CramSession
The Original Study Guide

Over 4 Million Downloaded

LPI

LPIC1 - Junior Level Administration Part 1

101 101 101 101 101 101 101 101 101 101 101

James Stanger - Author
Emmett Dulaney - Technical Editor

Steven Johnson - Managing Editor

Your Trusted
Study Resource for
Technical Certification

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

LPIC1- Junior Level Administration Part 1 (Exam: 101) Study Guide

Copyright © 2006 by PrepLogic, Inc.
Product ID:001915
Production Date: October 2, 2006

Warning and Disclaimer

Volume, Corporate, and Educational Sales

Abstract

This study guide prepares candidates to pass the LPI 101 exam, which is designed to ensure that professionals who use Linux have foundational knowledge of the operating system. This exam is designed to test knowledge of essential commands, technologies, and environments used in Linux. It is meant to test knowledge levels for various Linux professionals, including system administrators, programmers, and system analysts. The LPI 101 exam is the first you must pass in order to become LPI Level 1 certified. To earn LPI Level 1 certification, you must pass both the LPI 101 and the LPI 102 exams. You can learn more about LPI's exams at www.lpi.org

What to Know

LPI's goal is to certify professional-level knowledge of Linux. As with all LPI exams, the 101 exam is vendor-neutral. The exam is based on a thorough Job Task Analysis (JTA) and uses standards such as the Linux Standards Base and other relevant standards and conventions. The 101 and 102 exams are designed to verify essential knowledge for system administrators, programmers, developers, and system analysts.

It is important to understand that even though the LPI says that the 101 exam tests basic capabilities for junior-level administrators, it is a challenging exam. To pass the 101 exam, you need to know detailed information about the following subject areas:

Hardware & Architecture

Linux Installation & Package Management

GNU & Unix commands

Devices, Linux File systems, and the File System Hierarchy Standard

The X Window Systems

The study guide will help you learn about the specifics of each of the learning areas discussed above. If you have additional questions, go to www.lpi.org to learn more about each of these areas.

Tips

Following are some study tips for the LPI 101 exam:

Get hands-on experience: Although a study guide such as this will help you learn about the exam, it is no substitute for hands-on experience. The LPI exams have a reputation for focusing on the knowledge areas that professionals require. It is highly recommended that you spend significant time working with a Linux system before attempting to pass these exams.

Focus on the objectives listed at www.lpi.org: The most experienced programmer or system administrator will have problems with a question for which he or she has not studied. Often, Linux professionals who take the LPI exam find themselves challenged by a piece of information they really do know, but have not used or studied recently.

Get a good night's sleep before the exam: You will find that you will remember commands and be able to think more clearly if you are well-rested and not in a hurry. The LPI exams require you to remember various commands. The exams also require you to think through issues and choose the best solution, just as any Linux professional would on the job.

First, answer all questions you know as you go through the exam. Mark questions that you do not know, and come back to them later. Do not waste time trying to answer a question you do not know. It is best to calmly go through the exam and focus on what you know, and then work through problem questions.

Write down notes as soon as your proctor allows you to sit down in front of the exam. You will not be allowed to bring in any notes, or any device that sends or receives a signal. However, you will be given sheets of paper or a couple of erasable pads to use. Immediately write any concepts or commands on these sheets. So, even though you cannot bring in a "crib" sheet with you, you can still create one from memory. This technique will help you considerably through the exam.

Arrange for alternative transportation, and arrive early: Do not arrive at the testing center at the last minute. Arrange to have someone stand by as a backup plan to drop you off at the examination room. You do not want to forfeit your exam fee. You also do not want to arrive at the examination "stressed out."

Choose an appropriate examination time: Arrange to take the exam at a time of day when your mind is the most focused. If you are a "morning person," then arrange a time in the morning. You will find testing centers will have times for almost any schedule.

Table of Contents

Topic 101: Hardware & Architecture

Objective 1.101.1: Configure Fundamental BIOS Settings: The /proc/ file system

To fulfill this objective, candidates must be able to verify IRQ, DMA and I/O settings on the system. Students also need to be able to change these settings. To perform these tasks, you first must understand the /proc/ file system.

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. Essential files and directories include those in Table 1:

Table 1

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.
/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.

To disable devices, you can take the following steps:

Edit files in the /proc filesystem. Understand, however, that if you edit these files, the settings you generate will remain valid only until the next boot. You can, of course, create scripts to automatically alter files in the /proc/ directory.

Change "jumper" settings on the devices you want to attach to the system.

As you add systems, verify that that there are no conflicting devices. For the exam, make sure that you understand the format of the files given in the above table. Also, make sure that you check on the /proc/ioports and /proc/interrupts files. It is possible for devices to share the same interrupt. However, they should not share the same I/O address.

Objective 1.101.1: Configure Fundamental BIOS Settings: Configuring Ports and Creating Headless Linux systems

Many times, an integrated peripheral can be enabled or disabled by going in to the BIOS settings and configuring the settings for each port. Ports for integrated peripherals can include:

Serial ports: Often used to connect a "null modem" cable, which can be used to transfer data between systems, connect to a modem, or connect to a keyboard.

Parallel ports (IEEE 1284): Used in various situations, but most often to connect to a printing device. Although parallel ports are less used in production systems, it is important to understand that you can configure speeds for these ports. Available speeds include:

Compatibility mode: The original specification, for simple 8-bit transfer.

Nibble mode: Designed for printers, allows two-way communication at 4-bits at a time, but only one way at a time. In this way, transmission is unidirectional.

Byte mode: A legacy mode that allows parallel ports to receive data more quickly. Disables the drivers that would place outgoing data onto the line.

Enhanced Parallel Port (EPP): Half-duplex communication, for non-printing devices.

Extended Capability Port (ECP): Enables half-duplex, bi-directional communication that allows compression. The fastest standard (2.5 Mbps).

SCSI devices: Production Linux systems often use SCSI devices. The order in which the SCSI card is detected can determine the difference between a system that boots into Linux, and one that simply hangs.

Serial ATA (SATA) devices: Serial devices include current hard drives.

IDE devices: You can configure the order in which floppy, hard, CD and DVD drives are configured. Although floppy disks are very rare today, many servers still have them, and they are necessary for the initial booting of systems when performing updates. Also, IDE drives are in many older systems, which many beginners in Linux may be using. After all, many older systems that are just “lying around” can run Linux and are very useful in a home lab setting.

USB ports: USB devices are not considered integrated devices by LPI. Nevertheless, you should understand that you can enable or disable the USB devices through BIOS.Headless Linux systems

There will also be times that you will have to use a “headless” Linux system. A “headless” system is a computer that does not have a keyboard or monitor attached. Sometimes, these headless systems are servers connected to a device that allows the server administrator or programmer to switch from one computer to the next. At other times, they are embedded Linux systems, where the operating system works behind the scenes, and there is no command-line or GUI interface for the user.

BIOS systems in headless computers require special configuration. For example, many x386 systems will not boot if a keyboard is not attached. To solve this problem, enter into the system's BIOS and configure it to ignore a missing keyboard.

You access a system's BIOS settings at boot during the boot time. During the boot time, the key you press to enter the system's BIOS configuration program differs from computer to computer. Sometimes, the F10 key will allow you to access the BIOS settings. With other systems, the DELETE key must be pressed. The system Power On Self Test (POST) screen will inform you.

Once you have accessed the BIOS configuration program, find the screen that allows the system to skip the keyboard check. Some systems will also cause problems if a graphics card is not installed. Similarly, find the settings that allow you to bypass checking for a card.

Troubleshooting headless systems

Below are some common issues when configuring headless system:

Keyboard and video will not work, even when they are attached: Make sure that you have activated the serial port and that the mappings are correct in the /etc/inittab file.

The system won't boot to a GRUB or LILO prompt: Many times, a system will not proceed if it detects that a keyboard is missing. If your system will not boot, make sure that you have disabled either the keyboard port, or that you have disabled the portion of the POST that checks for the keyboard. The option to look for is often called “Halt on,” or “Keyboard halt.”

Note: It is possible to use a null modem cable to test ports to make sure that they are usable.

Checking serial ports

When checking serial ports, use the dmesg command. You will be able to determine the ports that are running:

```
$ dmesg | grep ttyS

ttyS0 at 0x03f8 (irq = 4) is a 16550A
```

Communication programs

When communicating with the ports, you can use various applications, including:

minicom: The standard terminal.

gtkterm: A GUI-based alternative, based on Hyperterminal.

If you find that you can't log in as root on the terminal, the /etc/securetty file may not have been updated. Any terminal listed in this file will allow root logins. If the terminal is not listed, root cannot log in. To update the file, simply place the terminal name in it. You must be root. For example, if you wish to allow ttyS0 to allow root logins, you simply add the following line:

```
ttyS0
```

You may not wish to have the terminal to boot into X, assuming that the headless system is being used as a server. To have it boot into a standard terminal, enter the following in /etc/inittab:

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

Finally, it is possible to allow the serial terminal to communicate with the prompt issued by GRUB, enter the following lines in /boot/grub/grub.conf:

```
serial --unit=0 --speed=9600
terminal --timeout=2 serial console
```

1.101.3 Configure Modem and Sound cards: Using the setserial command

Make sure that you understand how to use the common commands to configure both a modem and sound card. The first step when configuring a modem or modem card is setting the proper port speed. To set the port speed, use the setserial command. A few examples are given below:

```
[root@jacob root]# setserial /dev/ttyS0 uart 16550A port 0x03f8 irq 4 baud_base 115200 spd_vhi
```

In the above command, you use the setserial command to configure the first terminal, called ttyS0. You can specify any port. The most common ports are ttyS0, ttyS1, ttyS2, and ttyS3. The additional commands specify the uart used, which in this case is the most common, 16550A. The UART is the Universal Asynchronous Receiver-Transmitter, which is responsible for managing serial transmissions between a computer and a peripheral, such as a modem. External modems rely on the UART that is placed on a PCI serial card, which is a modular component. If you are experiencing a problem with your UART, simply replace the serial card with an updated UART. Internal modems, however, have their own UART. In this case, you will have to replace the entire modem.

The command then specifies the I/O port address and IRQ to use (0x03f8 and 4, respectively). The baud base option sets the basic baud rate, which is usually set to 115200, the fastest baud rate the 16550A UART can support. Finally, the spd vhi option sets the UART to 115 Kbps. Additional setserial options are given below:

`spd_normal: Tells the UART to run at 38.4 Kbps.`

`spd_hi: Sets the UART for` 57.6 Kbps.

`spd_vhi:` Programs the UART to use a speed of 115 Kbps.

`auto_irq: Use any available IRQ.`

skip_test: Do not conduct a test of the UART.

autoconfig: Has the kernel automatically configure the serial port.

You can also use the setserial command to find out information about multiple ports on the system. To do this, use the -g option:

[root@jacob root]# setserial -g setserial -g /dev/ttyS*

The setserial command will then search all ports that begin with ttyS, and give the current settings. To disable a serial port, use the following command:

`[root@jacob root]# setserial` to set it to "uart none"

Winmodems

One of the most common legacy problems with Linux systems are modems that use proprietary software and the system's CPU to function properly. Such modems are not the industry standard, and they are called “Winmodems.” Generally, these do not work with Linux systems. To detect a Winmodem, you can:

Consult the Hardware Compatibility List (HCL) of your Linux distributor.

Use the lspci command to view output concerning the device. If you see the text “Winmodem,” purchase another modem.

Internal modems can be Winmodems. External modems are never Winmodems. To solve the Winmodem problem, the best solution is to ignore the existing Winmodem and obtain an external modem.

Note: Winmodems are less of a problem now than they used to be 5 years ago. Still, consider the Winmodem problem if you are experiencing difficulties. Also, don't forget that you may be using a card with an older UART chip.

1.101.4 Setup non-IDE Devices: SCSI device names

The most common non-IDE device in Linux is the SCSI device. SCSI devices are serial devices and can be “daisy chained” together. Traditionally, SCSI devices have been considered to be faster and more reliable than SATA, and especially IDE devices. Many SCSI devices exist, including hard drives, printers, and scanners. In a production server environment, hard drives are the most common SCSI device. SATA devices have become much faster and reliable, recently, though SCSI remains an important standard.

SCSI device names

IDE and SATA devices are named after a certain pattern. The IDE device named /dev/hda is the first device. The device named /dev/hdb is the second device, and so forth. SCSI devices are named after the pattern shown in Table 2:

Table 2

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

The device with the ID of 7 (the eighth device) is usually the SCSI controller card.

The scsi_info command

Using the scsi_info command reads the contents of the /proc/scsi/scsi and renders it into user-friendly information. You can use it to gather information about the SCSI device. When working with SCSI devices, remember that you must:

Properly configure the SCSI BIOS settings: Make sure to provide the proper SCSI ID number, including the Logical Unit Number (LUN), which allows the system to determine the difference between individual SCSI devices.

