootnet.img` are the main ones. As their name may imply, `boot.img` is used for local installations, while `bootnet.img` allows for network-based installations. Once you have the appropriate image, you can copy it to a floppy disk from DOS with the `rawrite.exe` utility (almost always provided on the CD). You will need to know the name of the image, and the device you are copying to (usually `a:`). If you already have a UNIX machine handy, you can use the `dd` command.

```
dd if=imagename of=/dev/fd0
```

will copy `imagename` to the first floppy drive (`a:`). Once this floppy is created, you can boot off of it to begin the installation.

Installation Options

One important thing to know before starting a Linux installation is your hardware. Know your model of video card, how much hard drive space to allocate to Linux, any ISA devices and their IRQ/DMA/port settings, and what port your mouse and/or modem are on.

Workstation/Server/Custom

Many distributions allow you to speed the installation by defining a role for your machine, such as Workstation or Server. Choosing one of these preselects the packages to install, and may influence the partitioning of your drives. For beginners, these selections save time, but, as you gain experience, you may want to choose a custom installation.

Partitioning

While Windows tends to rely on drive letters (C:, D:, etc.), Unix has a single filesystem concept. To the system, multiple drives combine into one by mounting (or attaching) to a particular point on the filesystem. This practice allows for easier expansion, since applications do not have to be reconfigured, and, in the event of corruption, limits the extent of damage.

At the very minimum, you will need two partitions for your system -- root (/) and swap. For performance reasons, Linux likes to have swap on its own partition. In practice, you will have multiple partitions:

Name	Min Size	Usage
Swap	128M	Virtual RAM -- stores inactive memory to disk until it is later used
/	250M	Root filesystem, includes basic libraries, programs, and configuration
/var	250M	Logs, spool files, lock files. For files that change frequently, hence the name
/usr	500M+	Most applications go here
/boot	16M	Kernels get stored here, used to overcome BIOS limitations during the boot sequence
/home	500M+	Home directories of users, including user specific configuration and data

The size of the swap file usually varies between a factor of 1-2 times the amount of physical memory. There are situations (databases, mainly), where increased swap is desired, but the 1-2 times, or a round number like 128M, is good.

Partitions like /usr and /home tend to fill up quickly, so if you have extra space it should be put there. Depending on the function of a server, extra space could go to other partitions. For example, a mail server might have little need for a /home partition, but would want lots of space in /var to store all the mail. Likewise, a file server would not have many binaries in /bin, but its /home might be heavily used.

Filesystems

The flexibility of UNIX allows a system to have different types of filesystems on the same computer, each mounted on its own partition. For example, a fault tolerant filesystem may be appropriate for /home, but a faster one, with less overhead, may be better for /usr.

ext2 is the standard filesystem for Linux. It offers good performance, and is stable, but does not take well to abrupt shutdowns. **ReiserFS** is a recent addition to the kernel tree, and is called a **journaling filesystem**. Each write to the drive is written to a logfile, so if the system is shutdown uncleanly, all the transactions can simply be replayed instead of having to reconstruct missing information, as in ext2. Performance is surprisingly good, as it makes excellent use of space on a filesystem with small files.

Security Options

During the installation, you will be prompted for many security settings. The first is likely a choice of authentication methods. **Shadowed passwords** are an option, and should always be used. Since UNIX stores the password as a hash (one way encrypted), an attacker could try to encrypt common passwords to see if any match up to the hashes. Shadowing the password file makes these hashes much harder to obtain. Using **MD5** means that the hashing algorithm is much stronger, though this may cause compatibility issues when connecting to other Unix systems.

In a networked environment, a central password server makes sense, much like NT's domain structure. The traditional way of doing this is via **NIS**, the Network Information Service. If you have a NIS server you can put it in at this time and you will be able to share the password database. Other methods of authentication include LDAP (Lightweight Directory Access Protocol) and Kerberos (a system created at MIT employing strong cryptography). Your network administrator will know which method is in use at your site. When in doubt, select the local option, and reconfigure later.

You will also be prompted for a root password. The **root** account is the administrator of the system: it can do anything. You will use this account to create other accounts, clean up filesystems, and perform other maintenance tasks. It is important that this password be kept secure.

Most modern distributions will now give you the opportunity to create a user account. User accounts, having less privileges, should be used whenever possible. The root account should be used only when needed.

X-Windows

If you have selected to install the X-Windows system, you will be required to supply information about your computer and video equipment. Having this information handy before you begin will make the installation easier. You will need to know the model of video card, amount of RAM on it, and your monitor's horizontal and vertical refresh capabilities.

You may also be given a choice, depending on the distribution, between KDE and GNOME. Both are functionally similar, so the choice is mostly aesthetic. It is wise to learn the basic usage of both of them.

Boot Loader

The Linux Loader, or LILO for short, is used to boot the Linux operating system. During the install phase, you will be asked for some details about how LILO will be configured. First, you will have to decide if LILO is to be placed in the master boot record (MBR), or the first sector of the boot partition. Usually, the first option should be selected, unless you have extra software used to manage large hard drives, in which the second option should be used.

One of the next things you will have to answer is if you wish to boot any other operating systems. LILO, the flexible software that it is, can boot Windows operating systems on your behalf. If you wish to do this, then specify the partition that the other operating system is on.

Another option you will have to choose is if you wish to pass anything to the kernel. Your distribution will likely make suggestions. Otherwise, this can be left blank.

More configuration options are presented in the next chapter.

Kernel

By default, the kernel will come with most drivers compiled as modules, meaning they take up no resources until they are loaded in. If you have the need to add a driver that isn't built, or if your vendor supplies the driver in source form, you will likely need to rebuild your kernel. Another reason you may need to do this is when you need a later version of Linux than is supplied with your distribution (i.e., for added functionality).

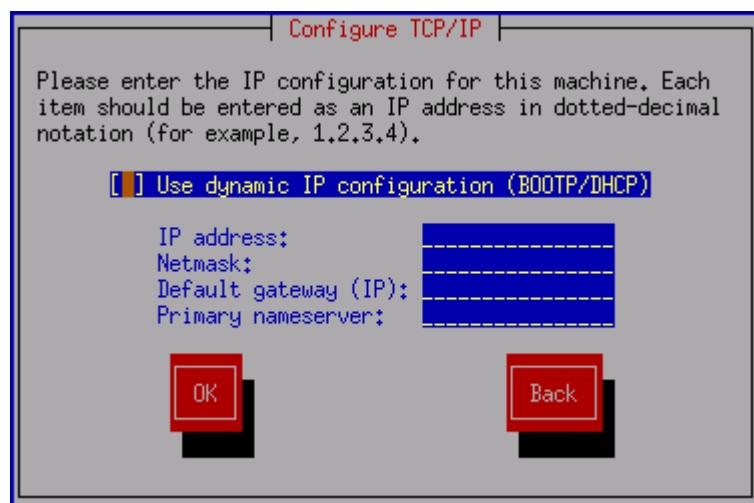
Configuration

Network Settings

LAN

Before configuring the network settings for a computer, ensure you have the proper information handy, such as the IP address, gateway, and DNS servers, unless you are using DHCP.

The network information is most easily configured from `netconfig` or `linuxconf`. `Netconfig` presents a simple screen as shown below.



`linuxconf` allows you to enter the same information, along with more specific details such as extra routes, under the `Config->Networking->Client tasks` menu. You will likely use `netconfig` to get a machine started, and use `linuxconf` to make any minor changes later.

Dial

Dialup connections are configured from `rp3-config`, an X11 wizard that will walk you through the setup. It will discover your modem, or prompt you for the location if it can't find it. It will also need to know the phone number, and user account information of your ISP. From `rp3-config`, you can choose the **dial** option to initiate the connection. For more information on the `rp3` system, and an alternate (`Kppp`), see

<http://www.redhat.com/support/manuals/RHL-7.1-Manual/getting-started-guide/ch-ppp.html>



Who Do You Trust for Your Certification Training?

PreLogic's dependable training products help thousands of professionals and students worldwide achieve their certification goals for A+, MCSE, Network+, CCNA, CEH, PMP, and more.

PreLogic Comprehensive Training Tools:

- CBT • Practice Exams • Audio Training
- Mega Guides • Discount Exam Vouchers

For Free Product Demos,
[Click Here.](#)

PreLogic

Be Prepared. Be Confident. Get Certified.

X-Windows

There are two main methods to configure X Windows. The first is with the `XF86Setup` command, which is a text-based program that asks you to select your hardware from several lists. It is not very user-friendly, which is why `Xconfigurator` was developed. `Xconfigurator` will attempt to auto-detect your hardware, and integrates testing into the procedure (`XF86Setup` will create the configuration for you, but if a setting is wrong you have to go through the whole program again). Typing in either of the above commands, as root, will start the process. For either method, it is helpful to have your monitor manual and video card manual handy, since you may have to look up frequencies and different options.

