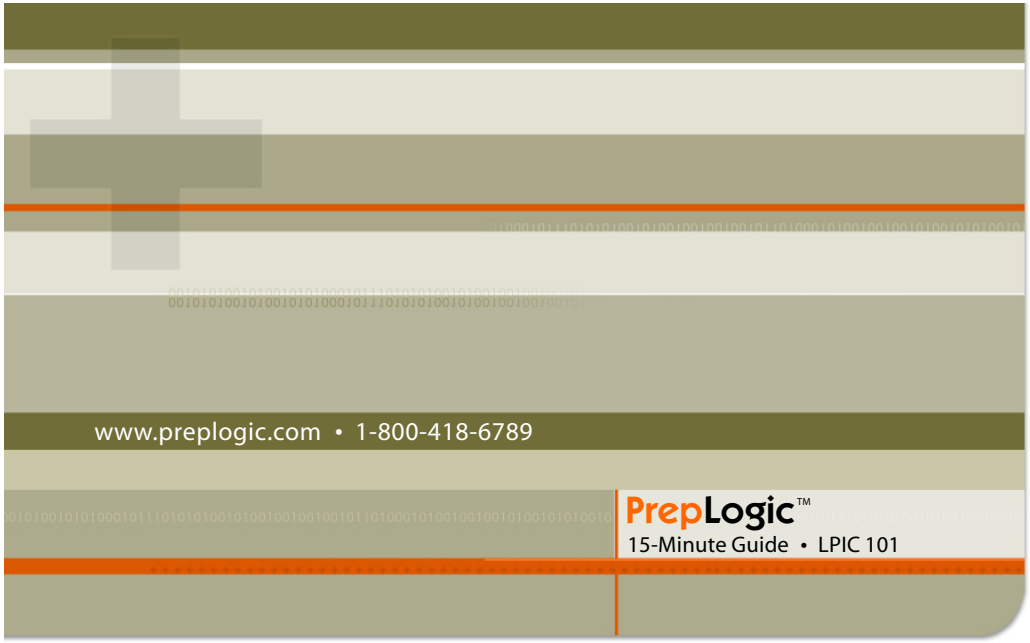

The /proc/ file system

The /proc/ file system is created at each boot time. The files are mirrored (i.e., copied) from existing system settings and placed into the /proc/ directory. It reflects the system's current configuration and contains files and directories that the system uses to configure how it functions.

File or Directory	Description
/proc/ioports	Provides detailed information concerning the hardware resources that the system is using. If a device is recognized, it will receive an I/O address. An example of an I/O address is: 03c0-03df. If the device does not have an I/O address, it has not been recognized.
/proc/interrupts	Describes the particular interrupt used by a device. Devices can share interrupts. If you are using a system with two CPUs, you will receive a listing of interrupts for each CPU.
/proc/dma	Describes channel that ISA and other devices register for Direct Memory Access (DMA).
/proc/pci	Contains information about operational PCI devices.
/proc/partitions	Provides detailed information about operational partitions.

table continued

File or Directory	Description
/proc/cpuinfo	Statistics about the CPU(s) currently operating,
/proc/uptime	Shows how long the system has stayed operational. The first number shows uptime in seconds. The second number shows how many seconds the running system has been idle.
/proc/sys/net/ipv4/	Contains various files that allow you to view, enable or disable network resources. For example, changing the value of /proc/sys/net/ipv4/icmp_echo_ignore_all from 0 to 1 will prohibit the system from sending or receiving all ICMP packets, including those used by the "ping" command.

SCSI device names

SCSI ID Number	Linux device file name	Description
0	/dev/sga	The first device
1	/dev/sgb	The second device
2	/dev/sgc	The third device
3	/dev/sgd	The fourth device
4	/dev/sge	The fifth device
5	/dev/sgf	The sixth device
6	/dev/sgg	The seventh device
7	/dev/sgj	The eighth device

Mobile Edition

Enjoy your 15-Minute Guide the way it was meant to be, on your iPod, iPhone, Blackberry and more! Visit us at preplogic.com/m

Setup Non-IDE Devices: Additional SCSI Information

The /proc/scsi/ directory

Files in the /proc/scsi/ directory will help you determine information about your SCSI devices. If these files are not present, no SCSI devices can be recognized on your system. A typical file in this directory includes /proc/scsi/device_info, which provides information about attached devices.