Terminate each device: If improperly terminated, the device will not be recognized, or will malfunction. You must use a special piece of hardware, called a terminator, to do so. To terminate, place a terminator at the end of the particular SCSI chain. If you do not, the SCSI signal will not be properly absorbed, and will bounce back through the SCSI cable, and cause communications to fail. Most newer devices have internal terminators. However, remember that many SCSI devices exist.

Load the appropriate modules: SCSI devices often require you to install specific modules, which act as device drivers. If you do not have the proper module installed, the device will not function.

1.101.4 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, as shown in Table 3:

Table 3

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.

Whenever a SCSI device is listed, information will be in the following format:

scsi(_adapter_number), channel, id, lun

When viewing this information in a Linux system or kernel log, you would see the following syntax:

host, bus, target, lun.

For the exam, make sure that you know how to:

Read the output of various files in the /proc/scsi/ directory.

Identify the purpose of a LUN.

Understand the importance of installing the proper SCSI modules.

Terminate SCSI devices.

Troubleshooting SCSI

When analyzing a problem with SCSI devices, consider the following:

Making sure that the SCSI card is properly placed (i.e., seated) into the motherboard slot.

Upgrading the card's firmware.

Checking the card's physical position in the motherboard. Sometimes, a card will work properly if it is placed before or after other PCI cards.

Check the driver used for the card; you may be using the wrong driver, or the driver may have become corrupted.

Check for the correct SCSI ID. The default host adapter ID is 7, so any new card you install should not use that one. Many professionals will use ID numbers 0 through 6, or 8 through 15.

Check for termination problems.

Check for internal and external devices; sometimes, the termination can become confused if both internal and external devices exist.

Sometimes, server administrators will try to mix different SCSI types on one adapter.

1.101.5 Setup different PC expansion cards: The lspci and lsusb commands

The lspci command

The lspci command allows you to view all PCI cards recognized by your PC. The -v and -vv options provide additional "verbose" information, including I/O addresses and additional vendor information. Below is an example of the lspci command output:

```
$ lspci
00:00.0 Host bridge: Intel Corp. 430HX - 82439HX TXC [Triton II] (rev 03)
00:07.0 ISA bridge: Intel Corp. 82371SB PIIX3 ISA [Natoma/Triton II] (rev 01)
00:07.1 IDE interface: Intel Corp. 82371SB PIIX3 IDE [Natoma/Triton II]
00:0f.0 VGA compatible controller: Cirrus Logic GD 5436 [Alpine]
00:1d.0 USB Controller: Intel Corp. 82801DB USB (Hub #1) (rev 01) (prog-if 00 [UHCI])
```

The first two lines contain information concerning the PC's "north bridge" and "south bridge." The "north bridge" controls the system's RAM, CPU, and AGP port. The "south bridge" controls the system's access to its BIOS, IDE, SATA, and USB devices. Notice that this is quite an old system, in that it still uses an ISA bus. The third line describes the system's particular IDE device. The fourth device describes the VGA controller. The final line describes a PCI USB controller card.

Using the lspci command is useful when you need to:

Identify the devices recognized on the system.

Determine the vendor of the recognized device, to see if you can find a device driver.

The lsusb command

The lsusb command provides detailed information about all attached USB devices. Below is an example of the lsusb command output:

```
[root@jacob root]# lsusb
Bus 001 Device 003: ID 0c76:0005 JMTek, LLC. USBdisk
Bus 001 Device 002: ID 046d:c00e Logitech, Inc. Optical Mouse
Bus 001 Device 001: ID 0000:0000
```

The first line describes a typical USB “jump drive” or “thumb drive.” The second line describes a Logitech mouse. The third line describes the actual USB controller. The -w option provides verbose information.

If you require especially verbose information, you can use the -v and the -vv options. The lsusb command does not work on systems that do not have the /proc/bus/usb/ file. Otherwise, you will receive the following error message:

```
Cannot open /proc/bus/usb, No such file or directory (2)
```

To determine exactly how a USB device is attached in the USB hierarchy, use the -t option.

1.101.5 Setup different PC expansion cards: Hotplug versus coldplug devices

Linux systems are capable of recognizing when a device has been removed. These Linux systems can therefore add or remove devices without first powering down. Such devices are called “hotplug devices,” because the hotplug command is used to interpret a significant event, such as the removal or addition of the device.

Hotplugging in Linux became possible with the 2.4 kernel, in 2001. It is now a standard feature and has been extended in the 2.6 kernel. Hotplugging is a feature available to USB, IEEE 1394 (FireWire), SCSI, Cardbus, and network devices, among others.

The hotplug command is capable of the following actions:

Recognizing that a device has been added or removed.

Loading or unloading the module that runs the device.

Stopping or starting a device based on an event other than the removal or addition of a device. Events include closing the lid on a laptop computer, executing a series of keystrokes (e.g., hibernate key sequences), and plugging or unplugging system power (in a laptop).

The hotplug commands and files include those in Table 4.

Table 4

Hotplug commands, terms, files, and utilities	***Description***
/sbin/hotplug	The actual hotplug binary.
/etc/hotplug/*	Directory that contains hotplug configuration files.
/etc/hotplug/NAME.agent	Files named after each particular hotplug event or device. Each file is called an “agent.”
/etc/hotplug/name/driver	Scripts that help agents load drivers and accomplish various startup and shutdown tasks. Agents load these scripts.
/etc/hotplug/usb/DRIVER.usermap	Provides dependency information for modules loaded by the agents and users.
/etc/init.d/hotplug	The startup script. Can be used to stop and restart the hotplug facility, as well as determine its status.

Hotplug devices must be properly recognized by the system. You can verify the addition of such devices by consulting the appropriate file and subdirectory off of the /proc/ directory (e.g., /proc/usb/*). If you have a problem with hotplug devices, you can comment the devices out of the /etc/hotplug/ subdirectories. You can also use the /etc/hotplug/blacklist file to exclude certain devices.

You can learn more about Linux hotplugging at http://linux-hotplug.sourceforge.net.

1.101.5 Setup different PC expansion cards: The difference between hotplug and coldplug

Coldplugging occurs when you must power down a system before adding a device. The coldplug facility is responsible for discovering devices. This facility searches devices listed in the /etc/sysconfig/hardware/hwcfg-static-* set of files. The coldplug facility also searches for files in the /etc/hotplug/ directory. The coldplay facility will discover any device that is not auto-detected by calling the hwup command.

Generally, the following devices are hotpluggable in Linux:

USB devices.

IEEE 1394 (FireWire) devices.

PCMCIA devices.

Mouse devices.

Printers.

Some network adapters.

The following devices are only beginning to receive hotplug support:

Hard drives.

RAM.

CPUs.

Most network adapters.

Most serial devices.

Most parallel port devices.

Troubleshooting hotplug

Sometimes things go wrong with hotplugging. When problems occur, take at least the following steps:

Verify that the module for the device is installed and working. Perhaps the problem does not lie with hotplug, and is with the driver.

If you are working with a USB device, make sure that the phrase "usb-serial" appears in the /proc/devices file.

When configuring hotplugging for USB devices, make sure that the usb-uhci and usb-ohci modules are installed. These modules are the universal device drivers for USB hubs.

Do not trust GUI-based applications to give you correct information. Check the /var/log/messages and /var/log/syslog files for errors.

If a device “hangs” at boot time, verify that the hotplug facility is not causing the problem. You can perform this verification by skipping the hotplug scripts at boot time. If hotplug is causing the problem, use the /etc/hotplug/blacklist file to exclude the offending device.

For network devices, hotplugging often fails to run files in the correct order, resulting in a failed network interface. You may have to create a custom script to solve the problem. Specifically, make sure that the device gets recognized first, and then have the script assign IP address information.

1.101.6 Configure communication devices

Although modems are not used as often as 10 years ago, it is still important to know how to use them. Modems are used in backup solutions. Also, you will be surprised to learn how many people still use modems every day.

The first step in configuring your modem is to avoid device conflicts. Take steps to verify that adding the modem to your system does not cause a problem. You can perform this verification by reading the following files:

/var/log/messages

/proc/interrupts

/proc/ioports

When viewing the above files, look for the following:

Messages informing you that a conflict exists.

Messages informing you that a device cannot be found.

Information showing that the same I/O address is being used by two devices.

Although two devices can share the same IRQ, in some cases, such messages can still suggest a conflict.

Earlier, you learned how to use the setserial command. If you plan to use an ISDN connection, make sure to load the proper device driver modules for your ISDN adapter. Use the mISDN package, which was formerly known as isdn4linux. Linux users on a Digital Subscriber Line (DSL) connection may need to install the Point-to-Point Protocol over Ethernet (PPPOE) package in order to communicate.

Configuring an outbound PPP connection

To set up a modem for outbound PPP connections, take the following steps:

Obtain a login account from an ISP, and record the user name and password.

Obtain the phone number to the ISP's modem bank.

In some cases, obtain the IP address of the remote system. Usually, the ISP assigns this IP address using the Dynamic Hardware Configuration Protocol (DHCP).

Also in some cases, the IP address of the local system (may also be provided by DHCP, though this DHCP server will be on the local network).

If you are using the Challenge Handshake Application Protocol (CHAP), you will have to put the password into a special file called /etc/ppp/chap-secrets, used to negotiate the connection. If you are using the Password Application Protocol (PAP), you will place your user name and password into the /etc/ppp/pap-secrets file.

Additional /etc/ppp/ files to consider when creating a dial-up connection include:

ip* scripts: Executable scripts designed to activate and deactivate the modem.

pppoe-server-options: Allows you to pass options to the pppoe command, which allows Linux systems to communicate with many DSL connections.

firewall-masq: Enables masquerading so that other local systems can use your dial-up connection to access the

Internet.

firewall-standalone: Allows you to protect your local system from attacks.

options: Passes options to the pppd command.

The peers/ directory: Used by applications that use pppd to connect to remote systems.

1.101.7 Configure USB devices

USB has become the *de facto* standard for peripherals. Linux systems are quite good at supporting various USB devices, from printers to hard drives to scanners. USB devices work automatically because of the underlying subsystem. When things go wrong, it is important that you know how to solve the problem. When configuring USB devices, take the following steps:

Verify that you are using the correct driver for the USB controller on your system. Several different USB controller devices and drivers exist:

Enhanced Host Controller Interface (EHC): An open standard. For USB 2.0.

The Universal Host Controller Interface (usb-uhci): Created by Intel for USB 1.0.

The Open Host Controller Interface (usb-ohci): An open standard. This, usb-uhci and EHC are the most common. For USB 1.0.

SL811: Created by Cypress Computing. Less often used.

Make sure that you have identified and loaded the correct USB driver module.

Make sure that the device has not been placed in the /etc/hotplug/blacklist file.

When using USB, verify that the proper USB controller module (e.g., ehc or usb-ohci) is installed. Also, make sure that the usbcore module is installed. You can do this by running the lsmod command, as follows:

```
$ lsmod
usb-ohci            17888   0  (unused)
  usbcore             56768   0  [scanner ibmcam usbvideo usb-ohci]
```

USB devices are loaded because the udev command, run during the hotplug sequence, discovers the USB devices. When the proper device drivers are loaded, the sequence is complete. One way to determine if all the modules have loaded is to run the usbmodules command. Usually, however, this command is run by the hotplug command.

The LPI exam requires that you also know about the usbmgr daemon. This daemon is responsible for loading and unloading kernel modules. Its configuration files are found in the /etc/usbmgr/ directory. Two key files in this directory include:

usbmgr.conf: Determines which modules to load and unload.

preload.conf: Tells usbmgr to load various modules and files when it runs, so that usbmgr can properly manage the USB subsystem.

Finally, you can verify the function of USB using the commands and files already discussed in previous pages. These commands include:

lspci

/etc/hotplug

lsmod

Standard system log files.

Appropriate files in the /proc/ directory.

When a USB device fails to load, the problem generally lies with the device driver and not necessarily the hotplug system.

Topic 102: Linux Installation & Package Management

1.102.1 Design hard disk layout (Weight: 5)

A partition scheme is an arrangement of partitions on the hard drive. Any operating system, from Windows to Macintosh OS X to Linux, requires a partitioned hard disk. Designing a disk partitioning scheme involves determining the business use for the system. A standard desktop system can have a relatively simple partitioning scheme. If you are using Linux in a production server environment, you will need a more elaborate scheme. They key to selection is understanding the business use of the Linux system.

Before discussing actual disk layouts, it is important to first understand the terms used when creating the layouts.

Essential terms

Block device: A system file that allows access to a device in large pieces of data, called blocks. Blocks can be placed into buffers for more efficient data reading. All hard drives are block devices. Usually block devices establish a cache that can hold multiple characters. A block device file is used to mount a volume. Common block device files include /dev/hda/, /dev/hdb, and /dev/hdb. These block device numbers refer to the first, second, and third installed hard drives, respectively. A character device is one that processes data one character at a time, rather than in buffered chunks (i.e., blocks).