There is a third, but very difficult, method of configuring X, involves editing the `XF86Config` file in either `/etc` or `/etc/X11`. The settings may be tweaked by this method after one of the automated processes have run, but to generate a new `XF86Config` file would be time consuming.

Internet Services

Inetd

Most Internet services are run from a process called `inetd`. `inetd` listens on behalf of a service, and, upon a connection, it spawns the service and passes over control. This is done to reduce the resources required to have all the daemons stay active and listen themselves.

`inetd` is controlled out of `/etc/inetd.conf`. A typical line looks like

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
```

The key elements here are the first field, which defines the port, number, in this case FTP (21). A list of the service-number mappings can be found in `/etc/services`. Column five dictates the user that the service will run under. Columns six and on are used to tell `inetd` how to start up the daemon. Column six is the daemon to run, and seven and on are the arguments. In this example, `/usr/sbin/tcpd`, the TCP wrappers, are being run, and are passed `in.ftpd -l -a`. `tcpd` then uses the arguments to run the program after performing security checks. If `in.ftpd` was to be run directly, column six would be `/usr/sbin/in.ftpd`.

To disable a service, comment out its line by putting a hash symbol (`#`) in front of the line. After restarting `inetd` (`killall -HUP inetd`), the service will no longer be active.

Most services run out of `inetd` are run through the TCP Wrappers (`/usr/sbin/tcpd`) in order to provide restrictions based on IP address. `/etc/hosts.allow` and `/etc/hosts.deny` control this. The form of a line in those files is

```
service: address1, address2, etc.
```

where `service` is the name of the daemon (i.e., `in.ftpd` for the FTP daemon configured above), or `ALL` for everything. A secure system would have `ALL:ALL` in `hosts.deny`, and would then permit access on a fine-grained level in `hosts.allow`. To permit FTP from anywhere, and SSH from the 10.0.0.0 network, `hosts.allow` would have

```
in.ftpd: ALL
```

```
sshd: 10.
```

A more detailed examination of these files is in the `hosts_access(5)` man page.

FTP

FTP is a service that is usually run out of `inetd`. Most distributions include the `WUFTP` daemon, which is configured in `/etc/ftpaccess`. The best reference for this is the `ftpaccess(5)` man page (`man ftpaccess`). Alternatively, `Linuxconf` can be used to configure FTP via `Config->Networking->Server Tasks->Ftp server`.

Web Server

HTTP can be run out of `inetd`, or run as a standalone daemon. The latter is the preferred method, because of the bursty nature of HTTP traffic. Apache is the most popular web server. The configuration file is `httpd.conf`, but the location depends on your distribution (`locate httpd.conf` will find it).

<http://httpd.apache.org/docs/mod/directives.html> has a complete list of all the available commands, and the default `httpd.conf` is well documented.

Those familiar with HTML will find apache easy to configure. Directives are simply statements, such as

```
ServerRoot "/usr/local/apache"
```

Options can be made to apply only to directories with the `directory` tag:

```
<directory /usr/local/apache>
    Options FollowSymLinks
</directory>
```

NFS

NFS is the Network File System, which allows machines to share directories. The `/etc/exports` file controls who can access which filesystems.

<code>/export</code>	<code>machine1(rw)</code>
<code>/public</code>	<code>(ro)</code>

The example above shares the `/export` tree with `machine1`, in a read and write fashion. `/public` has no restrictions on who can connect, but it is read only. Run `man exports` for more information on all the options available in this file. You will have to restart the `mountd` process in order for any changes to become visible. The `exports(5)` man page has a list of all the options available, along with examples.

POP

The Post Office Protocol, or POP, is used to let remote users retrieve messages from their mailbox. It runs out of `inetd`, and should require no configuration. In `Linuxconf`, accounts can be made POP-only from `Config->Users Accounts->POP Accounts`.

Rlogin

Rlogin services are, for the most part, replaced by `ssh` due to the lack of security in the Rlogin protocol. If the service is enabled in `inetd.conf`, then passwordless authentication can be configured by the creation of a `.rhosts` file in the user's home directory, or a global wide `/etc/hosts.equiv`. The general form of this file is

```
+host user
```

which would let the specified user, from the specified host, log in to the local user account with no password. As you can see, this is easily spoofable, and is not recommended.

SMB

Samba, the Windows file sharing package, consists of two daemons. "smbd" takes care of the file sharing itself, and "nmbd" takes care of name resolution. They are both configured through the same file, smb.conf. Configuration of Samba is best done through the SWAT (Samba Web Administration Tool) interface, which runs on port 901 (<http://localhost:901/>). However, a basic knowledge of the configuration directives is necessary.

Besides the complete list of configuration directives at <http://us2.samba.org/samba/docs/man/smb.conf.5.html> the HOWTO is a good reference:

<http://us2.samba.org/samba/docs/Samba-HOWTO-Collection.html>

The simplest smb.conf demonstrates the form of the file:

```
[global]
workgroup = MYGROUP
[homes]
guest ok = no
read only = no
```

The format is much like a windows .ini file: there are several headings encased in square brackets. [global] is the main section, it sets the main options for the program (in this case, the workgroup is MYGROUP). Shares also have the same format. In this example, the homes share is defined as requiring authenticated access (guest ok = no). Homes is a special share: it maps the user's home directory to \\server\username. A share such as \\mymachine\fred will map to user fred's home directory (assuming you haven't explicitly defined a fred share) according to the rules in [home].

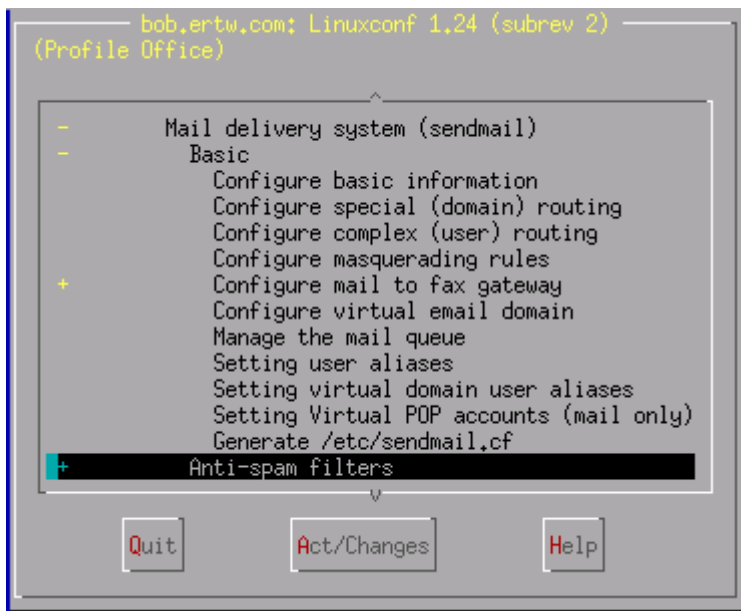
A share can be defined as simply as