The /proc/scsi/ directory will also contain additional subdirectories. They are often in the following format:

/proc/scsi/scsi0

/proc/scsi/scsi1

Each of these subdirectories contains additional subdirectories and files that allow you to determine the nature of the controller cards and additional installed devices. SCSI driver devices are assigned directories and files according to the following pattern:

/proc/scsi/<driver_name>/<scsi_adapter_number>

The <driver_name> section will be named by the vendor. The syntax of the <scsi adapter_number> portion of the device has a very specific naming scheme.

SCSI addressing scheme	Description
SCSI adapter number	Describes the position of the particular SCSI adapter card. The system's BIOS generally detects SCSI devices arbitrarily at boot. SCSI adapter numbers are assigned by the kernel, beginning at zero (e.g,. 0-6). Often called the "host" portion by Linux systems. An adapter is often called the Host a Bus Adapter (HBA). An HBA can control more than one SCSI bus, or channel.
Channel number	Lists the channel (or channels) used. Called the "bus number" by Linux systems.
ID number	Often called the "target number."
Logical Unit Number (LUN)	Simply called the "LUN" by Linux systems.

Linux Installation & Package Management

Format: The practice of applying a file system to a partition. Also refers to the file system used on a partition. For example, the question, "What is the format of the partition?" is basically asking what file system was applied to the partition. Many different formatting commands exist, including mkfs, mkfs.ext2, mkfs.ext3, mkfs.reiserfs, and so forth.

- **Volume**: A formatted partition that is given a name. A volume can reside on the local system, or on a remote system. For the LPI 101 exam, you need only be concerned with local volumes.
- **Master Boot Record**: A small portion of the hard disk that contains all information about the hard drive's layout (i.e., partition scheme). Information includes whether or not the partition is active and its location on the hard drive; the MBR is 512 bytes in size. It is the first "block" of the hard disk. The MBR is loaded into memory at boot time and informs the system about the location of partitions on the system. The MBR is created by the fdisk application.
- **Mount point**: A directory on the local hard drive that is used to house a partition. Examples of mount points include the /, /boot/, and /home/ directories, among others. You can also create your own mount points in order to mount a partition/volume.
- **Boot loader**: An application responsible for loading an operating system and MBR into memory. A boot loader generally has several stages. Each stage has different responsibilities. The most popular boot loaders in Linux include GRUB (Grand Unified Boot Loader) and LILO (Linux Loader). GRUB has become the standard boot loader.
- **Swap space**: Hard disk space that is used by the operating system as Random Access Memory (RAM). It is often called "virtual memory." The virtual memory is placed into a special file called a "swap file." Virtual memory is obtained from the swap file one portion at a time. Each portion is called a "page." Swap space is created by the mkswap command in Linux. Generally, the swap space on your system should equal roughly twice your computer's RAM. However, if your system has 2 GB of RAM, your swap space needs to be only twice the amount of RAM above 2 GB. Never have less than 32 MB of RAM.

Basic and production server partition schemes

Mount point	Device and partition name	Description	Size
/boot	/dev/hda1	Contains files necessary for booting. (Remember about 1024 cylinder limit.)	100 MB, at the most.
/	/dev/hda2	Contains all the files and directories common in a Linux system (e.g., /home/, /var/, /tmp, /etc/, /bin/, /usr/, and /usr/bin/.	77 GB
swap	/dev/hda3	Treated as RAM.	2 GB
proc	/proc/	Created at boot time	Variable

Mount point	Device and partition name	Description	Size
/boot	/dev/sda1	Boot partition	100 MB, at the most, same as the layout for a basic "desktop" system.
/	/dev/sda2	Also known as the "root" file system.	50 GB. Still the largest partition.
/home/	/dev/sda3	Contains the /home/ directory only. Now, the /home/ directory is not in the same mount point as /, which allows the administrator to unmount the /home/ directory and troubleshoot problems.	449 GB
/var/	/dev/sdb1	Contains files for the Web and database servers (Apache and MySQL, respectively).	400 GB, in order to accommodate the entire server and log files.

table continued

Mount point	Device and partition name	Description	Size
/tmp/	/dev/sdb2	Contains files generated by user sessions on the Web server. The space must be larger than in a standard "desktop" system.	97 GB
swap	/dev/sdb3	Used as RAM, just as in a desktop system.	2 GB, according to the formula described previously.
proc	/proc/	Created at boot time.	Variable

Install a boot manager

Grand Unified Boot Loader (GRUB): The *de facto* standard for booting Linux. For more information, go to the GNU GRUB homepage, at http://www.gnu.org/software/grub.