Partition: A discrete portion of disk space. A hard disk can have multiple partitions, or just one. Partitions are created using various tools, including the venerable fdisk application (all Linux distributions), Disk Druid (Red Hat Enterprise and Fedora), and AddPartition (Suse/Novell).

File system: A set of programs and modules responsible for how files and directories are created, accessed, and manipulated. Examples of file systems used in Linux are ext2, ext3, reiserfs, and JFS. To be usable, a partition must first have a file system.

Additional terms

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.

The boot partition and older systems

A special partition that contains the Linux kernel, as well as boot loader information and applications. Generally needs to be only 100 MB at the largest. Older PC-based computers cannot read boot files that reside above cylinder 1024. If a system will not boot correctly after installation, chances are it has encountered this limitation. Resolutions to the problem include:

> Re-installing and creating a /boot/ partition that resides beneath cylinder 1024. This cylinder limit will be different from system to system, but chances are, if you limit the partition to, say, 20 MB, you will not encounter any problems.

Using GRUB - In many cases, GRUB does not have this limitation.

Purchasing a PC made in the past five years.

Basic and production server partition schemes

A hard disk layout for an 80 GB hard disk on a “desktop” system with 1 GB of RAM would look something like Table 5.

Table 5

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

Table 6 gives an example of a layout for a production-based server for server with three 500-GB SCSI hard disks and 4 GB of RAM that houses many users, a Web server (Apache Server), and a database server (MySQL). The layout given below would be ideal for the system's two hard disks. The third would be used as RAID, which will be discussed later.

Note: Any “left over” space reflects the fact that a hard disk running in real-time leaves a certain amount of unused space.

Table 6

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.

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

Notice how many more of the directories in Table 6 are placed in their own partition, as opposed to Table 5.

Remember the following when studying partition schemes:

The root directory is often (/) the largest partition/directory. This is because it contains the /bin/, /usr/, /usr/bin/, and /sbin/ directories, even if you place other directories (e.g., /var/, /home/, /tmp/) in their own partitions. These directories contain most of the binaries used to run Linux and its associated services (also known as “daemons”).

Placing the /home/ directory on a separate partition allows you to conduct backups, as well as troubleshoot problems.

Experienced systems administrators know that the /var/ directory is a busy place. It contains log files and often the files for services (i.e., daemons), such as the Apache Server. Placing it in a separate partition can help you isolate problems and help ensure that your Web server keeps running in case of a problem with another partition.

Placing a particularly “busy” directory on its own partition is often a good idea. For example, suppose that a particular directory consumes all the hard disk space. You can more easily replace the disk with a larger one by simply unmounting the full partition/disk. Then, you can replace the disk and then re-mount the new partition/disk, and then restore all of the necessary information from the backup.

Make sure you understand the naming scheme for partitions on a hard disk. In Linux, the first partition on an IDE or SATA disk is known as /dev/hda, the second is /dev/hdb, and so forth. In a Linux system using an SCSI drive, the first partition would be known as /dev/sda, the second would be /dev/sdb, and so forth.

Creating separate partitions allows you to choose the best file system for each partition.

If you create a more elaborate partition scheme, you will be able to secure partitions more easily. For example, you can create read-only, non-executable and read-only partitions, which can help secure server configuration files and user information that is not expected to contain executable applications or change.

Three types of partitions exist:

Primary: Only four primary partitions can exist on a disk. Each of these four partitions exists within the MBR.

Extended: One of the primary partitions can be designated as an extended partition, which reduces the number of primary partitions from four to three. This partition can itself contain additional “logical” partitions.

Logical: A partition that exists within an extended partition. These logical partitions do not exist within the MBR.

Both IDE/SATA and SCSI disks have logical partition limits. They are:

IDE: 63

SCSI: 15

Partition tools

The primary tool to create partitions remains the fdisk. The fdisk command can be run as a "one-time" command, or interactively. When run as a one-time command, it will run and then issue a report of its findings and other activities, as will ls, cd, ping, or ifconfig. When run interactively, the fdisk allows you to enter a session, where you can read partition information and create a hard disk layout.

Many GUIs exist that use the fdisk as the underlying tool that does all the work. However, you should know all of the command-line options for the exam. Additional applications exist that do not use the fdisk at all. You will learn more about partitioning in a later section.

Additional notes

Ensure the /boot partition conforms to the BIOS requirements for booting.

Do not confuse the root file system (/) with the /root/ directory. The "root" file system can include various directories, including the home directory for the root. The "root" file system also includes any other directory that is not placed into its own partition. The /root/ directory is the home directory for the root user account.

1.102.2 Install a boot manager

A boot manager loads the operating system at boot time. It has three responsibilities:

To read the Master Boot Record (MBR), which is the very first 512 bytes of the hard drive. Of that first 512 bytes, 446 contain the actual boot loader information. 64 bytes contain all partition information (called a "partition table"), and 2 bytes contain a signature, which verifies the validity of the MBR. Alternatively, to read a file that contains boot information.

To load an operating system (e.g., Linux) or another boot loader (e.g., the boot loader for Windows or Apple operating systems).

Allow a choice of operating systems or associated boot manager.

A boot loader can reside in one of two locations:

In a file in the /boot/ partition. This partition must exist in the first sector of the hard drive.

In the MBR. If you already have a boot manager installed on the MBR, you will want to install your Linux boot manager in the /boot/ partition.

The location of the boot loader is your choice. You can install a boot manager at any time, including during or after system installation. Reasons to install a boot loader after installation include:

Upgrades: You may need to upgrade the boot loader software to its latest version. In many cases, an upgrade will be much the same as an installation.

You need more functionality: You may need to change from one boot loader to another in order to boot faster.

Regardless of where and when you choose to install the boot loader, a boot manager must reside in the first 1024 bytes of an operating system's hard disk, if you have an older system. Common boot managers used to boot Linux or other operating systems that are used with Linux include:

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.

Note: You can learn more about boot loaders for Apple-based operating systems at http://penguinppc.org/bootloaders/.

Many additional boot loaders exist, but the ones discussed above are the most common. The LPI exam and this guide will focus on GRUB and LILO for 386-based architectures.

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).

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.

GRUB

GRUB is the most powerful boot loader. It is not subject to the 1024-cylinder limit, as is LILO. Also, whenever you make a setting in the GRUB configuration file, you do not need to run the /sbin/lilo command. Grub can also boot from network disks, hide partitions so the system will not read from them and re-number partitions accordingly. GRUB is actively maintained.

The /boot/grub/menu.lst and /boot/grub/grub.conf files

Debian-based Linux systems (e.g., Ubuntu and Knoppix) use the file named /boot/grub/menu.lst. Systems such as SuSE/Novell Linux and Red Hat use the file named /boot/grub/grub.conf. The grub.conf file must also be used on any MSDOS-based floppy, because the .lst ending cannot be read, due to DOS's inability to read beyond the standard 8.3 convention (e.g., eight letters, a period, and then three letters afterwards). Both files, however, have the same function: They represent GRUB's main configuration file.

This file contains the settings used to boot the system, and provides the menu that the boot loader presents to the user. The file is in two sections: Default options, and the kernel list. The first section allows you to pass parameters to the kernel, enable password protection at boot time, and other settings. For this exam, you do not need to know about passing parameters to the kernel.

It is important, though, to understand a typical entry in the kernel list. Below is a simple entry:

```
title        Ubuntu, kernel 2.6.10-5-686

root          (hd0,1)

kernel         /boot/vmlinuz-2.6.10-5-686 root=/dev/hda2 ro quiet splash

initrd        /boot/initrd.img-2.6.10-5-686

savedefault

boot
```

In the above entry, the root file system is on the first partition of the first hard drive. The kernel is on the /dev/hda2 system, and is mounted read-only, with no “splash” screen. This kernel is installed as vmlinuz-2.6.10-5-686. The initrd line tells the system which ramdisk to load, in this case, initrd.img-2.6.10-5-686 file. The “savedefault” default entry means that the system will load the previously booted kernel at the next boot time. Finally, the “boot” entry instructs the system to boot the kernel.

This file is often a symbolic link to the /boot/grub/grub.conf file because in a Debian system, the file is always named /boot/grub/menu.lst. **The file**

The grub-install command

The grub-install command allows you install GRUB for the first time, or to re-create the GRUB installation in a problem system.

The following messages are common when systems fail to boot:

Grub loading Stage 1.5

Loading Stage 2 . . .

You can use grub-install to solve the problem. When troubleshooting a boot problem, you will have to:

Find a rescue floppy or CD.

Boot the system.

Mount the troubled primary hard drive partition.

Change your system root to the troubled hard drive using the chroot command.

Make backups of any and all files, especially /boot/grub/grub.conf and /boot/grub/menu.lst.

Remove all files except /boot/grub/grub.conf and /boot/grub/menu.lst.

Run the grub-install command.

To learn more about GRUB, go to www.gnu.org/software/grub or http://www.gnu.org/software/grub/manual/grub.html#initrd.

After installation, you can enter an interactive session using the /sbin/grub command. Interactive sessions allow you to permanently modify all grub configuration files and settings to customize the boot process.

LILO

The LILO command is still often used, especially in older systems (e.g., circa 2000 and before). It is still maintained, and is covered on the LPI exams. Unlike GRUB, LILO does not ship with utilities that allow interactive configuration; rather, you edit the lilo.conf file, then issue the /sbin/lilo command to read its contents and update LILO's boot parameters.

Installing LILO

As with any boot manager, you can install LILO during or after installation. You can install LILO either from the source, from an RPM package, or from a Debian package. You then use the /sbin/lilo command to create the LILO map file, which is stored at /boot/map.

Configuring LILO

You configure the LILO boot menu and the kernel(s) to boot by editing the /etc/lilo.conf file. This file has two sections. The first section contains the default settings. The second section actually lists the images and kernels to boot. A sample lilo.conf file follows:

```
prompt
timeout=15
default=linux
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
message=/boot/message
linear

image=/boot/vmlinuz-2.4.18-14
      label=linux
      initrd=/boot/initrd-2.4.18-14.img
      read-only
      append="root=LABEL=/"
```

In the above example, the “prompt” entry has LILO provide a menu. The “timeout” entry tells LILO to wait a certain number of seconds (in this case, 15) before booting the operating system.

Using LILO

Unlike GRUB, once you edit the /etc/lilo.conf file, you must then run the /sbin/lilo command to update the /boot/boot.b file. If you do not, the /boot/boot.b file will not be updated; the changes you made will still exist in the /etc/lilo.conf file, but will not be recognized.

The “default” value tells the Linux system to load a particular kernel (in this case, Linux), unless specifically directed otherwise. If, for example, you were dual booting between Linux and Windows, you could change the “default” setting to either the Windows or Linux image. The choice is yours.

The “boot” entry informs the system which hard disk LILO is installed. The “map” entry, as you might guess, tells where the map file is located, and the “boot” entry tells the system where the boot sector file for the MBR resides. The “message” entry allows you to give a message to anyone who watches the system at boot time. The last entry in this particular lilo.conf file is “linear,” which tells LILO to use a different addressing scheme than the traditional sector/head/cylinder form of addressing. LILO does this translation at boot time. Using the “linear” option sometimes causes problems with boot disks, because the translation may not work from one system to another.

In the second section, the “image” entry allows you to specify and name the kernel you are going to boot, in this case, a very old 2.4 kernel that runs a copy of BIND (a DNS server) that has been running continuously for over 18 months. You will see that the “image” entry has multiple sub-entries. Each is profiled below:

label: Names the kernel.

initrd: Specifies the ramdisk to be first loaded into memory.

read-only: Specifies that LILO should mount the root file system as read-only at first. Usually, the system will later re-mount the root file system as read-write in most cases.

append: Allows you to pass options to the kernel. In this case, the option is applying the label of "/."

Loading other boot loaders

The "chainloader" entry in GRUB allows the operating system to load another boot manager. You will see "chainloader" in Linux installations that dual boot with Windows, for example.

Alternate boot devices, LILO and GRUB

You need not store the boot loader or the MBR on the hard disk. You can boot from various devices, including:

A CDROM

A boot floppy

A USB drive

A FireWire drive.

A resource across a network connection.

To create an emergency boot from a USB device, take the following steps:

Set the computer's BIOS to boot the device you want to use. Suppose, for example, you want to boot a USB drive. If your BIOS does not support booting from a USB disk, then you will not be able to continue with your plans, no matter how well you execute the following steps:

Format the disk with a Linux file system (e.g., ext2, ext3, reiserfs).

Place the boot loader on the disk. You have two choices:

Install the boot loader on the MBR.

Install the boot loader as a file either on the USB disk's root partition, or in the /boot/ directory/partition.