```
[tmp]
path=/tmp
guest ok = yes
```

which will create a tmp share pointing to /tmp. The smbd daemon must be restarted in order to make the configuration files take effect.

Sendmail

Configuration of sendmail is extremely complex, and is a job usually left for the senior administrators. However, linuxconf can be used to make common changes via the Config->Networking->Server Tasks->Mail delivery system menu.



As you can see, there are many options to be configured. Important menus are the `Configure basic information` menu, which is used to define your domain name, and what your system is to do with email (deliver, or send on to a gateway). The `Setting user aliases` menu is where you assign extra addresses to users (i.e., abuse@example.com goes to admin@example.com), or redirect mail to other domains.

Telnet

If telnet is to be used, it should be strictly controlled through the TCP Wrappers. Since all communications, including passwords, are sent in the clear, a person sniffing packets could compromise your system. The use of SSH is preferred.

TFTP

TFTP is an unauthenticated version of FTP that runs over UDP. Common usage is in network devices, or diskless workstations, where the machine obtains its kernel from a TFTP server. When implementing TFTP, create a directory on your computer that you consider to be the TFTP root, and don't store any sensitive information in there. Since the TFTP daemon can not break outside this root, it can not access files like password or user files, as it could if the root were the normal filesystem root directory. This special directory is then passed to the daemon in `inetd.conf` as the last argument.

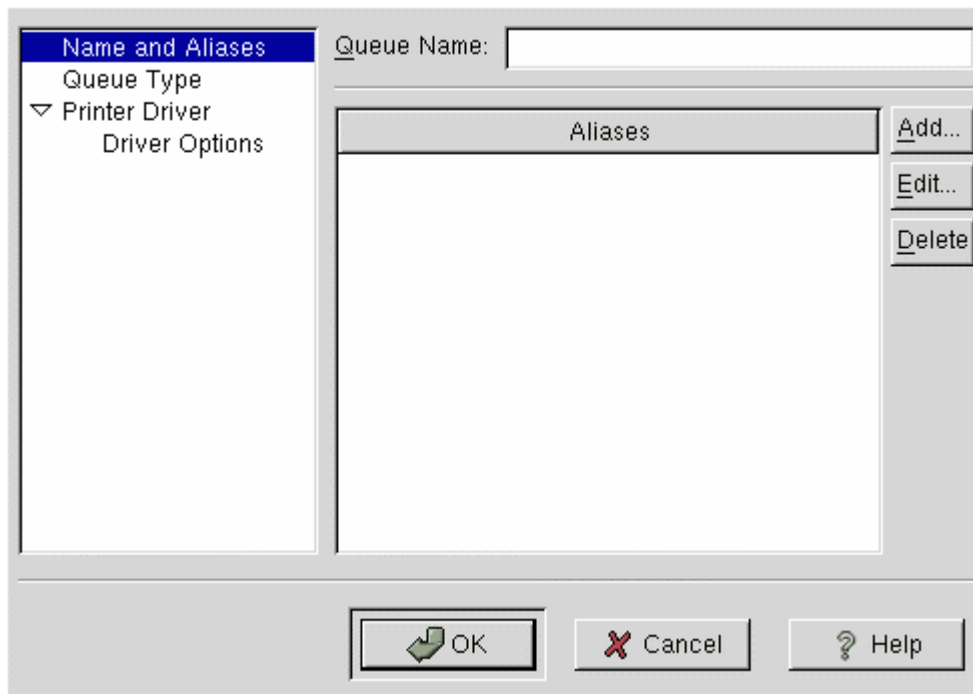
Another special note about TFTP is that, by default, the file must exist before it is written. This is to prevent unauthorized people from creating files. Simply run `touch filename` in the appropriate directory to allow the remote client to write to filename.

Printers and Other Hardware

UNIX has a spooling system, much like Windows NT, except it is more flexible and formal in that there are multiple processes which handle different aspects of the printing procedure. The `lpd` daemon is responsible for listening to printer requests and spooling them. Jobs are submitted through the `lpr` program.

Configuring a printer is done through the `printtool` program. Clicking on the "new" icon gives a menu as shown below. Each print queue requires a name, and may have several aliases. It is advisable to have one printer with a

name or alias of "lp", since this is the default printer name for the system.



Selecting the "Queue Type" option lets you select where this printer resides, either on a physical port (/dev/lp0 corresponds to LPT1:, lp1 to LPT2:, etc), another UNIX server, a Microsoft NT share, a Novell server, or a print server address. From the "Printer Driver" menu, you can select the model of printer.

This last option is important. Before a job is sent to the printer, it is passed through a filter. This filter can do almost anything, from converting a raw .jpg image to PCL or PostScript, or converting PostScript to the native printer language. To properly do the conversion, the type of printer must be known.

Other Configuration Files

UNIX is a system controlled by text-based files. Each file has a specific purpose, which adds to the ease of administration, but requires the administrator to memorize more information. Most are tab-delimited files, and extended syntax can be obtained through the man system.

/etc/fstab lists the filesystems available to the system:

Device Name	Moun t	Point	Type	Options	Dum p	Pas s
/dev/hda6	/		ext2	defaults	1	1

All columns are self explanatory except for five and six. The dump flag indicates if the filesystem should be backed up, but is rarely consulted. The Pass flag determines the order that the filesystems will be checked. / should be 1, the rest should be 2, except the non disk ones (/proc), which should be 0 to indicate no checking is needed.

/etc/inittab dictates the programs run in various runlevels. The important thing to know from this file is how to change the default system runlevel.

```
id:3:initdefault:
```


The second column in the above line specifies the system runlevel, in this case, 3, which is multi-user, no GUI. 5 is multi-user, GUI.

`/etc/csh.login` and `/etc/csh/cshrc` are invoked on login for all users on csh-like shells. For bash shells, `/etc/profile` is used. Items like global path settings, `umasks`, `ulimits`, and other settings should be set here. These are run before the user-specific versions are executed.

`/etc/motd`, `/etc/issue`, and `/etc/issue.net` are used for giving users information upon login. The Message Of The Day (MOTD) is given after logging in. `issue` and `issue.net` are given before the login prompt on local based terminals, and network based terminals respectively.


`/etc/ld.so.conf` is a list of directories in which the dynamic loader, `ld.so`, will look for shared libraries. After modification, it is essential to run `ldconfig`.

LILLO

The Linux Loader is responsible for booting the operating system on x86 systems. `/etc/lilo.conf` is the file used to configure it. A skeleton file looks like:

<code>boot=/dev/hda</code>	<code># install on the MBR of the first drive</code>
<code>prompt</code>	<code># give the user the opportunity to input</code>
<code>timeout=50</code>	<code># 5 seconds (50 tsec)</code>
<code>default=linux</code>	<code># default is the linux label</code>
<code>image=/boot/vmlinuz</code>	<code># define a kernel -- /boot/vmlinuz</code>
<code>label=linux</code>	<code># tag as linux (see above for default)</code>
<code>read-only</code>	<code># mount as read only (necessary for linux)</code>
<code>root=/dev/hda6</code>	<code># root device (/) is /dev/hda6</code>

Thus, it is possible to add extra images, and use LILO to reboot into different kernel versions. In practice, you will always want to have an older kernel to use in the event that an upgrade fails. View `man lilo.conf` for all the details and options. One common addition is `linear`, which forces a different way of specifying sectors on the drive. This fixes a common problem when LILO cannot boot the system.



Are You Ready to Take the Exam?

Comprehensive Exam Preparation:

- Progress tracking
- Detailed answers and explanations
- Packed with quality practice questions
- Customizable learning features

[Try a Demo Now](#)

A+, MCSE, CCNA, CEH,
CISSP, PMP, and more.

PrepLogic
Be Prepared. Be Confident. Get Certified

Get this **Study Guide** and many more for **FREE** at

CramSession
www.cramsession.com

Modules

Modules are akin to device drivers, in that they enable functionality for a specific device or feature. Almost anything in the kernel can be made into a module, which reduces the size of the kernel and conserves resources until they are needed. It also does away with the need to recompile the kernel each time a new device is added, because modules can be kept handy without taking up memory. They are stored under `/lib/modules/kernel-version`; i.e., `/lib/modules/2.4.4`

The `lsmod` command takes care of listing the currently resident modules

#	lsmod		
Module		Size	Used by
pppoe		6416	2
pppox		1328	1 [pppoe]
Ppp_generic		15936	3 [pppoe pppox]

Column one shows the module name. Above, the modules are related to the PPP service (Point to Point Protocol, for remote access). Column two shows the amount of memory the module is taking up. Column three shows a reference counter, which lets you know if the module is being used or not. A value of 0 means it is not currently being used. The fourth column is a list of the referring modules, or those that depend on the current module. For example, the `pppoe` module relies on the `pppox` module as shown above. They both require the `ppp_generic` module.

The dependency tree is built at boot time by running `depmod -a`.

To add a module, you use either of the `insmod` or `modprobe` commands. `Modprobe` is better at handling module dependencies, but `insmod` is better at forcing modules to load if something goes wrong with `modprobe`. The syntax for them is similar

```
insmod pppox
```

```
modprobe pppox
```

`insmod` can also take the `-f` flag, which means "force", in the event that the version is incorrect or something else goes wrong. `Insmod` can also accept a full path name in the following fashion:

```
insmod /lib/modules/2.4.2/kernel/drivers/net/pppox.o
```

Note that when the full pathname is specified, the `.o` extension is required. Otherwise, the name is looked up in the dependency tree.

Administration

Users and Groups

As a multi-user operating system, Unix uses a concept of users and groups to assign permissions. A user must have a primary group, and may be assigned to multiple secondary groups. The user called `root` is a special account, as it has virtually unlimited privileges and is used for the administration of the system.

Users are defined in the `/etc/passwd` file, one user per line. A typical line looks like