- **Linux Loader (LILO)**: Today considered a legacy boot loader, but still often found. The LILO homepage is at, http://lilo.go.dyndns.org.
- **NT Loader**: Used to boot Windows systems, but often used with either LILO or GRUB. Ships with Windows systems.
- **Yaboot**: Used to boot Linux on modern Macintosh OS X systems. The Yaboot homepage is at http://yaboot.ozlabs.org.
- **BootX**: Used on older Macintosh systems. The BootX homepage is at http://penguinppc.org/bootloaders/bootx.

Boot loader terms

Below are important boot loader terms:

- **Superblock**: The first 512 bytes of any partition. In file systems, such as as ext2 and ext3, the superblock is copied in spaces throughout the partition. The superblock contains the block sizes used in the partition.
- **Inode (Information node)**: The term used for a file that outlines information about all files and directories on the hard drive. An inode contains information about the file's location on the hard drive, as well as the file's permissions, owner, and file type (block file, character file).

Mobile Edition

Enjoy your 15-Minute Guide the way it was meant to be, on your iPod, iPhone, Blackberry and more! Visit us at preplogic.com/m

The inode tables – Contains numbers that help the file system find and refer to files and directories.

Bootload Stages:

A boot loader boots an operating system in roughly two stages –

- **Stage 1**: The system BIOS reads the initial 2 bytes of the MBR, then begins loading the boot loader. The boot loader then loads the MBR (partition table), then loads the portion of the boot loader that begins activating the operating system.
- **Stage 1.5**: Searches for the executable file that contains stage 2 of the boot loader. This stage is sometimes considered a separate stage, but is really part of stage 1. This is how the LPI exam views boot loaders.
- **Stage 2**: Finishes loading the boot manager, and then provides a menu to the user that allows you to choose a particular file system. When Linux is chosen, Grub helps read the ramdisk (a portion of RAM used as a hard disk), and then load the kernel.

Extracting code

Application	Description
gzip	Can compress or decrompress any file. A file compressed using gzip will often have the .gz ending, or the .tgz ending, if gzip is used to compress a tar file. Examples of using gzip include: • gzip -d tarball.tgz: Decompresses a file named tarball.tgz. You will still need to run the "tar" command against this file. See below. • gzip -l tarball.gz: Lists the attributes of the file.
gunzip	Decompresses a file compressed by gunzip. For example: • gunzip tarball.gz: Uncompresses the contents of the tarball.gz file. • gunzip tarball.tgz: You will still have to run the "tar" command against this file to view the contents of the source file.
unzip	Another compression utility. To uncompress a file named tarball.gz, you would issue the following command: unzip -u tarball.gz

table continued

Application	Description
bzip2	A more efficient compression utility. To uncompress a file named tarball.gz, you would issue the following command: bzip2 -d tarball.bz2
bunzip2	The standard command for unzipping bzip2 files: bunzip tarball.bz2
uncompress	An older, less-often used utility. Files created with the compress utility often have a large Z at the end: tarball.tgZ Use uncompress as follows: uncompress -Z tarball.Z
tar	Files with source code in them are often called "tarballs," because they have been created by the tar application. The tar command takes individual files and directories and combines them into one single file. Useful for source packages, because it is capable of combining an entire directory structure into one file. The tar command in and of itself does not compress files. Examples of the tar command include: • tar -xvf tarball.tar: Extracts the contents of a file named tarball.tar. • tar -zxvf tarball.tgz: Extracts and decompresses the contents of a tarball named tarball.tgz. The -z option allows you to summon the gzip command; tar does not have native compression or decompression ability. • tar -jxvf tarball.bz2: Decompresses and extracts the contents of a tarball named tarball.bz2, which was created using the bzip2 compression application.

Use Debian package management

At one time, the LPI 101 exam allowed you to choose between RPM and Debian-based package management. This is no longer the case; you must learn to use both package managers. This section will discuss the applications used in the Debian-based package manager. The next section will discuss RPM commands.