Verify that the root device is set to the drive from which you want to run Linux. Use the chroot command.

If you want to have the USB drive contain its own version of Linux, edit the initrd file so it refers only to your USB device. You must also do the same with the init scripts to refer only to your USB device.

To boot a full-fledged version of Linux, such as Knoppix, you would take the following steps:

Set the computer's BIOS to boot the device you want to use.

Format the disk with a Linux file system (e.g., ext2, ext3, reiserfs). You could use a command such as mke2fs -j /dev/sda1, which would install the ext3 file system on the USB first USB drive.

Use the tune2fs command to ensure that fsck is not run on the device, so the boot process occurs smoothly: tune2fs -c 0 -i 0 /dev/sda1

Mount the ISO file that contains the operating system you wish to use on the USB drive. You can use either the mount or the losetup commands.

Mount the USB drive.

Copy the contents of the ISO file to the USB drive.

Create the /boot/grub directory, if it does not already exist. In some distributions, you need to create the /boot/mnt/grub/ directory instead. Copy the boot information to both of these directories.

Your disk is now bootable. You can repeat the above steps for various device types (e.g., FireWire).

1.102.3 Make and install programs from source

Most modern Linux distributions have their own package managers. Red Hat uses the Red Hat Package Manager (RPM). SuSE/Novell Linux uses YaST, as well as RPM. Debian-based systems use apt, apt-get, dpkg, alien, and GUI-based applications such as Synaptic.

The truly universal installation method is the use of source packages, often created using the tar and gzip, zip, or bzip2 applications. This section discusses the essential things you need to know to compile applications and daemons from source.

Note: You will learn how to use the zip, gzip, and bzip2 commands to create archives in the 102 exam study guide.

Extracting code

When extracting source files, you will need to know how to use several applications. They are described in Table 7.

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
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

Application	Description
	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.

Once you have extracted the contents of the “tarball” or source package file, you can begin to customize the files that allow you to compile the application you want to install. Often, one is not lucky enough to simply issue the “make” command. You must first either edit a special configuration file called the “makefile,” or you must issue a custom set of parameters.

The Makefile

Whenever you compile applications from source, you will use various applications and libraries, including:

The Gnome C compiler: Known as gcc or cc, this application does most of the heavy lifting. The makefile and configure scripts you use will explicitly refer to these applications.

The Gnome CC+ compiler: Known as g++, compiles C++ code.

It is possible, of course, to run these applications and specify all the options and commands yourself. With simple source files that you wish to turn into applications, this is often the case.

However, developers generally supply a special file, called a makefiles, for you, if the application or daemon you want to create is at all sophisticated. A makefile is a set of instructions the compiler uses to compile source files into an executable application or daemon. A makefile can even create multiple applications. Whenever a developer creates source files, he or she considers the “broadest” system that exists. Sometimes – usually, in fact – these broad instructions will work well for you. However, you will frequently have to edit the makefile so that it recognizes the shared library and file locations that are particular to your system.

You may need to make the following changes using a standard text editor:

Path changes: All makefiles contain references to the compiler, various applications, and shared libraries. It is common for one Linux distribution to differ slightly from another in regards to the location of these libraries.

Custom option statements: The makefile may also allow you to "hard code" options in to the actual file.

Include statements: Sometimes, you will need to add statements, which can be written in to the makefile itself.

The make command and compilation sequence

Once you have a makefile that contains the proper instructions, you can use the script to generate the binary programs you want. When using the make command, you generally pursue the following sequence:

make: This command compiles the binaries, though it usually leaves them in the installation directories. You need not be root when executing this command, and really should not be, as it is generally considered bad system etiquette to run as root all the time. To run the "make" command, you must be in the same directory or subdirectory as the makefile.

make install: Places the binaries in the usual directories in the system path so they can be executed. For example, this command will place the application in the /usr/sbin/ or /bin/ directory. You usually need to be root to execute this command.

make clean: An optional command that cleans up all temporary files.

Sometimes, a makefile allows you to run "make uninstall," which removes the entire installation.

Note: Remember, you must be in the same directory or subdirectory as the makefile to run all of the above commands.

The configure command

Most developers provide configure scripts, in addition to standard makefiles. The primary responsibility of a configure script is to automatically probe the system environment and generate the appropriate makefiles. Configure scripts also allow you to manually add parameters yourself, in case the configure script cannot discover the correct environment variables and settings for your situation. To add these parameters, simply add options to the configure script as you would add parameters to a standard Linux command (e.g., ls, dir, ifconfig, ping). You, for example, type the following get help from most configure scripts:

```
./configure --help
```

Note that when using the configure command, you will have to precede it with the ./, to ensure that it will run. Note also that you must be in the same directory as the configure script. You do not need to be root, usually. A configure script is generally a much better way to create a custom makefile than trying to edit one yourself.

Options for the configure script include:

Specifying custom paths for main applications (e.g., gcc) and supporting applications or daemons.

Providing options for your particular distribution or login environment.

Specifying location of shared libraries.

Once you execute the configure script, you can type make, make install, make clean, and other make options.

Source Debian and RPM files

Most users, administrators and programmers are accustomed to having the contents of Debian and RPM files to be ready-made binaries. However, source Debian and RPM files exist. These files simply install source tarballs, which you must then install as shown above.

Debian-based systems install source Debian files in the /opt/ directory. Debian source files often have RPM-based systems install source RPM files in the /usr/src/ directory. Source RPM files often end with ".src.rpm."

1.102.4 Manage shared libraries

Most executable applications refer to shared libraries in order to execute properly. If these libraries are unavailable for any reason, the application may not run at all, or may function in an unexpected manner. Make sure you know how to identify and load these shared libraries. Typically, these shared libraries are found in the following locations:

The /usr/lib/ directory.

The /lib/ library.

The /usr/X11R6/lib/ directory.

The /usr/lib/GNUstep/System/Library/Libraries/ directory.

These libraries usually have a .so ending. They are automatically referred to by applications. The applications must know where these libraries reside.

The ldd command

You can use the ldd command to determine exactly which libraries any executable file needs. For example, view the libraries used by the ls command on an Ubuntu system:

```
user1@host1:~$ ldd /bin/ls
                    librt.so.1 => /lib/tls/i686/cmov/librt.so.1 (0xb7fd2000)
                  libacl.so.1 => /lib/libacl.so.1 (0xb7fcb000)
                          libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7e96000)
             libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0 (0xb7e86000)
              /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0xb7fea000)
             libattr.so.1 => /lib/libattr.so.1 (0xb7e82000)
user1@host1:~$
```

More sophisticated binaries, such as the apache2 daemon, require considerably more libraries, as demonstrated below:

```
user1@host1:~ $ ldd /usr/sbin/apache2
                 libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7fd6000)
        libcrypt.so.1 => /lib/tls/i686/cmov/libcrypt.so.1 (0xb7fa8000)
 libpcre.so.3 => /usr/lib/libpcre.so.3 (0xb7f98000)
 libz.so.1 => /usr/lib/libz.so.1 (0xb7f87000)
              libssl.so.0.9.7 => /usr/lib/i686/cmov/libssl.so.0.9.7 (0xb7f58000)
libcrypto.so.0.9.7 => /usr/lib/i686/cmov/libcrypto.so.0.9.7 (0xb7e61000)        libaprutil-0.so.0 => /usr/lib/libaprutil-0.so.0
(0xb7e4d000)
        libldap.so.2 => /usr/lib/libldap.so.2 (0xb7e1a000)
        liblber.so.2 => /usr/lib/liblber.so.2 (0xb7e0e000)
        libdb-4.2.so => /usr/lib/libdb-4.2.so (0xb7d38000)
        libexpat.so.1 => /usr/lib/libexpat.so.1 (0xb7d18000)
        libapr-0.so.0 => /usr/lib/libapr-0.so.0 (0xb7cf9000)
        librt.so.1 => /lib/tls/i686/cmov/librt.so.1 (0xb7cf2000)
        libm.so.6 => /lib/tls/i686/cmov/libm.so.6 (0xb7ccf000)
        libnsl.so.1 => /lib/tls/i686/cmov/libnsl.so.1 (0xb7cbb000)
        libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0 (0xb7cab000)
        libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7b76000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0xb7fea000)
        libresolv.so.2 => /lib/tls/i686/cmov/libresolv.so.2 (0xb7b64000)
        libsasl2.so.2 => /usr/lib/libsasl2.so.2 (0xb7b4e000)
        libgnutls.so.11 => /usr/lib/libgnutls.so.11 (0xb7aea000)
        libtasn1.so.2 => /usr/lib/libtasn1.so.2 (0xb7ada000)
        libgcrypt.so.11 => /usr/lib/libgcrypt.so.11 (0xb7a90000)
```

```
    libgpg-error.so.0 => /usr/lib/libgpg-error.so.0 (0xb7a8c000)
user1@host1:~$
```

If the ls or apache2 commands were not able to find these libraries, they would not function as expected, or would fail to execute.

The ldconfig command

At times, you will need to install new libraries. You may be required to update the file that informs these applications where the libraries reside. The run-time linker file that allows your application to refer to these libraries is located at /etc/ld.so.cache.

At times, you may need to inform the system about new libraries. To do this, you use the ldconfig command.

The LD_LIBRARY_PATH environment variable

The LD_LIBRARY_PATH environment variable allows you to provide your system with the location of additional shared libraries. You can set it to refer to multiple directories, if you prefer. You perform this setting with a colon. The directories in this path are read first. Therefore, if an improper directory is given, even if you have libraries in the correct directory, you may have a problem. You can check this variable by issuing the following command at a terminal:

```
echo $LD_LIBRARY_PATH
```

1.102.5 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.Table 8 describes the common Debian commands.

Table 8: Debian package management 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.

Debian Package Management Command	***Description***
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.

In the following section, you will learn more about each of the above applications, with an emphasis on apt-get, dpkg, and alien.

Installing, upgrading and uninstalling using apt-get

The apt-get command is the most sophisticated way to manage Debian packages. The /etc/apt/sources.list file usually contains a default set of repositories. It is possible to edit this file and add various repositories, depending on what you want to install. Once you have added the remote repositories, you can issue commands to install applications and daemons.

Following are examples of using apt-get:

To install a package named coolstuff.deb: apt-get install coolstuff.deb

To uninstall everything in the same package, except for the configuration files: apt-get remove coolstuff.deb

To uninstall the entire package, including its configuration files: apt-get --purge remove coolstuff.deb

To upgrade all installed packages: apt-get upgrade

To upgrade an entire Linux distribution to the latest: apt-get -u dist-upgrade

Note: All these commands must be executed as root.

One of the primary benefits of apt-get is that it will automatically discover dependencies, then install them. This feature is different from RPM, which will simply notify you of a failure, then quit.

The /etc/apt/sources.list file

You can edit this file using any text editor. Following is a sample entry:

```
deb-src http://security.ubuntu.com/ubuntu hoary-security main restricted
```

You need not know the specifics of this file to pass the LPI exam. However, it is important that you know that this file can be edited to include new repositories.

Installing, upgrading and uninstalling using dpkg

The dpkg command is not as sophisticated as apt-get. It will not install dependencies as easily, and it is best for installing local files. However, it must be used when you cannot get a package using apt-get, or when you want to list information about installed packages. It is also useful when you want to list the contents of a package before installing it. Below are the commands to install a package using the dpkg command:

To install a package named “coolstuff”: dpkg -i coolstuff.deb

To uninstall the same package, except for the configuration files: dpkg -r coolstuff.deb

To uninstall all elements in the package: dpkg -r --purge coolstuff.deb

To list the contents of a package: dpkg -c coolstuff

To list information about an installed package: dpkg -l coolstuff

Find packages containing specific files or libraries, which may or may not be installed.

Obtain package information such as version, content, dependencies, package integrity and installation status (whether or not the package is installed).

The /var/lib/dpkg/ directory contains various files, including /var/lib/dpkg/status, which dpkg uses to remember what has been installed. If this file is missing, apt-get will not work properly. You can recover this file from the /var/lib/dpkg/status-old file, or from the /var/backups/dpkg.status.* file.

Alien

The alien command allows you to convert packages from other managers to Debian packages. It can convert the following packages:

RPM

Slackware (.tgz)

Solaris (pkg)

For example, suppose you have the coolstuff.rpm file, and you want to install it in your Debian system. You would simply issue the following command:

```
# alien coolstuff.rpm
```

Alien will then convert the file into a new file with a .deb ending. You can then install it using dpkg. You can also use alien to convert Debian packages to RPM and other formats, if you prefer.

1.102.6 Use Red Hat Package Manager (RPM)

Understanding how to use RPM is as vital as understanding package management in Debian systems. The RPM utility has become quite popular, because it is found in Red Hat, Mandriva, and even Suse/Novell systems. RPM has the same capabilities as dpkg. You can install local and remote files using RPM, and you can also determine the state of installed files.