```
dave:x:1010:100:David Smith,,,:/home/dave:/bin/bash
```

A colon (:) delimits fields within the line. The first field is the user name, in this case, "dave". The second field is for the password, in this case 'x' means the password has been shadowed (see section 2). Field three is the userid, referred to as the UID. In Linux, this is a 16 bit number, allowing a maximum of 65,536 users. It is this number that the system uses internally to represent a user. Field four is the primary groupid of the user, defined in /etc/group, and will be explained shortly. Next is a description of the user, often called the GECOS (gee-koss). Some sites choose to add information like phone numbers in this field, separated by commas. Since this field is not important to the operation of the machine, almost anything can go in here.

The final two entries are very important to the security of the user and her environment when logging in. The second last field is the user's home directory, which is where many user specific configuration and files will be stored. The last field is the shell, which is the command line interface the user will see. Common values are /bin/bash (the Bourne shell), /bin/tcsh (a variant of the C shell), and /bin/false. The latter is used when you're setting up a user that cannot log in interactively to the system, such as a mail only user, or a service account.

Groups are defined in /etc/group, and follow a similar scheme:

```
sys:x:3:root,bin,adm
users:x:100:
```

Field one is the name of the group. Field two is for a group password, but this is legacy and not used any more. The groupid, or GID, follows. Optionally, users can be placed into this group as a secondary by adding the user names to the end, separated by commas. As above, the sys group (GID 3) has root, bin, and adm as members. GID 100, users, has no secondary members. From above, though, you can see that David Smith's primary group is 100.

Adding and deleting users is usually done via the `useradd` and `userdel` programs, respectively. The usual form of the command to add a user is:

```
useradd -c "GECOS" -s shell -g GID username
```

from this, the UID will be automatically assigned. If no GID is specified, either "users" will be assumed, or a new group will be created matching the username, and the user placed into it (most distributions now do the latter). This process also creates the home directory, by default.

The user can be deleted by

```
userdel -r username
```

The `-r` option deletes the user's home directory. It is recommended that the user's files be archived if you choose this option.

The `groupadd` and `groupdel` commands exist to add and delete groups, and their usage is similar.

A password must be assigned after the user is created, or the user will not be able to log in.

```
passwd username
```

This command will prompt for a password and confirmation. It can be used at any time to change another user's password, or for the user to change her password by leaving out the username option.

The Root User

The Root user is a special user, since it has permission to do anything on the system. It is always identified by its UID of zero. It is advisable to only use root when needed, thus you should never log in directly as root. The "su" command will allow you to switch users from your current user to root:

```
$ su -
Password: mypassword
#
```

Normal restrictions that prevent users from hurting other users, such as ownership and permissions, do not apply to the root user. Thus, make sure you think about every command before you hit enter!

Files

Most everything in Linux is represented as a file. To control access to files, a permission system is used. A file is owned by a user, belongs to a group, and has a set of attributes that dictate who can write to it. There are three main permission attributes: read (r), write (w), and execute (x). These are then applied, to the user (u), group (g), and others (o).

```
$ ls -l example
```

-rw-r--	1 sean	users	0 May 28 19:15 example
---------	--------	-------	------------------------

The example file above is owned by user sean, from the users group. The permissions are user read/write, group and other read. This can be written as u=rw,g=r,o=r.

For simplicity, the permission attributes are given octal values of 4, 2, and 1 respectively, and are written in the order of user, group, and other. Thus, the previous example would have permissions of 644. A table illustrates.

	<i>U</i>	<i>G</i>	<i>O</i>
r	4	4	4
w	2		
x			
	6	4	4

Permissions are changed via the chmod (change mode) command.

```
chmod permission filename
```

Permissions can either be in octal form (600, 640, etc.), or in long hand (u=rw,g=r). The former form is preferred, as the latter does not explicitly set the mode. Any set of people (u,g,o) that are not specified are not changed.

Permissions can be added or removed by using + and - respectively.

Sometimes you will see the permissions listed with four digits. In this case, the first digit specifies special properties:

Octal Value	Symbol	Description
1	T	On directories only the owner of the file can delete the file. AKA the "sticky bit"
2	s (group)	On files with +x, the program can inherit the group of the owner (instead of the executor). On directories, new files are created with the group of the directory rather than the owner. AKA the SETGID bit
4	s (user)	On files with +x, the program can inherit the uid of the owner. AKA the SETUID bit

The owner and group of a file can be changed by chown (change owner) and chgrp (change group). For security reasons, only root can change the owner. Furthermore, a user must belong to the group that he is changing the file to.

```
chown fred myfile
chgrp users myfile
```

The Filesystem

Locations

The UNIX filesystem is much like a tree. The root directory is called /, or simply the root. Basic system libraries and binaries go under /lib and /bin respectively. A special directory, called /sbin, is for binaries that only the system administrator will generally use. Traditionally, sbin stood for statically linked binaries, but this is no longer necessarily true.

A similar structure exists under /usr and /usr/local, with the former being for general purpose applications and software that will be common across multiple machines. The latter, /usr/local tree, is often a special use directory for locally installed software.

Other directories of note are /etc, which stores configuration files, and /dev which contains all the device files.

Directories and Files

Basic commands are needed to work with directories and files.

A directory listing is obtained with the ls command:

```
$ ls
a.txt b.txt c.doc
```

You may want to use the -l flag to specify that file sizes are to be shown.

```
$ ls -l
total 3
```

-rw-r--r--	1	sean	users		2 Jun 30	13:39	a.txt
-rw-r--r--	1	sean	users		2 Jun 30	13:39	b.txt
-rw-r--r--	1	sean	users		2 Jun 30	13:39	c.doc

A file can be copied with the cp command:

```
cp myfile /tmp
```

will copy "myfile" to the /tmp directory. If /tmp didn't exist as a directory, then it would be created as a file. A similar command exists to move and rename files, called mv. Its use is the same as cp. Be careful not to use it as the DOS ren command when working with multiple files:

```
mv *.doc *.txt
```

will not have the effect intended! When a * appears in a shell command, the shell **expands** it to match any files. Thus, in a directory with a.doc, b.doc, and c.txt, the command would expand to

```
mv a.doc b.doc c.txt
```

Luckily, when specifying multiple files, the last argument must be a directory, so this will result in an error.

Directories are created and deleted via the mkdir and rmdir commands respectively.

Files are deleted with `rm`. The `-r` flag to `rm` means to recurse into subdirectories. `-f` will suppress some warning messages. Thus, to delete the contents of `/usr/local/test` (including the `test` directory itself), run:

```
rm -rf /usr/local/test
```

You can find your current directory with `"pwd"` ("present working directory").

Symbolic links are a way of having multiple names for the same file, without having two copies on disk. There are two types, hard and soft. A hard symbolic link must reside on the same filesystem as the original file, because the filesystem considers the files the same. A soft link is used more often, because it can cross filesystems and refer to directories.

```
ln -s /usr/bin/telnet /usr/local/bin/telnet
```

creates a link, called `telnet`, in `/usr/local/bin`, which points to `/usr/bin/telnet` (note the order: source file then destination file). Omitting the `-s` would create a hard link.

Managing Services

At any given time, the system is in a certain runlevel:

Level	Description
0	Shutdown
1	Single User Mode
3	Multi User Mode, Text
5	Multi User Mode, Graphical
6	Reboot

Runlevels 2 and 4, while not generally used, are still technically valid. The importance of the runlevel is that it determines what daemons to start and stop. For example, in single user mode, very little would be running, but in runlevel 5, X-Windows and all the internet daemons would be running.

`/etc/rc.d/init.d` contains programs that start and stop various services. Each file is a shell script that accepts, at minimum, either `start` or `stop` as a command line parameter. `/etc/rc.d/rcX.d`, where `X` is a runlevel, contains symbolic links to each file in `init.d`. The links are named either `SXXservice` or `KXXservice`, where `XX` is a number denoting priority. Upon entering a runlevel, all the `K` files are run in order with the `"stop"` parameter, then the `S` files with the `start` parameter. Thus, if `/etc/rc.d/rc3.d` contained

```
K20nfs K30web S10oracle S50scanner
```

upon entry, `nfs` and `web` would be stopped (in that order), and `oracle` and `scanner` would be started (in that order). These symbolic links can be managed by hand, or by various scripts that come with each distribution (`service`, `ntsysv`, etc.)

Changing runlevels is handled by the `init` command. To change into runlevel



The PrepLogic Mega Guide

PrepLogic took the CramSession Study Guide and made it better!

- Over 100 pages
- More in-depth content
- Expanded resources
- Includes review practice questions

Get \$10 Off
Get it Now

Coupon Code: **MEGA10**

A+, MCSE, CCNA, CEH,
CISSP, PMP, and more.

PrepLogic
Be Prepared. Be Certified. Get Certified.

5, simply type

```
# init 5
```

The `init.d` directory is also helpful if you want to restart daemons during normal operation. For example, to restart the web service, you can run

```
# /etc/rc.d/init.d/web stop
```

```
# /etc/rc.d/init.d/web start
```

Print Queues

Command	Action
<code>lpq</code>	Show all the jobs in the queue
<code>lprm jobid</code>	Remove jobid from the queue
<code>lpc</code>	Control queue, i.e., start and restart spooler

The commands above default to the queue defined by the `PRINTER` environment variable, or the `lp` printer. Adding `-P` printer will choose the printer.

Connecting to Other Machines

There are various ways to connect to other computers.

Telnet is the most common, but suffers from lack of security, as passwords are sent plaintext.