Debian Package Management Command	Description
apt-get	The preferred method for installing and managing packages. Used to install, uninstall and manage all packages on a system. Uses package repository sites listed in the /etc/apt/sources.list file to obtain files. The most capable and sophisticated tool.
/etc/apt/sources.list	The file that allows apt-get to consult various repository servers.
dpkg	A manual command, much like rpm, used to install individual packages. Most often used to install local packages, but can also be used to install individual packages from remote systems.
dpkg-reconfigure	Used to reconfigure installed Debian packages.
/etc/dpkg/dpkg.cfg file	Provides default options for dpkg every time it runs. Any value you enter needs to be separated by whitespace and/or an `=' sign.
alien	Converts non-Debian packages to those that can be installed by either apt-get or dpkg.
dselect	A graphical front end for managing Debian packages. Not an X Window application. Can be run from any terminal (e.g., SSH, Telnet). Not emphasized in the LPI exam.
Synaptic	An X Window front end to apt-get. Not emphasized in the LPI exam.

Killing processes

You can issue "signals" to processes in order to terminate them. To terminate, or "kill" a process, you can use the following applications:

- **kill**: Allows you to terminate applications, but only if you know the application's process id (known as the "PID"). For example, to kill a process, you must first learn the PID (by, for example, using the ps command), then issue the kill command. For example, here are the steps for terminating an application called ethereal:

  ```
  ps aux | grep ethereal
  user1   12885  0.0  1.7  28844 18552 pts/0    S    00:17   0:00 ethereal
  ```

- **kill** 12885
- **killall**: Terminates applications by name: killall ethereal
- **Top**: Highlight the process, then press k, and specify the application's PID.

Signaling applications with kill and killall

The kill and killall commands allow you to send various types of signals to a process. The two most important signals are:

- **KILL** (or, -9): Kills a process authoritatively. Does not allow a system to override. Sometimes, applications will refuse to be terminated. The -9 or KILL signal ignores such overrides and ensures that the process is terminated.
- **HUP** (or, -1): Used to stop and restart a process.

Killing problem X Window applications after logout

Sometimes, after an X Window session has closed, processes that should run only during an X Window session will still be running. To properly terminate such problem applications, you will need to do the following:

- Discover the application's PID (e.g., use ps or even top).
- Use the kill or killall command on the process.

Make sure that you can read all of the output from the kill and killall commands, as well as determine the PID and the Parent PID (i.e., the process that launched the sub-process). You will want to know how to kill a "child" process, rather than the parent process, and vice versa.

Mobile Edition

Enjoy your 15-Minute Guide the way it was meant to be, on your iPod, iPhone, Blackberry and more! Visit us at preplogic.com/m

Search text files using regular expressions

When searching through text files, you can use regular expressions and applications, such as grep and sed. The following is a brief discussion of these powerful applications:

- **grep**: Searches through standard output or standard input to select a matching pattern. Also known as egrep, fgrep, and rgrep. Often used with the pipe (|) to search through program output and report back only the relevant information you want to see.
- **sed**: "Sed is short for "stream editor." Ideal for sifting through standard output and file contents to perform substitutions. It reads information one line at a time. Sed uses the classic regular expressions to edit files non-interactively. Whereas an editor such as vi, emacs or pico, allows you to load a program into an interface for editing, you run sed as a one-time command, using options and regular expressions to edit the file as it streams through a memory buffer.

Access Control Lists

Most modern, popular operating systems use an Access Control List (ACL) to determine which users can access a particular resource. This is the case for all mainstream Windows, Solaris, BSD and Linux systems. An ACL is a database of information that the system consults to protect and organize resources. Any system that uses an ACL also requires the following:

- **The concept of resource ownership**: Any resource on the system (e.g., any file or directory) must be assigned a user.
- **File permissions**: Any resource must be given a rule.

Command	Description
chmod	Used to change permissions on files and directories. The syntax for the chmod command is: chmod permission_value file_or_directory
chattr	Adds special permissions on an ext2 or ext3 file system, including the following options: • -i: Sets the "immutable bit," which means that the file cannot be deleted or modified, even by root. • -a: Sets the directory or file (in some systems) so that information can be appended. Nothing that has been added can be deleted. It is important to understand that these attributes do not appear in the standard or long directory listing provided by the ls command. You must use the lsattr command.
lsattr	Lists the attributes assigned by the chattr command.
umask	Determines the default permissions a file or directory has when it is initially created. The umask can be set to be different from one user to the next.