RPM allows you to install, manage, and uninstall RPM packages. To create packages, you need to create a “spec” file. The spec file is simply a text file that contains all the instructions that RPM needs to compile individual files into an RPM file. You also need to assemble all the relevant files into a directory. You do not need to know anything more than this for the LPI 101 exam.

You do, however, need to know how to use RPM to install and manage files. Below are some examples of common RPM commands:

To install a file named coolstuff.rpm, with verbose output and showing hashes to denote installation progress: rpm -ivh coolstuff.rpm

To remove the same file: rpm -e coolstuff.rpm

To list the files contained in a file named coolstuff.rpm: rpm -l coolstuff.rpm

To list all files in the package, as well as the package description: rpm -qpil coolstruff.rpm

To list all packages required by the package you want to install: rpm -R coolstuff.rpm

To upgrade an RPM package: rpm -U coolstuff.rpm

To discover what package an installed file comes from: rpm -qf /sbin/coolstuff

To determine if a particular package is installed: rpm -qa |grep *packagename*, where *packagename* is the name of the particular package you are searching for.

To ignore dependency problems and install the file: rpm --nodeps coolstuff.rpm

To verify the signature on an RPM file you want to install: rpm --checksig coolstuff.rpm

An RPM file is named using the following convention:

name-version-release.architecture.rpm

For example, the following file is for a Red Hat Linux RPM that installs Xine, a popular multimedia video player:

xine-0.99.4-5.fc5.x86_64.rpm

Notice how the name of the application is given, then the release version (0.99.4-5-fc5), then the architecture (64-bit x86), and then the file name ending (.rpm).

Additional RPM files

The following are additional files you need to know for the LPI 101 exam.

/etc/rpmrc: Allows you to customize how RPM runs. Usually needs to be created by default. Contains instructions that allow you to customize how RPM runs. You can also edit this file if you want to create RPM packages yourself.

/usr/lib/rpm/: Contains sub-directories and files that allow you to create RPM packages, as well as scripts and files necessary for RPM to run properly.

Topic: 103 GNU & Unix Commands

1.103.1 Work on the command line

No one can be called a Linux professional unless he or she is fully proficient in command-line navigation and understands how to use the login environment. Working at the command line means that you understand the various shells that you can use in Linux, as well as how to use variables, determine and modify the path, execute commands inside and outside the path, and invoke recursive commands.

Shell overview

A "shell" is a login environment that accepts commands and runs applications. Various types of shells exist, including:

The standard shell (sh): One of the earlier shells.

The bash shell (bash): Called the Bourne Again Shell (bash), because it was created by Stephen Bourne. The standard shell in most Linux systems.

The korn shell (ksh): Created by David Korn. Often regarded as a good shell for programmers.

The c shell (csh): Another example of a shell that has strong programming features. Introduced the concept of aliases, in which one command is automatically mapped to run another command. For example, if you execute the "ls" command, an alias can be created that actually runs "ls -l" automatically instead. The "ls -l" command is an alias to ls, in this case.

Each of these shells has standard commands and an associated language. Use these commands and the associated language to create shell scripts. These scripts are simply applications that allow you to automate procedures. Programmers and systems administrators frequently use shell scripts to automate repetitive tasks (e.g., finding files, running applications with various options).

Common shell features include:

Login files: Executable files that contain instructions. File names and functions differ from shell to shell. The bash shell uses the .bashrc file to create aliases, determine the look and feel of the terminal prompt, and set color options for the terminal (e.g., files with .rpm endings are in red, files that are executable are in green). The .bash_profile file allows you to set variables, determine the umask (the default permissions for a file), and run the .bashrc file.

History: The ability for the shell to “remember” past commands that have been executed.

Command-line editing: The ability to not only remember past commands, but allow modifications to be made to the past command in order to create new commands.

I/O redirection: A function that allows the resulting output of a command to be “piped” into a new command as an argument. With I/O redirection, you can use the shell to combine two or more commands.

Regular expressions: A string of commands, or set of strings, that are used to modify the results of a command. Examples of regular expressions include the “wildcard” (*), which is used to match any and all characters, and standard brackets (e.g., [and]), which are used to match any character found within those brackets.

Working with variables

A variable is the term used to describe data stored in memory under a name. Make sure that you understand the following about variables:

How to set a variable: A variable can be set by simply typing the name of the variable, an equal sign, and the value of the new variable. The syntax is as follows:
variable_name=value

For example, to create a variable named MEANINGOFLIFE, and assign it a value of 42, you would issue the following command:

MEANINGOFLIFE=42

How to export a variable: Once you have created a variable, it exists only in the existing shell's memory. In order for this variable to be used by other users and shells, you must then **export the variable using the export command. The syntax for using the command is as follows:**

export VARIABLE_NAME. For example, you would issue the following command to export the MEANINGOFLIFE variable, you would issue the following command:
export MEANINGOFLIFE

How to read a variable: Use the echo command and the dollar sign ($). To read a variable, precede it with a dollar sign. To read the MEANINGOFLIFE variable at a terminal, you would issue the following command: echo $MEANINGOFLIFE

The path

The path is an environment variable set at boot time. It can be modified by a login file during user login. The path can be viewed by issuing the following command:

```
echo $PATH
```

Following is an example of a typical path for a non-root account in an Ubuntu Linux system:

```
user1@albion:~$ echo $PATH
/usr/local/bin:/usr/local/sbin:/sbin:/usr/sbin:/bin:/usr/bin:/usr/bin/X11:/usr/games
user1@albion:
```

You can add to the path variable, if you want to. For example, you may need to create a custom directory that is not in the path. You can do this by using the following command:

```
PATH=$PATH:/newdir/newapp/
export PATH
```

Notice that when you set the path, you do not just add the /newdir/newapp/ directory. Doing so would eliminate the existing path, reducing it to /enwdir/newapp. As a result, you would not be able to issue any commands from, say, the /bin/, /usr/sbin/, or /sbin/ directories, because they would no longer be in your path.

The path must be preceded with the $PATH designation, then with any additions. If you wanted to add two additional directories, you could issue the following command:

```
$PATH:/newdir/newapp/:/newusers/userapps/
```

Remember to export the variable so that all users and applications can refer to it.

Important commands for viewing and modifying the path include:

pwd: Prints the directory your shell is currently accessing. It stands for “print work directory.”

set: Used to create and modify variables. Especially useful for modifying system path. Can also be called the “setenv” command. Know what set does when executed without arguments (prints the set of existing environment variables), and what it does with arguments.

unset: Deletes variables.

env: Used with arguments, allows you to create a custom environment for a particular command. Used without arguments, prints out all settings concerning the login environment. Make sure you can read all output from env, and understand how you can affect how a program runs by allowing it to execute in its own temporary, virtual environment.

exec: Closes the existing shell (and associated terminal) and executes a command in the place of the shell that executed it. Older systems used the exec command as part of a logout script to save memory and log out more quickly. It is most often used to quickly change from one shell to another. Consider the following sequence:

```
user1@host1:~$ exec sh

sh-3.00$
```

Notice how the shell has changed. It changed because you used the exec command to exit cleanly from the old “bash” shell and then entered into the new standard shell (sh).

Issuing commands outside the path

Sometimes, you will have installed or compiled a binary that does not exist in your path. Three options are available:

Modify the system path.

Precede the command with a ./, without a space. For example, if you need to execute a file named newapp, you would issue the following command:

```
./newapp
```

The effect of this command is that it will run in the same shell you have just opened.

Use a “dot command,” which loads a new subshell to the existing shell.

Consider the following as you take the exam:

If you try to run a command that is not in the system path, you will receive a message that says, “bash: *command_name*: command not found.” You will not receive a message that says “permission denied.” The former message simply states that the command was not found inside the system path. The latter implies that the command was found, but that you do not have sufficient permissions to execute it.

ecursive commands

Make sure you understand the following concepts:

Directory tree: A series of subdirectories beneath a parent directory. Viewable by issuing the ls -lR command.

Use a “dot command,” which loads a new subshell to the existing shell.

You can issue commands so they are processed through an entire directory structure. For example, listing the /etc/ directory using the ls command without any options will simply list the files and directories. Using the -R or --recursive options actually goes through an entire subdirectory off of the /etc/ directory.

1.103.2 Process text streams using filters

Many commands create text output onto the terminal screen. The output on this screen is of two types:

Standard output: Results from a command when it is executed successfully, and returns data it is designed to return. If you want to specify standard output to redirect, you must use the number “2.”

Standard error: Information sent by the application when it encounters a problem. If you want to specify a standard error to redirect, you must use the number “2.”

“Standard input” is the name for what you type to create a command.

Sometimes, all you need to do is run the command and view the results on the screen. At other times, you will need to take the output from this screen and manipulate it. The LPI exam will spend a considerable amount of time testing your ability to send the output of various commands through filters. Table 9 discusses several of these commands.

Table 9: Text stream filters

Command	***Description***
wc	Counts the number of words found in a text file.
nl	Counts the number of lines in a text file.
expand	Processes a text file and converts tabs to single spaces. Useful when you have a script that is ready to read spaces, as opposed to tabs.
unexpand	Does the opposite of expand: It converts single spaces into tabs.
fmt	Can be used to align margins in text files, split and join text lines, and create new columns of any width you need.
hexdump	Reads contents of standard output or of a text or binary file and issues output in hexadecimal format.
cut	Prints only a part of a particular file. You can cut any percentage of the file you want.
paste	Used to merge lines in a file.
sed	A sophisticated command that allows you to edit the stream of characters and filter, substitute or insert strings. You can use sed to remove certain text strings or characters. You can also use sed to select certain fields in a file, such as /etc/passwd, so that you get only the user information that you want.
split	Used to divide a file into smaller files.
join	Used to create a single file out of multiple files.
sort	Combines (i.e., concatenates) lines from multiple files and sends them to standard output or a file.
tr	Allows you to delete or substitute characters in a document and put the new text out to standard output.
uniq	Short for “unique,” this application allows you to remove duplicate lines in a file.
pr	Adds pagination and a title to an existing document.

Make sure you understand that any of the commands listed in these tables can be linked together in sophisticated quasi-applications. For example, you can send the output of any of the above commands as an argument for a subsequent command using the “pipe” regular expression (known as the | character), or you can direct the output into a text file using the >.

To list all files, but then list only files that have the text string “secure,” use: ls | grep secure

To list all files, then place them into a file named secure.txt, use: ls > secure.txt

Table 10 discusses additional commands that are used to read the contents of text files.

Table 10: File-sorting commands

Command	***Description***
cat	Reads the contents of a file and directs the contents to standard output.
tac	A reverse version of cat. Reads the contents of a file, nd then reverses the order, so the standard output lists the file from bottom to top.
head	Reads the first 10 lines of a file. Make sure you understand the options to this command.
tail	Reads the last 10 lines of a file. Again, make sure you understand the options to this command.

Note: The commands in Tables 9 and 10 are part of the GNU textutils package.

1.103.3 Perform basic file management

Linux professionals are able to easily navigate the command line; they do not rely on GUI-based tools. To be competent at the command line, you must be able to copy, move, and delete files and directories.

Make sure you understand how to use the commands profiled in Table 11.

Table 11: Command line navigation commands

Command	***Description***
cp	Copies files. Learn all options, including: -R: Copies files recursively -u: Copies only newer files if the name is the same.
find	Used to find files and directories. A sophisticated command. Make sure you understand the command's sequence: find path options For example, to search the entire hard disk for a file named important.txt, you would issue the following command: find / -name important.txt The above command shows how you specify the path, and then specify a name. You can search for partial names, as well as files by creation and modification date.
mkdir	Creates an empty directory.
mv	Moves files. Unless you use the -i option, the file you are moving will replace any existing file of the same name. Many systems automatically alias the mv command to mv -i. Additional important options include: -u: Moves file only if source file is newer. -f: Forces automatic overwrites.
ls	Lists all files and directories. Carefully learn all options and various options and option combinations, including: -l: Provides a “long listing” for a directory or file, which includes permissions, owner, inode, the latest time and day the file or directory was modified, and file size. -h: Provides the file size in kilobytes and megabytes. -lda: When used on a directory, shows all information about that directory, as opposed to its contents.
rm	Removes a directory or file. Will not delete a directory. Important options and combinations include: -fr: Forces a recursive delete, and it will not ask for confirmation. Can also use -R or –recursive. -i: Requires confirmation for deletion. A common option for aliasing.
rmdir	Removes an empty directory.

Command	***Description***
touch	Allows you to change the “timestamp” of a file. A timestamp includes the file modification time. Will also create a standard text file, if no file by that name exists.

Additional concepts to understand in regards to navigating the command line and managing files include:

Wildcard commands: The wildcard character (e.g., *) can be used with the above commands.

File globbing: Expanding abbreviated file names into complete file names.