```
$ telnet othermachine
```

SSH, the secure shell, is much better, but it may not be installed. Its usage is similar

```
$ ssh othermachine
```

While on another machine, you can have X-Windows sessions forwarded to your screen. If you are on the console of `mymachine`, but telnetted to `othermachine`, you can set the `DISPLAY` variable on `othermachine`.

```
othermachine$ export DISPLAY=mymachine:0.0
```

Any X sessions will go to your console. Depending on how secure your distribution installs X-Windows, you may need to allow `othermachine` to connect to your X-Server

```
mymachine$ xhost +othermachine
```

If you are logged in via `ssh`, none of this necessary, as it will forward X automatically over the encrypted tunnel (another reason to use SSH). SSH also allows you to set up password-less authentication in a secure manner.

Shell Scripting

Shell scripting refers to creating programs that are interpreted by the shell, much like a DOS batch file. Each shell script starts off with a line denoting the shell:

```
#!/bin/bash
```

This will force the shell to be `/bin/bash`, even if the user is using a different shell. Remember that the script file must be marked as executable for it to run directly from the command line.

Within a shell script, any commands that you normally type are OK; i.e.,

```
#!/bin/sh
# rewind the tape, tar up the home directory
mt -f /dev/nst0 rewind
tar -czf /dev/nst0 /home
```

You can also assign and read environment variables

```
echo You are using printer $PRINTER
```

Or test variables:

```
if [ "$PRINTER" == "" ]; then
echo \$PRINTER is not set, using LP
PRINTER=lp
else
echo You are using $PRINTER
fi
```

Note that when setting a variable, omit the \$.

Within man pages, you will see that programs return result codes. This number is available in the special \$? variable.

```
grep -q sean /etc/passwd
if [ $? -eq 0 ]; then
echo Sean has an account
else
echo Sean does not have an account
fi
```

Note in the example above, -eq was used to compare, where in the PRINTER example, == was used. The former is used to compare integer expressions, the latter is for string expressions. Sometimes the if statement will be written as

```
if [ "x$PRINTER" == "x" ]; then
```

so that even if \$PRINTER is an integer, the expression is still a string.

A variable can also be set to the output of a program by using **backticks**:

```
USER=`grep sean /etc/passwd`
```

Shell scripting is a large topic, so further reading will be required:

<http://www.osconfig.com/unixshell1.html>

System Maintenance

Storage

Creating Filesystems

Each physical disk (hda, sda, etc), can be broken down into slices known as **partitions**. The fdisk command is used to manage these disk partitions. Fdisk is invoked by passing the name of the physical disk:

```
# fdisk /dev/hda
```

Once inside the fdisk program, the following commands are used

Command	Action
P	Prints the current partition table
N	Creates a new partition. You will need to specify the start and end cylinders on the disk
w	Writes your work to disk. This will remove any existing partitions
t	Changes the partition type. 82 is for Swap space, 83 for regular filesystems. You can pull up a complete list with the 'L' key
d	Deletes a partition

Once the partition is created, you have to format it:

```
# mkfs -t ext2 /dev/hda1
```

will create an ext2 filesystem on the first partition of hda1.

Fixing Filesystems

Filesystems can be fragile. If power is lost to a machine, data may not be properly flushed to disk, resulting in a filesystem inconsistency. **Fsck** (filesystem check) is used to attempt a repair of the disk. On bootup, fsck will be run, but if errors occur, you will have to perform the task manually.

```
# fsck /dev/hda1
```

will run fsck on hda1. You will be required to confirm that you would like fsck to fix the filesystem when an error is found. Running fsck on an active filesystem is not a good idea, since data can be written to it. It is advisable to unmount the disk before fsck'ing.

Mounting

A filesystem on its own is of no use, so it must be mounted on a mount point; i.e., a directory.

```
# mount /dev/hda5 /usr
```

will mount /dev/hda5 on the usr directory. Unmounting is similar:

```
# umount /usr
```

```
# umount /dev/hda5
```

Either the directory or device can be specified for an unmount. Thus, the two commands above are equivalent.

/etc/fstab is a file that contains the device to directory mappings, and was described in the Configuration section. If a device is listed in this file, then the mount command requires only one of the device or mount point.

Scheduling Tasks

Rather than requiring the administrator to manually run routing jobs, the **cron** facility can be used to schedule commands to be run on a regular basis. The daemon that handles this is **crond**, and its jobs are controlled via **crontab** (by no coincidence, the list of jobs itself is called the crontab).

```
crontab -e - edits the current user's crontab
```

```
crontab -l - lists the current user's crontab
```

`crontab filename` - replaces the current user's crontab with the contents of filename

Additionally, root can use the `-u username` option with the above commands in order to specify another user.

The crontab itself has a special format which allows the date and time to be easily set. Six tab-delimited columns are used:

```
minute hour day month weekday command
```

A * in any of the first five columns means that anything is allowed. Thus,

```
0 0 * * * /usr/local/bin/runbackup
```

will run the `/usr/local/bin/runbackup` each day at midnight. The man page for `crontab(5)` has further examples and shortcuts.

By default, any output of a cron job goes to the owner of the crontab. This behavior can be overridden by specifying

```
MAILTO=fred
```

at the top of the crontab. In this example, any output would go to fred.

Patches

When a program crashes, it usually leaves behind a **core dump**, in the form of a file named `core`. This file contains the memory occupied by the program, and is useful for debugging and determining where the program crashed. As in the example below, core files are usually very large. The `file` command will tell you which program generated the core dump.

```
# ls -l core
```

```
-rw----- 1 sean      sean      14798848  Jun 11   22:30  core
```

```
# file core
```

```
core: ELF 32-bit LSB core file of 'soffice.bin' (signal 6), Intel 80386, version 1,
from 'soffice.bin'
```

A core file is only useful on the computer it was generated on. On another computer, it is likely that the addresses in the core file do not match with the originating computer, and the debugging will be impossible. If a program constantly dumps core, it would be wise to contact the author. He will be able to give you instructions on what to do with the core (usually by asking you to use a debugger to generate a back trace). If you have no desire to debug the core file, you can safely erase it.

Since most software in Linux is distributed in source format, **patches** are one way of upgrading the code. Rather than downloading a new version, the differences (**diffs**) can be applied to the old tree to update it. This is common with the kernel, where the compressed size is in the 20MB range, while a patch is around the 1MB mark.

From the top of the directory, `/usr/src/linux`, you can apply the patch via

```
patch -p1 < /path/to/patch
```

Usually patches are compressed (`patch.gz`), so you will want to uncompress them first (`gunzip patch.gz`).

Processes

Each task that the kernel is working on is assigned a process id (PID). Each process has a parent process (PPID). The parent of all processes is init (PID=1). Init is responsible for creating and managing processes.

The ps command is used to list processes. The systems administrator will likely use the -ef flags, which specify all processes, with extended information:

```
# ps -ef
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	Jun16	?	00:00:04	init [5]
root	2	1	0	Jun16	?	00:00:00	[keventd]
root	3	1	0	Jun16	?	00:00:00	[kapm-idled]
...							
Root	478	1	0	Jun16	?	00:00:00	syslogd -m 0

The columns, in order, are the owner of the process, the process ID, the parent process ID, an internally used value, the start time of the process, the controlling terminal (? means that there is no terminal associated with the process), the CPU time in seconds, and the command. The latter can be changed by the process itself (i.e., sendmail uses this space to display its current status). Init (PID=1), and square brackets indicate kernel tasks that can not be killed.

To kill a process, the kill command is used. Killing a process actually sends a signal, which the program can trap and handle in its own way.

```
kill 478
```

will kill PID 478, the syslog daemon in the example above. Since PIDs are assigned in sequence (rolling over at 32767), syslog will not always be PID 478! Sometimes, a process is hung, and will not respond to the kill, so a stronger kill must be used:

```
kill -9 478
```

This sends signal 9 (SIGKILL) to the process, which forces init to immediately destroy it. A normal kill sends signal 15 (SIGTERM), which gives the process a chance to clean itself up before closing. Another signal you may use are 1 (SIGHUP), which is usually interpreted as a signal to reread the configuration file. Instead of numbers, you can specify the name (without the sig). Thus, kill -HUP 478 is the same as kill -1 478. A full list of signals and their meanings is in signal(7).