Using recursion: You can see how the -r option for the rm command allows you to go through an entire directory tree.

Experiment with each of these concepts before taking the exam.

1.103.4 Use streams, pipes and redirects

Table 12 shows essential stream redirection commands.

Table 12: Redirecting streams

Command	***Description***
>	Allows you to direct a text stream to a text file: ls -l > file_list.txt. The file named file_list.txt will be created, and will contain the standard output resulting from the ls command. If file_list.txt already exists, all existing contents will be replaced with the output from the ls command. Using the /dev/null file, you can redirect the standard output and standard error so that it is automatically deleted: ls -l > /dev/null 2>&1
>>	Similar to the single arrow, it will redirect the standard output to a file. However, redirected output will be appended to the file. In the following command, any text inside of file_list.txt will not be replaced. The new output from the ls command will simply be added to the end: ls -l >> file_list.txt Also known as the “append operator.”
<<	Used to begin a session that allows you to create input for a file.
<	Reads the contents of a file and then applies the contents as an argument to a command: sort < file_list.txt

Command	***Description***
|	Known as the "pipe." Used to combine multiple commands. Pipes the output of a command to the input of another command. The following command runs the ls command, then pipes the output to the less command, which allows you to page through the listing using the space bar: ls -l | less
`	Known as the "backtick character" or "grave accent." This is the key below the tilde key (~) on standard U.S.-based keyboards. Used to substitute separate standard output from one command into a line of text that helps create a new command.
tee	Reads information from standard input and then writes the output to standard output, or to another file. As a result, you will be able to see the standard output of the command, and also have it as a file: ps aux | tee -a ps_output.txt
xargs	Prepares standard output to be used as an argument or arguments to other commands: To list all files and place them into one line: ls | xargs echo To move all of the files from dir1 to dir2, which allows you to show each command executed: ls dir1 | xargs -i -t mv dir1/ dir2/

Consider the following tips about redirection:

Understand the difference between standard quotation marks, single quotation marks, and the backtick character. If you enclose an expression in backticks, you tell the shell to assign the result of a Linux command to a variable, instead of printing that command to the screen

Practice using redirect characters thoroughly before the exam.

1.103.5 Create, monitor and kill processes

When an application loads into memory, it is called as a process. Below are ways you can manage processes so they run exactly as you want.

Starting a process that will run without being associated to a terminal

You can run any command you want from a terminal. If you just type the command and press enter, the process will remain "attached" to the terminal. You will no longer have access to the terminal. However, if you add the ampersand (&) after the command you type, the program will detach itself from the terminal, giving you access to the prompt. Using the ampersand after a command works as follows:

```
user1@albion:~$ethereal &
```

This method is called running a job in the background.

Running a command in the foreground

To run the command so it is back in the terminal, use the fg command:

```
fg ethereal
```

Now, the command will run again in the foreground. To stop a process from running, use the Ctrl z keystroke combination. For example, to stop the program named ethereal, you would issue the following command:

```
Ctrl z ethereal
```

To see if any processes are running in the background, issue the jobs command:

```
jobs
```

If any background jobs are running, you will be informed. For example, if ethereal is running in the background, you would see the following:

```
[1]+ ethereal &
```

You can then issue the fg command, or kill the program, as discussed in the following section.

Running programs after logout

Because you issue commands during a login session, standard operating procedure is for all processes to be terminated before final logout occurs. Preceding a command you want to run with the nohup command causes the program to continue running, even after logout:

```
user1@albion:~$nohup find / -name '\*.txt
```

Now, the find command will remain running after the user logs out.

Monitoring and sorting processes

Make sure you understand the following applications to monitor the processes that are running on your system:

ps: Shows current processes that are running. Has many different options. Following is an example of the ps command, showing all processes on the system in a “tree” format, so you can see which processes have spawned other processes:

```
ps axjf
```

It is important to understand that options to ps can be specified in three different ways. Options specified UNIX-style are preceded with a dash. BSD-style options are not preceded with a dash. GNU-style options are preceded with two dashes (e.g., --). Familiarize yourself with each of the option styles.

top: A menu-based application. The processes that occupy the most memory are listed at the top of the menu, hence the name of the application. Using top, you can change execution priority, kill processes, and sort through various menus.

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.

1.103.6 Modify process execution priorities

When a process is launched from a terminal or by any other means, it has a default priority of 0. The possible priority range is from -20, which is the highest priority, down to 19, the lowest priority. Following is a brief overview:

Lowest priority: 19

Default priority: 0

Highest priority: -20

As you can see, you have a total range of 40 priorities. Make sure you understand that a negative number actually means a higher priority, which is somewhat counter-intuitive.

Changing program priority at runtime

The nice program can be used to run a process with a particular priority. For example, to run the ethereal program with the highest priority, you would issue the following command:

```
user1@albion:~$ nice -20 ethereal
```

As a result of this command, ethereal will receive higher priority than any other process.

It is important to understand that you can use the nice command only when starting a process. If you want to change the priority of an application, you need to use the renice application.

Changing priority of a process that is already running

The renice application must be run as root, and is capable of changing the priority of a running process.   The syntax for using renice is as follows:

```
renice priority -p pid
```

The steps for “re-nicing” a process are as follows:

Discover the PID of the process you want to re-nice.

Use the renice command as root, specifying the new priority, and the PID of the process you want to renice.

Following is an example where ethereal's process priority is changed from -20 to 10:

```
user1@albion:~$ ps aux | grep ether
user1     9133  0.0  1.7  28844 18552 pts/1    S+   09:51   0:00 ethereal
user1@albion:~$ sudo renice 10 -p 9133
9133: old priority -20, new priority 10
```

You can also change the process priority using the top command. To do so, run top, then press the “r” key. You can specify the PID of the process that you want to renice.

For the exam, you will also need to know the following about process priority:

Using kill or killall to list processes that your login account is allowed to kill (kill -l, or kill -L).

Killing multiple processes with one command (e.g., kill 42 43 44 45 46).

Telling killall to not return a terminal prompt until it verifies that a process is killed.

1.103.7 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.

Regular expressions

A regular expression is string of instructions designed to accomplish specific tasks. You can combine multiple regular expressions. Table 13 describes some of the regular expressions used in sed.

Table 13: Common regular expressions

Expression	***Description***
\t	Denotes a tab. You can use this expression to either add or search for tabs in a stream.
s/old/new/g	Replace the "old" text with the "new."
-n	
G	Add a line of blank space.
/d	Delete a line or character.

Sed examples

Below are some examples that will help you understand sed's capabilities.

To use sed in a "find and replace" manner, finding the phrase Windows and replacing it with Linux:

```
    sed 's/Windows/Linux/g' originaldoc.txt >newdoc.txt.
```

To print every other line of the /etc/passwd file:

```
sed '1~2!d' /etc/passwd
```

To change a text file named paper.txt from single space to double space, and a file named lines.txt to triple space:

```
sed G paper.txt
sed 'G;G' /etc/passwd
```

Delete all lines in a file named paper.txt except for lines 41 and 42:

```
sed '1,2!d' /etc/passwd
```

It is also possible to use a text file that contains sed scripts. You can specify them using the -f option.

1.103.8 Perform basic file editing operations using vi

Vi is a sophisticated text editor. Although applications such as OpenOffice.Org Writer and Microsoft Word appear more capable, a person skilled in vi and XML can create and edit business-ready documents. You are expected to know basic vi modes and commands. Table 14 describes common vi commands you will be expected to know.

Note: In Linux, vi is actually an open source program called "vim." It is designed to act exactly like the original version of vi.

Vi is a "bimodal" editor, meaning that it operates in two modes. The modes are:

Command: You can issue commands, such as move the cursor, open files, delete lines, copy and paste, and save changes.

Insertion: You can type in text and delete characters.

Vi opens in command mode. In this mode, you can move freely around the document, and enter various commands, which are discussed below. You exit command mode by pressing the "i" key. After pressing the "i" key, you can then move your cursor around the document and enter text. You can leave the edit mode by pressing the escape key (ESC).

Vi is bimodal because it was created in a time when connections where slow and would not support sophisticated GUI menus. Also, the bimodal approach allowed easier support for various terminals and keyboards. Table 14 common vi commands you can issue in command mode.

Table 14: Common vi command-mode commands

Command	***Description***
j	Move cursor down one line.
k	Move cursor up one line.
gg	Jump to top of document.
G	Jump to first character on a line.
H	Move to top of screen. Note: This command does not necessarily move your cursor to the top of the document, just the current screen.
u	Undo a change.
l	Move cursor forward one character.
c	Cut text.

Command	***Description***
p	Paste text.
d, forward arrow key	Deletes a character.
a	Enter insert mode.
next! foo.txt	Opens a new document named foo.txt.
e	Move cursor forward one word.
:bnext	Move from one buffer to the next.
yy	Copy text.
p	Paste text.
/	Allows you to search for any word. Type the "/" character, then press ENTER.
o	Enter insert mode and enter a new blank line below the cursor.
:!	Allows you to execute any external command. For example, :!bash gives you a bash shell. Simply type "exit" and press ENTER to return to vi. The !ls command lists files in the current directory.
:history	View history of commands you have executed for the current vi session.
dd	Deletes a full line of text.
dw	Deletes next word after the cursor.
d$	Delete everything to the end of the line.
ZZ	Save changes and exit the file currently loaded into the vi buffer.
:w!	Save changes. The file/buffer remains open.
:wq	Save (i.e., "write") changes and exit the file.
:q!	Quit the file without saving.

You will need to learn additional commands and become quite proficient with vi. Do not focus on other text editors (e.g., emacs or pico).

Make sure to carefully read through the vi/vim documentation in the /usr/share/doc/vim/ directory, and study the main pages. As always with the LPI exam, hands-on, practical field experience will always be beneficial. Also, the vimtutor application provides an excellent overview.

Focus on learning:

Moving the cursor around in a document.

Opening multiple windows.

Opening a terminal from within vi.

Creating custom startup options using the vimrc file, such as enabling line numbering and color options. The file is often located at /etc/vim/vimrc.

Recovering from a vi "crash" and working with the "swap" file.

Copying and pasting text.

Topic 104: Devices, Linux Filesystems, Filesystem Hierarchy Standard

1.104.1 Create partitions and file systems

You have already learned how to create a disk layout. You must also know how to create partitions, maintain partitions, and create a file system on these partitions. One of the first things to understand about modern Linux file systems is that they operate efficiently by storing essential data in special files called journals. These journals make it possible for systems to store information more efficiently, write and retrieve data faster, and recover quickly from errors.

During installation of a system or the creation of a new partition, you will need to choose a file system. Table 15 shows the most common file system types.

Table 15: Common Linux file systems

File system	***Description***
ext2	The most common early file system. Still used for floppy disks, as journaling file systems have difficulty fitting on a floppy. Not a journaling file system.
ext3	The most common journaling file system. Based on ext2.
reiserfs	The first mainstream journaling file system. Instead of working with all data, it considers only recent file activity stored in special "meta data" files that describe the system.
jfs	Uses metadata, like reiserfs. A journaling file system. Funded by IBM.

Once you have decided on the file system to use, you can then use the mkfs command to create a file system on the partition layout you have created. Table 16 describes the essential applications for creating partitions.

Table 16: Partition creating commands

Command	***Description***
fdisk	Creates system partitions. Can be used as a one-time command, or as an interactive command.
mkswap	Converts swap space on a partition.
mkfs and mke2fs	Creates a file system. Many systems will have multiple mkfs files, each with a unique ending that indicates the file system it will create. For example: mkfs.ext3 mkfs.reiserfs mkfs.ext2 You can also use the mke2fs command to create the ext2 and ext3 file system.

Using fdisk

You can use fdisk as either a stand-alone or an interactive command. To enter interactive mode, simply type fdisk, followed by a valid partition on the hard drive as root. In the following case, you would use fdisk to begin viewing and editing the first installed SCSI disk:

```
fdisk /dev/sda
```

Once you have entered interactive mode, you can issue various commands to view the existing partition table or create new partitions. Interactive-mode fdisk commands include:

p: Prints the partition table.

n: Create a new partition.

d: Delete an existing partition.

q: Quit the session, and do not make any changes to the partition table.

w: Write changes to the partition table, then exit the session.

Make sure you understand at least the following about fdisk:

Listing existing partitions in interactive mode, and as a one-time command: fdisk -l, and p (in interactive mode).

Creating new partitions in interactive mode.

Erasing the MBR

If you need to remove the master boot record for a system, you can issue the following command, as root:

```
fdisk /mbr
```

Understand, however, that issuing this command will make it impossible to access any existing data on that system. Also, remember that using fdisk on an existing partition will destroy all data on that partition, including the file system.

Using mkfs

The mkfs command has the ability to create the ext2 and ext3 file systems. If you wanted to create the ext3 file system for a partition named /dev/hda3, you would issue the following command, as root:

```
mke2fs -j /dev/hda3
```

or, you could issue the following command:

```
mkfs.ext2 -j /dev/hda3
```

The -j option tells mkfs.ext2 to create journals, rather than simply create the ext2 file system.

Using mkswap

The syntax for using mkswap is as follows:

```
mkswap -c partition_name  partition_size
```

So, should you want to create a 2 GB partition on the /dev/sdb3 partition, you would issue the following command:

```
mkswap -c /dev/sdb3 2
```

You would then issue the swapon command to activate the new swap space:

```
swapon /dev/sdb3
```

The above command will create all the partition swap space. For the LPI 101 exam, you do not need to worry about using less of the partition; it is assumed that you will create a dedicated partition for swap space.

You can, of course, disable the swap file:

```
swapoff /dev/sdb3
```

Note: You need to be root to execute the above commands.

1.104.2 Maintain the integrity of filesystems

Now that you know how to create a partition scheme and install a file system, you need to know how to maintain it properly to avoid problems.

Disk maintenance commands

Table 17 gives an overview of the most essential commands you will use when working with hard disks.

Table 17: Disk maintenance commands

Command	***Description***
du	Reports how much disk space a particular directory is using. “du” is short for Disk Usage. Make sure that you know about all the command options, including -h.
df	Reports the total amount of disk space that is consumed by all partitions. Again, know all your options, especially -h.
fsck	Used to scan disks. Looks for problems with a file system, then tries to correct them. The file name of fsck is actually a front end for multiple file checkers, as there are multiple file systems used in Linux (e.g., ext3, reiserfs, jfs).
e2fsck	Like fsck, but specifically designed for the ext2 and ext3 file systems.
debugfs	A file system debugger that allows you to review file system activity, including deleted inodes and files.
dumpe2fs	Provides detailed file system information for the ext2 and ext3 file systems. Includes superblock information.
tune2fs	Customizes parameters for the ext2 or ext3 file system.

Verifying the integrity of file systems

When it comes time to use the fsck command, remember the following:

Fsck is run each time the system boots.

Unmount the partition before you use fsck. Otherwise, you could corrupt the file system on the partition and lose your data.

Run fsck against partitions.

You can edit the /etc/fstab file to determine whether or not fsck is run automatically at boot time. If a “0” is placed in the 6th column, called the “pass” column, fsck will not run. If a 1 is entered, fsck will be automatically run.

If fsck finds any problems, recovered data will be placed in a file called lost+found. This file is stored on the mount point for a file system.

One of the ways you verify file system integrity is to use the dumpe2fs file system. To use it, you must specify a partition you want to use:

```
sudo dumpe2fs /dev/hda2
Filesystem volume name:   /
Last mounted on:          <not available>
Filesystem UUID:          503a4d9c-01ed-4952-86c6-5ac922c99e2a
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal filetype needs_recovery sparse_super
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              3129344
Block count:              6249285
Reserved block count:     312464
Free blocks:              563233
Free inodes:              2933253
First block:              0
Block size:               4096
Fragment size:            4096
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         16384
Inode blocks per group:   512
Last mount time:          Wed Jun 14 17:51:04 2006
Last write time:          Wed Jun 14 17:51:04 2006
Mount count:              8
Maximum mount count:      30
Last checked:             Tue Jun 13 02:25:25 2006
Check interval:           0 (<none>)
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Journal inode:            8
First orphan inode:       704857
Journal backup:           inode blocks
```

As you can see, you can obtain valuable information concerning any ext2 or ext3 file system available. If you use dumpe2fs on a non-ext2 or ext3 file system, you will be informed that the superblock and "magic number" (the information that specifies the superblock) are bad. Following is a typical sequence that occurs when you specify the wrong file system:

```
$ dumpe2fs

dumpe2fs 1.35 (28-Feb-2004)

dumpe2fs: Bad magic number in super-block while trying to open /dev/hda1

Could not find valid filesystem superblock.

$
```

Necessary skills

As you study for this part of the exam, make sure you understand how to:

Monitor free space and inodes using df and du.

Repair simple file system problems using fsck.

Use tune2fs to review existing settings, and (as root) configure options, including:

Adjusting maximum counts between file system checks (-c).

Changing the amount of times a system thinks it has booted (-C).

Listing superblock information (-l).

1.104.3 Control mounting and unmounting filesystems

The LPI 101 exam expects you to know how to:

Mount and unmount the most common file system types on attached and removable devices (e.g., ext2, ext3, vfat, ntfs, iso9660, nfs).

Edit the /etc/fstab file to customize how devices are mounted at boot time, as well as by users.

To properly mount files and configure systems, you will need to know the following commands:

/etc/fstab: Provides instructions for the system concerning the partitions to mount. It is read at boot time. It is also referenced by the mount command when you give instructions to mount a CD ROM or removable device (USB). Make sure you can read each of the fields in the file.

mount: Used to mount a partition. You can use it to specify a particular file system.

umount: Unmounts a partition.

The /etc/fstab file

Each line in the fstab file contains instructions on how to mount a partition. Each line is composed of six fields, discussed in Table 18. A simple example of an fstab file is as follows:

```
# <file system> <mount point>   <type>  <options>      <dump>  <pass>
proc           /proc          proc    defaults       0       0
/dev/hda2       /              ext3    defaults,errors=remount-ro 0       1
/dev/hda5       none           swap    sw             0       0
/dev/hdc        /media/cdrom0   udf,iso9660 ro,user,noauto  0       0
/dev/hda1       /mnt/windows    ntfs defaults,noauto 0 0
```

Table 18: /etc/fstab fields

Field	***Description***
File system	The partition that will be mounted (e.g., /dev/sda1).
Mount point	The directory name for the partition that will be mounted (e.g., /home/.
Type	The file system type. Can be ext2, ext3, reiserfs, iso9660, ntfs, swap, /proc, or any other file system type supported by Linux.
Options	Provides custom mounting options (e.g., read-only, automatic mounting, only certain users).
Dump	Used by the dump command and other backup applications to determine which partition is backed up, and which is not.
Pass	Determines whether or not the partition will be automatically scanned by fsck at boot time.

In regards to the /etc/fstab file, make sure you know how to:

Understand the different types of file systems to be mounted:

iso9660: CD ROM

ntfs: Windows NT/XP/2003/Vista file system (usually only mounted read-only).

Swap: For swap files.

Proc: For the /proc virtual file system, dynamically generated by all Linux systems.

udf: Universal file system for CD-ROM based media.

The /etc/fstab versus the /etc/mtab file

Understand that the /etc/fstab file does not describe the file systems that are currently mounted. The /etc/mtab file give you this information.

Below is an example of a simple mtab file:

```
/dev/hda2 / ext3 rw,errors=remount-ro 0 0
proc /proc proc rw 0 0
sysfs /sys sysfs rw 0 0
devpts /dev/pts devpts rw,gid=5,mode=620 0 0
tmpfs /dev/shm tmpfs rw 0 0
/dev /.dev unknown rw,bind 0 0
none /dev tmpfs rw,size=5M,mode=0755 0 0
usbfs /proc/bus/usb usbfs rw 0 0
```

Note: It is important to understand that this file reflects the current status of the system; the /etc/fstab file contains all of the files that can be mounted. The /etc/mtab file lists all of the file systems that ar e curerntly mounted.

1.104.4 Managing disk quota

A disk quota determines how much disk space a particular user can consume. A user can be a “real person,” or can be a system service or daemon (e.g., Apache server, PostgreSQL, or MySQL). Quotas are imposed on a partition-by-partition basis.

Table 19: Quota commands

Command	***Description***
quota	Displays all the existing quota information for a particular user. Root can check the quotas of other users by using the -u option.
edquota	Sets the quota for a particular user.
repquota	Provides a summary for a particular partition.
quotaon	Activates quotas for a particular system.
quotaoff	Deactivates quotas for a particular system.

To establish a disk quota for a file system, you first need to edit the /etc/fstab file to recognize quotas. The system will need to reboot so it can recognize these changes. You have two choices when doing quotas:

usrquota: Establishes user-based quotas.

grpquota: Establishes group-based quotas.

You can establish user-based and group-based quotas, or just one of the two. Following is a line from the /etc/fstab directory that has user and group quotas enabled:

```
/dev/hda2       /               ext3    defaults,usrquota,grpquota,errors=remount-ro   0       1
```

Once you have changed the /etc/fstab file, you then create the quota.group and quota.group files for the partition you want to protect with quotas:

```
touch /home/quota.user
touch /home/quota.group
        chmod 600 /partition/quota.user
        chmod 600 /partition/quota.group
```

Now, use the quotaon command to begin quotas on the system. Then, you can use the edquota command to establish a quota and the repquota command to learn about the state of the quotas operational on the system.

Make sure you understand the following:

Soft limit: If users exceed this limit, they are warned and given a grace period (a period of time) to delete files and get beneath the quota.

Hard limit: An absolute limit that cannot be exceeded. Only in force, if a grace period has been set, and if a user exceeds that grace period.

Note: To set quotas, you generally need to be root.

1.104.5 Use file permissions to control access to files

The LPI 101 exam requires you to know the following about file permissions:

Reading numerical/octal permissions (e.g., 511 for a file.

Reading “symbolic” permissions (e.g., r-x--x--x).

Changing permissions using both numerical and symbolic values.

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.

You will not be asked to provide abstract information about ACLs. However, you will be asked to understand the ways in which Linux systems use various commands to implement its ACL policies. When it comes to the exam, focus on controlling file and directory access by user. Table 120 discusses the commands you will need to understand.

Table 20: File permission commands

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.

Understanding permissions

Consider the output from a standard directory listing:

```
$ ls -l
-rw-r--r--   1 user1 user1     382 2006-06-13 08:02 success.txt
drwxr-xr-x  13 user1 user1    4096 2005-12-30 23:05 novell
```

The first listing is for a file. You can tell, because the name "success.txt" is listed, because in Linux, you can give a directory a .txt name ending, if you prefer. You can also tell because of the preceding – at the beginning of the listing. If you were viewing a directory, you would see a d, as you see in the second example.

Permissions are noted in a linux file or directory as follows:

directory/file designation | user permissions | group permissions | everyone permissions

It is important to understand user, group and everyone (or, “world” as some people call it) permissions. The permissions for user specify what the owner of the file can do with that file. The second field specifies what people in a group can do if their group is given ownership of the file. The third group applies to any user on the system. It is possible to give full permission to the user and partial permissions to the group and everyone. It is common to give full permissions to the user, but little or no access to user groups or to everyone.

Each permissions group is composed of three possible values:

r: Read permission. Has a value of 4.

w: Write permission. Has a value of 2.

x: Execute permission. Has a value of 1.

These values can be added numerically. Go back to the examples given above. The file named success.txt has w-r for the first value, **r-- for the second value, and r--** for the third. The shorthand for the permissions on this file are 511. For the directory, the permissions are 755.

Permissions and groups

You can use the group field to grant file access to groups of users and workers. Make sure you know how to:

Read group-based permissions.

Understand permissions in a multi-user environment.

Limit access to files in a directory to a particular group.

Troubleshoot file permission problems.

Changing permissions

These permissions can be changed using the chmod command. For example, to change the permissions for the file so the owner of the file has full permissions, but groups and everyone has no permissions, you can issue either of the following commands:

```
chmod 700  success.txt
```

```
chmod u=rwx,go=  success.txt
```

Make sure you understand how to manipulate permissions for both files and directories.

Special permissions

You already know the standard permissions for a file (user, group, and everyone). Make sure you understand how to apply the following permissions:

suid (Set User ID): When applied to a file, allows normal users to execute a file and gain the permissions of the owner of the file. Many times, the owner of this file is root. At times, security problems can be introduced by suid files, especially if a powerful command (e.g., su or a shell) is marked suid. The result would be that any user could act as root on the system. When applied to a directory, suid permissions do not mean the same thing, and do not have security implications. Rather, suid permission on a directory mean that any subsequently created file or directory will be owned by the owner of the directory, rather than by the user who created the file or directory. **You can set the suid and full permission for the user and no permissions for anyone else on the file success.txt as follows:**

```
chmod u+rwxs,g=,o= success.txt
chmod 4700 success.txt
```

The resulting permissions would read as follows:

```
-rws------  1 user1 user1 0 2006-06-16 00:04 test
```

Notice the “s” in the space that normally specifies execute permission for the user. This means that the file is set for suid.

sgid: This permission is set on a file or application. The resulting process is owned by the group that owns the file, rather than by the user that executed the application. When you list an sgid bit on a file, the “s” bit is set as a capital S in the execute portion of the group permissions, rather than in the user portion. You can also set the sgid bit on a directory, which means that all files created in this directory are owned by the owner of the directory, effectively overriding any other user groups. Specify this bit as follows:

```
$ chmod 2700 success.txt
chmod u+rwx,g=s,o=x success.txt
$ ls -l
-rwx--S---  1 user1 user1 0 2006-06-16 00:04 test
```

The “sticky bit:” Set on a directory, the sticky bit means that files created in this directory can be deleted only by their owners, the owner of the directory, or root. Set the sticky bit as follows:

```
$ chmod 1700 allfiles
chmod u+rwx,g=o=t allfiles
$ ls -l
drwx-----T 1 user1 user1 0 2006-06-16 00:04 allfiles
```

Now, the allfiles directory has the sticky bit set. The capital “T” would be a lower-case “t” if the execute permission bit was set.