Who Do You Trust for Your Certification Training?

PrepLogic's dependable training products help thousands of professionals and students worldwide achieve their certification goals for A+, MCSE, Network+, CCNA, CEH, PMP, and more.

PrepLogic Comprehensive Training Tools:
 CBT • Practice Exams • Audio Training • Mega Guides • Discount Exam Vouchers

For Free Product Demos, [Click Here.](#)

PrepLogic
Be Prepared. Be Confident. Get Certified.

Get this **Study Guide and many more** for **FREE** at

CramSession
 www.cramsession.com

Logs

The syslog daemon is responsible for collecting logs generated by programs, and writing them to different files. While a complete discussion of this program is not needed for this exam, you will need to know the default logfiles, and what goes in them.

Logfile	Description
/var/log/messages	General system messages, such as status reports from various daemons
/var/log/maillog	Messages dealing with mail transfer, such as logs of sent and received messages, and any other sendmail warnings
/var/log/secure	Logs of connection attempts, logins, and security related notifications
/var/log/cron	Status reports from the cron daemon
/var/log/lastlog /var/log/wtmp	These files are used to provide historical data about logins. The "last" command is used to interpret these files. It is vitally important not to delete them, as it will cause problems for users logging in.

Other files in the /var/log/ directory may be as a result of other installed applications. For example, many distributions have the samba daemon log to various files in /var/log/samba/.

Backups

There are two general programs used to make backups. The first is tar (Tape Archive), and the second consists of the dump and restore commands.

TAR

To create an archive, run

```
tar -czf backup.tar.gz /home
```

will create a gzipped tarball called backup.tar.gz, containing the contents of the /home directory. This file will usually be copied to a CD or tape for safe keeping. To restore /home,

```
tar -xzf backup.tar.gz
```

One important thing to keep in mind is that inside the tarball, all filenames will start with home/ (the leading / is always stripped). Thus, when the file is unpacked, a home directory will be created in the current directory. This is because the file specifier in the create command explicitly used the home directory. The other way to do this would have been to run the tar command in /home, and use * instead of /home. You can check what's inside the tarball with

```
tar -tzzf backup.tar.gz
```

Dump/Restore

Dump and restore allow the administrator a bit more flexibility, but are more complex to use. They work on drives/partitions, rather than directories. To perform a full back up /dev/hda1 to tape device /dev/nst0, run:

```
# dump 0usf 1048576 /dev/nst0 /dev/hda1
```

The first parameter given is the dump level, in this case 0. 0 means a full backup. A level one backup refers to all the files changed since the last level 0. Level two will get all the changed files since the level one, and so on. The u flag means to update the /etc/dumpdates file, which helps dump in calculating which files to archive for each dump level. 's' means that you will be passing the tape length (in feet), which helps it know when to prompt for a new tape. This number comes after the flags, which is what the 1048576 refers to (DDS1 tape). 'f' indicates the output file, which is given as /dev/nst0. Finally, after all the flags and associated values are given, goes the device to dump. This will not compress the output: it is expected that the tape drive will do that for you. To compress manually, run:

```
dump 0usf 1048576 - /dev/hda1 | gzip -c > /dev/nst0
```

could be used. The file '-' means the standard output, which is then piped through gzip, and manually sent to /dev/nst0.

You can verify a dump file against the disk with

```
restore -Cf /dev/nst0
```

assuming the tape is positioned at the current dump file.

Some commands helpful to move the tape are:

mt -f /dev/nst0 rewind	# rewind the tape
mt -f /dev/nst0 fsf 1	# fast forward one position
mt -f /dev/nst0 tell	# what position are we at?

To perform a full restore, get into a freshly formatted partition,

```
restore -rf /dev/nst0
```

more often, you will want to only restore a few files at once. Interactive mode is then used:

```
restore -if /dev/nst0
```

The man pages for both dump and restore provide complete information on their use.

Security

Physical

Physical security is very important, since many security settings can be overridden at the console. Ensure that servers are in a controlled environment, behind locked doors, and with limited access. Don't leave root logged in!

Passwords

Passwords should expire after a certain amount of time. This way, if a password is compromised, then the cracker will not be able to hang around your system forever. To see the aging information on an account, use the chage command:

# chage -l testuser	
Minimum:	0
Maximum:	99999
Warning:	7
Inactive:	-1
Last Change:	Jun 22, 2001
Password Expires:	Never
Password Inactive:	Never
Account Expires:	Never

To force a password change every three months (90 days), and give a seven day warning, use

```
# chage -M 90 -W 7 testuser
```

When the password is about to expire, the user will get a warning message upon login:

```
login: testuser
```

```
Password:
```

```
Warning: your password will expire in 1 day
```

System

Maintaining system security can be a lot of work. Mailing lists are a great way to keep on top of current problems with your software, and to let you know of updates. You should be subscribed to your distribution's security mailing list at the least. A web sites of interest is:

<http://www.linuxsecurity.com>

Here's a small checklist to follow after installing, and to periodically verify

- ❖ Check your inetd related services (/etc/inetd.conf, /etc/xinetd.d/), and make sure you know what is running. Turn off anything you are not using.
- ❖ Check the services that are starting in your runlevel (/etc/rcX.d). The chkconfig --list command will be helpful
- ❖ Replace telnet with SSH if at all possible. <http://www.openssh.org>
- ❖ Browse the list of installed packages, and remove items you'll never need (news servers, etc.)
- ❖ Look at the accounts in /etc/passwd to make sure there are no surprises
- ❖ Use the wrappers (/etc/hosts.allow, /etc/hosts.deny) to limit who can connect
- ❖ Keep an eye on logs. Logcheck (<http://www.psionic.com/abacus>) is a handy tool for this
- ❖ Keep up on those patches!
- ❖ Keep the number of setuid and setgid files to a minimum

Troubleshooting

Documentation

Proper documentation of your system is critical. In the event of a problem, you may not be able to rely on the information your system is giving you. During the install phase, a list of the options chosen should be kept in case a re-install is needed. Changes to configuration files (and regular backups) should also be logged. Not only does this help during a recovery, but also when determining the cause of a problem.

Any software that doesn't come from your distribution should be noted in case you want to check for upgrades later, or change options. Keeping all the source files in a directory such as `/usr/local/src` is also helpful.

Tools

A keen grasp of the various UNIX commands will be helpful when troubleshooting a problem. When dealing with logfiles, it is often easier to maintain a constant watch on a particular logfile when trying to reproduce the error.

```
tail -f /var/log/messages
```

will keep a constant watch on the messages log (new messages will get appended within the terminal). If the error occurred in the past, you might be interested in only certain log entries.

```
grep imapd /var/log/messages
```

will return only the lines containing the "imapd" string. If there you find a reference to a file, but can not find it, there are two ways of finding the file:

```
locate foo.pl
```

```
find / -name foo.pl -print
```

are roughly equivalent. Depending on the distribution, the locate command may not be able to see into all directories, so the find version is often preferred.

Sometimes a program will generate too much output on the screen to read. Redirecting this to a file is then necessary.

```
program > logfile # truncate logfile and dump all output to it
```

```
program >> logfile # append output to logfile
```

The above uses suffer from not being able to catch the error stream. Some extra shell commands are required to get both the standard output stream, and the standard error stream

```
program >> logfile 2>&1 # redirect both stderr and stdout to logfile
```

If you would like to see the output on the screen, and save to a file, the tee command is used

```
program | tee logfile
```

The next thing that is essential is to be able to use one of the many UNIX editors. VI is perhaps the best choice, since it is going to be available on almost any UNIX variant, and, in the event that you have to boot up in a minimal mode, it may be the only editor available.

VI uses commands instead of menus. Generally, you are either in edit mode (where you are typing text), or in command mode (where you are giving commands to vi). To switch from edit mode to command mode, hit the ESC key. To get from the default command mode into an edit mode, use one of the following commands (note that everything in VI is case sensitive):

a - start editing after (append) the current character

A - start editing at the end of the current line

i - start editing at the current character (insert)

I - start editing at the beginning of a line

o - create a new line after the current, and start editing

O - insert a new line and start editing

From command mode, you can enter :w to write the file, or :w filename to write to filename. :q quits, and :q! quits without saving. A full VI tutorial can be found at <http://www.epcc.ed.ac.uk/tracs/vi.html>. VI is complicated, but once the basic commands are mastered, editing becomes very quick and easy.

Diagnosis and Repair

A methodical approach must be taken to solving Linux problems. If the report comes from a user, carefully document the symptoms, especially error messages. Unix is very terse, so exact wording is often necessary. Try to separate users' opinions (logging in is slow) from the actual symptoms (it is hanging right before it tells me that I have new mail).

Boot

Boot problems will generally fall into one of two categories. Kernel booting, and startup scripts. Booting of the kernel is handled by LILO, the Linux Loader.

The second class are usually the easiest to solve. As noted before, each service has its own script in /etc/rc.d/init.d (or /etc/init.d in some distributions). Thus, if the failing script is known, it can be tested while the system is still operational. The software section below gives some advice on using logfiles to determine problems. In the case of a daemon not starting up, it is sometimes more helpful to follow the logic of the script in another session so that all errors are noticed. One common problem is with libraries (below). The actual error may not be noticed on bootup since the script might redirect it, but if the startup commands are typed in by hand, you will likely see them and be able to determine what the problem is.

In the case of heavy disk corruption, you may only be able to boot into single user mode and must fsck the disks by hand. If you would like to force a single user mode boot, you can enter

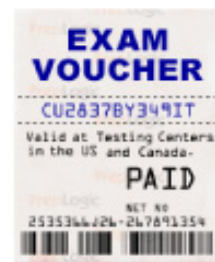
```
linux single
```

at the LILO prompt. Most daemons will not be started, so you have the opportunity to troubleshoot without corrupting data further.

Problems where the operating system itself fails to boot may require you to use a rescue disk. A rescue disk is a kernel plus a root filesystem with utilities (fsck, fdisk, vi, etc.) on it. Essentially, you are booting into a minimal Linux configuration, and must mount your real filesystems manually. Distributions handle this in different ways. With Red Hat, you can use the installation CD and select the "rescue" option. Sites like <http://www.freshmeat.net> also carry distribution neutral versions, which may be handy to keep in your toolkit.

If the case is that the kernel itself is corrupted, you may be able to use the rescue toolkit to boot your own root filesystem. Generally, you can boot with

Take Your Exam for Less!



Discount Exam Vouchers from PrepLogic

Why pay retail price for the exam when you can save up to 40% with discount exam vouchers?

Buy Your Voucher Now

PrepLogic

Be Prepared. Be Confident. Get Certified.


```
root=/dev/hda1
```

passed to LILO in order to boot onto /dev/hda1.

LILO may stop before booting the kernel. Each letter that is printed gives some indication of how far it got. A description is given at

<http://www.linux.cu/pipermail/linux-l/2000-November/006554.html>

the README file for LILO (/usr/doc/lilo*/README) also has some excellent troubleshooting procedures (including an expanded version of the table in the link above).

Resource

Resources, such as memory and CPU, are shared between all processes. A common problem is that a process consumes 100% of the CPU, or starts allocating excessive memory. While these are usually caused by software errors, the administrator must return the system back to the normal state. In these cases, restarting the daemon is required. If the daemon is persistent, a kill -9 might be necessary. The "top" utility is helpful for finding which process is hogging the resources.

Before killing off the process, make a note of the symptoms, such as 100% CPU usage, any recent log entries, number of connections, etc. Restarting the daemon should return the system to normal. If this is a reoccurring problem, then you should check for updates to the software. You are likely not the only person experiencing this error.

Dependencies

Though shared libraries reduce memory and disk usage, they can also cause problems if the correct version is not available.

```
# amflush
```

```
amflush: error while loading shared libraries: libamanda-2.4.2p2.so: cannot load shared
object file: No such file or directory
```

All the library dependencies can be checked with the ldd command. This command is particularly helpful if you need to know what version of library that the binary is looking for.

```
# ldd /usr/sbin/amflush
      libamanda-2.4.2p2.so => not found
      libm.so.6 => /lib/i686/libm.so.6 (0x40025000)
      ...
```

From this example, you can see that amflush is missing libamanda, and uses version 6 of libm (the math library). Missing libraries are usually caused by improper installation, accidental deletion, or a library directory that is not specified in /etc/ld.so.conf (don't forget to run ldconfig after changing this file.)

When using .deb or .rpm binary packages, you will not be able to install the software without filling the required dependencies:

```
# rpm -i openjade-1.3-13.i386.rpm
error: failed dependencies:
      sgml-common >= 0.5 is needed by openjade-1.3-13
```

In this example, the sgml-common package is required before installation of openjade can continue.

Software/Configuration

Most software will generate some sort of logging information. If it is a background daemon, chances are that it logs through syslog, and you'll find what you need in `/var/log/messages`. Some exceptions are apache, which logs to its own file (`/var/log/httpd/error_log` or `/usr/local/apache/logs/error_log`), and samba (`/var/log/samba/` or `/usr/local/samba/logs/`).

If users complain that they are experiencing problems with your internet services, such as not being able to retrieve their mail, or are getting errors on the web server, look at the logs. Here is an example of an imap user giving an incorrect password:

```
poochie imapd[5177]: Login failure user=test host=localhost [127.0.0.1]
```

From this log entry, you can see that the service is `imapd` (the `grep` technique from above is helpful in finding this), and that user "test" had a login failure. The hostname is `localhost` (so this was a local user).

Once the cause of the error has been determined, the fix is usually a configuration change. In the example above, the fix would be to change the user's password (assuming it was forgotten). With programs like apache and samba, fixes tend to be changing file permissions or adjusting the configuration file.

Sometimes errors can only be fixed by restarting the daemon. Like the CPU and memory errors above, checking for updates in the software is the recommended long-term solution. Errors in this category include unresponsive daemons, or a slow service. A complete reboot of the system should never be necessary, unless you have exhausted all other options.

Hardware

Hardware problems will generally result in some kernel messages being printed. The `dmesg` command is used to show the kernel log buffer. Messages in this log dealing with errors will include the device number. The following is an example of the kernel reporting that it is running into some bad sector errors on drive `hda`:

```
hda: read_intr: status=0x59 { DriveReady SeekComplete DataRequest Error }
hda: read_intr: error=0x40 { UncorrectableError }, LBAsect=8166445, sector=214207
end_request: I/O error, dev 03:09 (hda), sector 214207
```

You should be able to reconcile the sector (8166445) with the filesystem that is giving the problem with `fsck`.

File Systems

File systems can become corrupt after unclean shutdowns, so the `umount` and `fsck` commands must be used to return them to a stable state as previously described.

Another problem with disks is capacity. A disk can become full by either storing too much information, or too many files. Due to the nature of the `ext2` filesystem (and many others), a certain amount of **inodes**, or file pointers, are created when the file system is formatted. The `df` command is used to show the current usage of both space and inodes. "`df -k`" shows free space in Kilobytes, and "`df -i`" will show inode usage. Even if `df -k` shows that ample free space exists, the system can still run out of inodes. Normally, this won't happen, but a system with thousands of small files will exhibit this behavior. This is probably as a result of a poorly written program that isn't cleaning up after itself, but could also be a sign that the partition must be reformatted with a higher inode count.

Backups

Backups can fail for many reasons, but the most common will be lack of tape (or disk, depending on how you back up). Either the correct tape was not ready, or the backup is too large to fit on a tape. Depending on the backup software used, you may get an explicit message to the fact, or a disk error.

Another problem with backups is the media. Tapes can go bad, and CD writes can fail. Even if the backup succeeds, it is important to verify the consistency of your backups. Testing can be done through the backup software, or through periodic test restores.

Identify, Install, and Maintain System Hardware

The best thing that you can do to prevent hardware problems is to make sure all your devices are on the distribution's hardware compatibility list. This list is generally available off of the vendor's web site.

Resources

Your best troubleshooting source is the `/proc` directory. `/proc` is a virtual filesystem -- it takes up no hard drive space, each file is a way to get information from the kernel. For example, the `/proc/uptime` file, when read, will return the uptime of the system.

IRQ

An **interrupt request**, otherwise known as an IRQ, is a system that allows devices to request the attention of a CPU, such as the arrival of a packet on an interface.

A list of devices assigned to IRQs can be determined from the `/proc/interrupts` file:

\$ cat /proc/interrupts		
	CPU0	
0:	137450255	XT-PIC timer
1:	848513	XT-PIC keyboard
2:	0	XT-PIC cascade
3:	19674	XT-PIC serial
5:	15607219	XT-PIC ide2, eth0
8:	39034	XT-PIC rtc
9:	62416173	XT-PIC es1371
11:	518706	XT-PIC usb-uhci, usb-uhci, advansys
12:	8195016	XT-PIC PS/2 Mouse
14:	1535447	XT-PIC ide0
NMI:	0	
ERR:	0	

When an interrupt is shared, it will show up with multiple devices, as in the example above beside IRQ 5 and 11.

If you do not want devices to share IRQs, the best method is to adjust the PnP settings in the BIOS. While Linux does support PnP to a limited extent, it is not an easy thing to manage, and it is not always successful.

Ioports are much like IRQs in that they provide a way for devices to share data with the rest of the computer. The list of used io ports are in `/proc/ioports`

```
$ cat /proc/ioports
0000-001f : dma1
0020-003f : pic1
...
cc00-cc7f : 3Com Corporation 3c900B-TPO [Etherlink XL TPO]
    cc00-cc7f : eth0
```

In this example, you can see that the Ethernet card is using the ioports from 0xcc00 to 0xcc7f. If another device is set up to use this, it could cause both devices to fail.

Memory

One symptom of bad memory is random segmentation violations in otherwise stable applications. The Sig 11 FAQ at <http://www.BitWizard.nl/sig11/> is an excellent resource on tracking down the cause of these errors.

CPU/Motherboard

CPU and motherboard problems usually manifest themselves by the computer failing to start. More subtle problems include random segmentation violations, and periodic rebooting. Ensure that your motherboard is clocking the CPU at the correct frequency, and that the voltages are correct.

If the motherboard or CPU fails entirely, a replacement is the only option.

Hard Drives

SCSI

Before installing any SCSI device, you must make sure that the new device's SCSI ID is unique on the bus. SCSI acts as a shared bus: if two devices have the same ID, neither will work properly. Don't forget that the SCSI card itself has an ID, usually 7. As a bus, it requires proper termination at both ends (internal and external). More modern devices implement termination automatically, so this may not be necessary.

Troubleshooting procedures involve testing the above requirements. If the SCSI bus was working before the new device was installed, then either the device is incorrectly configured, or something was dislodged during installation. Testing of a new bus might be difficult in Linux, because the Windows applications that come with the card will not work. Try booting the machine without any devices on the chain, and then slowly add devices.

The `/proc/scsi` directory carries a lot of helpful information on the state of SCSI in your system. `/proc/scsi/scsi` lists all the devices seen on the bus. `/proc/scsi/modulename`, where `modulename` is the module name of your SCSI card, lists valuable diagnostic information.

If your SCSI card is not detected on boot, ensure that `/etc/conf.modules` has an alias line similar to

```
alias scsi_hostadapter advansys
```

If not, Linux will probably not find the device. In this example, the SCSI card uses the `advansys` module. You can find a complete list of available modules in `/lib/module/KERNELVERSION/kernel/drivers/scsi` for 2.4 kernels, or `/lib/module/KERNELVERSION/scsi` for 2.2 and earlier kernels. As discussed before, you can insert the module with the `modprobe` command.

IDE

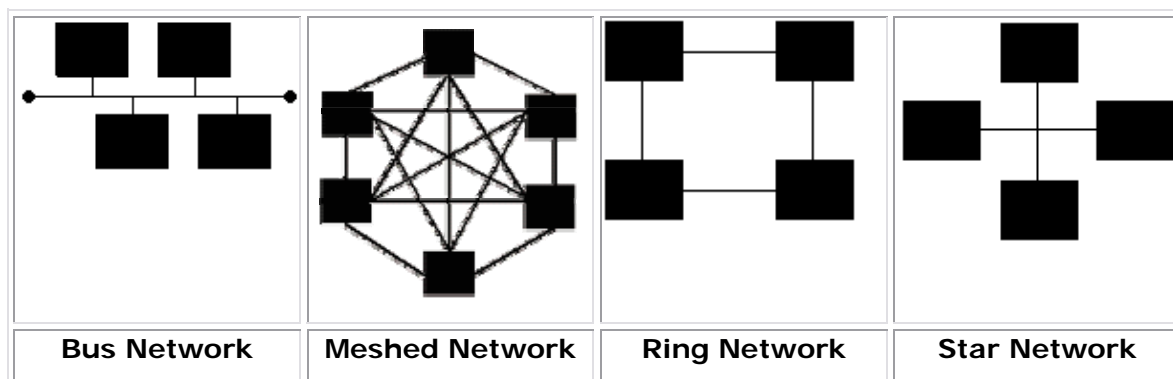
IDE devices are less work to configure than SCSI, but generally do not have the speed or flexibility of SCSI. Each chain has a master and a slave device. Normally, you have two chains, the primary and secondary, but expansion cards can extend that up to ten chains.

The device will have a name of `/dev/hdX`, where X is a letter corresponding to the chain and position. The primary-master will be 'a', the primary slave 'b', and so on. The busses themselves are referred to as `ideN`, with the primary bus `ide0`. `/proc/ide` contains several directories and files depending on the hardware you have installed.


When installing an IDE device, you must know beforehand where it will be, and set the drive to be a master or a slave accordingly. Drives are detected on bootup, so you can watch for them in the initialization sequence. The `dmesg` command shows this again if you missed it, though `/proc/ide` will give more than enough information.

Network

A network is a system that allows for the sharing of resources. There are four main topologies of networks:



Bus - A bus can be modeled as a long chain, each device connecting to at most two neighbors. This is the simplest of all, but a single break in the chain will segment the network. The chance for collisions is greatly increased as the number of devices increases.



The PrepLogic Mega Guide

PrepLogic took the CramSession Study Guide and made it better!

- Over 100 pages
- Expanded resources
- More in-depth content
- Includes review practice questions

Get \$10 Off
Get it Now

Coupon Code: **MEGA10**

A+, MCSE, CCNA, CEH, CISSP, PMP, and more.

PrepLogic
Be Prepared. Be Confident. Get Certified.

Get this **Study Guide** and many more for **FREE** at

CramSession
www.cramsession.com

Star - In the star topology, each device connects to other devices through a central controller, usually a hub or switch. A cable break will only isolate one device, rather than segmenting the network in the case of a bus. Collisions can be the same as in a bus topology, or can be lessened by increasing the intelligence in the central controller.

Ring - A ring is like a bus that closes in on itself. It is ideal for token based protocols, where only the device that holds the token can talk, virtually eliminating collisions. In the case of protocols like FDDI, the ring can survive a break by using two concentric rings.

Mesh - This is quite a complex configuration, since it will require $(N^2-N)/2$ links, where N is the amount of devices. It has better survivability in the case of link failures, which makes it ideal for WAN configurations. The cost and complication of management make it unattractive for the LAN, however.

Network adapters are used to connect to networks. In Linux, you can get a complete listing of active network adapters by running ifconfig:

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:A0:24:D3:C4:FB
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:60965615  errors:16  dropped:0  overruns:16  frame:16
          TX packets:54918449  errors:0  dropped:0  overruns:0  carrier:19
          collisions:1002834  txqueuelen:100
          Interrupt:9  Base address:0xfc40

eth0:1    Link encap:Ethernet  HWaddr 00:A0:24:D3:C4:FB
          inet addr:192.168.1.2  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          Interrupt:9  Base address:0xfc40

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:33148157  errors:0  dropped:0  overruns:0  frame:0
          TX packets:33148157  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0  txqueuelen:0
```

Device Name	Function
lo	Loopback interface
ethN	Ethernet
pppN	Point to Point

Device Name	Function
tunIN	Tunnel device

A ":" after a device name denotes a sub interface, thus eth0:1 above is a virtual device running on eth0 (the first Ethernet device). As you can see, there is a lot of information available from the ifconfig command, such as the hardware and IP addresses of the devices.

Peripheral Devices

Peripheral devices include typical items like monitor, mouse, and keyboard, and also devices like scanners and printers.

Monitor, mouse, and keyboard problems are usually very easy to diagnose. Installation under Linux is likewise very easy. When upgrading video hardware on a machine running X-Windows, it is best to reconfigure X to take advantage of the new hardware.

The other devices will most likely need some sort of driver to be installed in order to function. For example, a printer will require the "lp" device to function. USB devices will require the USB drivers, in addition to any specific drivers for the device:

# lsmod			
...			
hid	11776	0	(unused)
input	3488	0	[usbkbd keybdev hid]
usb-uhci	20720	0	(unused)
usbcore	49664	1	[usbkbd hid usb-uhci]

Most distributions take care of making sure that the USB system is ready.

Modems are best set up through one of the wizards mentioned earlier for ppp configurations, rp3-config or Kppp. Most problems with modems will be related to IRQs and ports. Linux does not currently support so called "WinModems", which are mostly software based. Remember that a modem that would be on COM1 in DOS would be on /dev/ttyS0 in Linux. COM2 is /dev/ttyS1, and so on. Mixing this up is a common error.