Note: The permissions explained above can be combined.

Make sure you understand the practical implications of the above commands and concepts. For example, setting suid permission on the wrong file can cause serious security problems. Making a directory sgid can help ensure consistent permissions in a directory. Setting the “sticky bit” in a directory can help improve security, because users in a directory cannot cause problems by accidentally deleting each other's files.

Additional considerations

Be sure to remember the following for the exam:

The chattr command is capable of setting additional permissions beyond those set above. The two most important options to chattr are discussed in Table 20. Make sure you understand why a file might refuse to be deleted, even by root.

Make sure you understand the purpose of the umask command. Be able to:

Read the information reported by the umask command.

Set a new umask for your account.

1.104.6 Manage file ownership

A file or directory must have an owner to work properly in a standard Linux system. You can set ownership using the commands listed in Table 21.

Table 21: File ownership commands

Command	***Description***
chown	Changes the owner of a file.
chgrp	Changes the group owner for a file.

For the LPI exam, you must be able to:

Read the output of the ls -l or ls -lh command and determine a file's owner.

Use the chown command to change the owner of a file or directory.

Use the chgrp command to change the group that owns a file or directory.

Recursively change ownership of files and subdirectories in a directory.

Apply these settings in situation where access to files in a directory must be limited to the proper group.

The chown command

The chown command syntax is as follows:

```
chown owner filename
```

For example, suppose that you have an account on a Linux system named user2. This means that any file you create will be owned by user2. If you want to change the ownership of this file to user1, you would issue the following command:

```
# chown user1 lpi.txt
```

Note: You must be root to change ownership on a file.

You can use the -R option to recursively change ownership on a set of files or subdirectories.

The chgrp command

Using the chgrp command is similar to chown, except that it applies to groups, rather than users. It has the same syntax as chown. The -R option remains the same. You must still be root to use chgrp, as well.

1.104.7 Create and change hard and symbolic links

A symbolic link is effectively the same as a "shortcut" in a Windows system. A symbolic link is a special file that simply refers to another, real file. It contains no data of its own. It simply tells the system, "Go over to a different location to find the file you want." The symbolic file is an additional entry in the file name table. A symbolic link can also refer to another symbolic link. Symbolic links are more common than hard links in most systems.

With a hard link, you create a new file that refers directly to the original source. The hard link is no mere pointer or shortcut. The hard link you create refers directly back to the disk block.

Using the ln command

The ln command is used to create both hard and soft links. Below is an example of creating a hard link:

```
ln original.txt newhlink.txt
```

The file named newhlink.txt now refers to the same content as original.txt.

To create a soft link, use the -s option to ln, as shown below:

```
ln original.txt newsoftlink.txt
```

Now, you have a reference to the original.txt. However, this new file does not refer directly to the data. It refers to the file name of the original file.

Comparing link types

A symbolic link can possibly slow down a system, because it is simply a reference to the "real" file. Because the system has to jump from one file name (the symbolic link) to the "real" file, and then to the resource the file name is referring to, the system can be slowed. This slowing is especially true in a "real-time" situation, such as when a symbolic file is used for video or audio playback.

Hard links eliminate slowdown, but they can also consume space. Hard links cannot be made between directories; they must be to a file in the same directory. Soft links do not have this limitation. Also, soft links can be made across different file systems (e.g., from an ext3 partition to a reiserfs partition), whereas hard links cannot.

Be able to:

Create a hard link.

Create a soft link.

Explain what happens when a hard link is deleted.

Explain what happens when a soft link is deleted.

Read the output of the ls -l command and identify how many soft links exist to a file.

1.104.8 Find system files and place files in the correct location

In the following section, you will learn about the importance of understanding the FHS to the LPI exam, as well as the essential commands to know when it comes to finding files on a partition.

The Filesystem Hierarchy Standard (FHS)

The FHS is meant to provide a consistent guide to vendors, programmers, people who develop RPM and Debian packages, and systems administrators. The FHS describes the appropriate directory location for system files.

For example, according to the FHS, X11 configuration files (the files for the X Window system that provides a GUI interface) belong in the /etc/X11/ directory. As a result, developers and others follow this standard voluntarily. You can learn more about the FHS and download the latest guide at www.pathname.com/fhs.

Applied FHS

The LPI exam will expect you to understand basic locations recommended by the FHS in an applied situation. It will not expect you to understand the following:

The purpose of the root file system (/, not the /root directory). The root file system has three purposes: To boot the system, to allow the system to be repaired, and to allow files to be recovered. For the LPI exam, understand that it is easy for new Linux users to confuse the phrase “root directory” with “root file system.” The “root directory” is the home directory of the root user, usually, /root. The root file system is known as /, and contains all other directories.

What files will be found in the /boot/ directory: Only files directly related to booting the system and allowing the boot manager to run should be included.

The files found in the /bin/ directory (e.g., essential binaries that users execute to administer the system).

The contents of the /proc/ directory.

The purpose of the /etc/ directory (configuration files).

The purpose for the /home/ directory, and why it is usually best if this directory is best placed on a separate partition (to enable faster recovery, troubleshooting, and to allow for the possibility of remotely mounting the partition in case of a problem.

The standard use of the /var/ directory, which includes storage of log files, as well as other volatile files, such as those used by Apache and various FTP daemons.

Finally, the FHS helps determine which directories are generally sharable, and those which should not be shared. The FHS recommends, for example, that the following directories can be shared:

/home

/usr

opt

/var/mail

/var/spool/news

However, the following directories should not be shared:

/etc/

/boot/

/var/run/

/var/lock

Generally, the FHS recommends that directories that hold essential configuration information should not be shared. Directories that contain binary commands or user files can be shared, though of course standard UNIX-based permissions should always remain in place.

Finding files

Table 22 describes the essential file finding commands you should know for the LPI exam.

Table 22: File ownership commands

Command	***Description***
find	Used to find the location of files on any partition. Allows the use of regular expressions and many options. Will search all files and directories, regardless if they are in the system path.
locate	Focuses on finding binary commands and supporting documentation in the system path.
slocate	A more secure version of locate, which allows you to control which files and directories will be found.
updatedb	Used to create or update the database that locate or slocate uses. Must be run each time the /etc/updatedb.conf file is edited. If run without arguments, simply processes the files in the path and re-creates the database.
/etc/updatedb.conf	The configuration file for the slocate database.
which	Allows you to locate a command that can be executed in the current environment. . Like locate and slocate, it searches the system path.
whereis	Like slocate, searches the path for binary commands and supporting documentation. However, it does not have a configurable database.
man	Although not a true file finding command, it is an essential command that allows you to learn about what files and commands do on the system. Relies on the makewhatis command.
makewhatis	Used to rebuild the databases used by man. When you update these databases, you can conduct more efficient keyword-based searches using the man command. Similar to updatedb. You simply type “makewhatis” (without the quotation marks) as root to use the command. Additional options apply, but they are not relevant to the LPI exam.

Using the find command

The find command uses the following syntax:

```
find path expression
```

The path can include a single directory, the system root, and other locations. Expressions can take many forms. You can use wildcards, system names, and other means to find exactly what you want.

For example, to find all files on the entire hard disk that have a .txt ending, you would issue the following command:

```
find / -name \*.txt -print
```

The -print option is can be omitted, but is traditional. It simply informs the find command to report what it discovers to standard output. You can also search for files with specific timestamps, and other attributes.

Below are several find commands:

Find a file that has google entered in it, but only in the /user/ directory:

```
find /usr -name *\google\* -type f -print
```

In the above example, notice how the wild card characters (*) were used. These make sure that the fine command looks for any instance of the text string “google” in the file name, even if the text string is in the middle of a file name, at the beginning, or at the end. The forward slash characters (/) are used to escape the shell. Otherwise, the wild cards would be interpreted by your shell (e.g., the bash shell in most instances), rather than by the find command. The type -f option searches for regular files, as opposed to block and character files.

If you wanted to search the entire hard disk, you would replace /usr/ with the/ character:

```
find / -name *\google\* -type f -print
```

You are not limited to searching for file names. You can also search for file permissions. The find command given below searches for files that have the ending of .php, and which match the permissions of 777:

```
find htdocs /var/html/php/ -name "*.php" -type f -exec 'ls -l' {} \; | grep 777
```

In the above command, you had to execute the -ls command to list the files that are found. Also, notice how you had to use the {} \; sequence. This sequence informs find that the execution portion of the command has ended, and to escape the shell.

If you wanted to find all files that ended in .php, then change their permissions to 755, you would issue the following command:

```
find htdocs /var/html/php/ -name "*.php" -type f -exec chmod 755 {} \;
```

It is possible to execute various commands using the find command. Suppose, for example, that you wished to find a type of file, then delete it. You can use the -ok rm options to search for the file matching your pattern. These options are useful, because they will prompt you to confirm if you wish the deletion to take place:

```
find ~user1/public/ -name '*.jpg' -print -ok rm {} \;
< rm ... /home/user1/bigfile1.jpg > ? y
/home/user1/bigfile1.jpg
< rm ... /home/user1/bigfile2.tiff  > ? y
```

If you do not want to have find ask you to confirm each file deletion, you can issue the following command:

```
find ~user1/public/ -name '*.tiff' -print -exec rm {} \;
```

You can use find to discover files or directories that are larger than a certain size. For example, to find any file that was larger than 56 MB, you would issue the following command:

```
 find . -size +57000k -type f
```

It is true that many of the find commands are counter-intuitive. However, the find command makes up for its complexity by being extremely flexible and capable.

Below are some guidelines to consider when using find:

Whenever you decide to execute a command using find, you must use the following sequence: {} \;. You would do well to memorize this fact.

You do not have to use the -print option at the end of the command.

When executing commands to remove files, make sure that you either use confirmation, or that you are very sure of the nature of your command. Otherwise, you will join the large ranks of unhappy people who have deleted essential configuration files and even directories by executing what seemed like an innocuous command.

Additional commands

Take the time to practice using additional commands. Some sample commands and their results are listed below:

```
$ slocate ifconfig
/usr/share/man/man8/ifconfig.8.gz
/sbin/ifconfig
```

```
$ whereis ifconfig
ifconfig: /sbin/ifconfig /usr/share/man/man8/ifconfig.8.gz
```

```
$ locate ifconfig
/usr/share/man/man8/ifconfig.8.gz
/sbin/ifconfig
```

If you wish to conduct a keyword search through the existing man page database, use the -k option to the man command. For example, the following command conducts a search for any instance of the word “belgium” throughout the database:

```
man -k belgium
```

Configuring the slocate database

You must edit the /etc/updatedb.conf file to configure the slocate database. Below is an example of the updatedb.conf file:

```
# This file sets environment variables which are used by updatedb

# filesystems which are pruned from updatedb database
PRUNEFS="NFS nfs afs proc smbfs autofs iso9660 ncpfs coda devpts ftpfs devfs mfs sysfs"
export PRUNEFS
# paths which are pruned from updatedb database
PRUNEPATHS="/tmp /usr/tmp /var/tmp /afs /amd /alex /var/spool /sfs"
export PRUNEPATHS
# netpaths which are added
NETPATHS=""
export NETPATHS
# run find as this user
LOCALUSER="nobody"
export LOCALUSER
# cron.daily/find: run at this priority -- higher number means lower priority
# (this is relative to the default which cron sets, which is usually +5)
NICE=10
export NICE
```

Notice in the above file that you can:

Use the PRUNEFS to remove any file system. In other words, you can ensure that any file system, such as reiserfs, is excluded from a search.

Use the PRUNEPATHS directory to remove entire directories/partitions.

Use the NEPATHS variable to make sure that directories exported using NFS are excluded.

Set the user that slocate and updatedb use to create the databases. The user "nobody" is standard; using the root user or any other "real" user is considered too dangerous in terms of security, because if anything were to go wrong with the updatedb and slocate commands, it would be possible for a root shell to be left behind, causing a security breach.

Set the priority at which slocate and udpatedb will run. The standard level is 10 for these commands. The value of 10 also happens to be the standard for most processes on a Linux system. If you were to set the value at -10, the process would have higher priority than just 10. The highest niceness priority is -20. The lowest is 19. It is advised that you keep the NICE value at 10 for this file.

What is the difference?

As usual, there is more than one way to accomplish the same task when it comes to finding files in Linux. Make sure that you understand the subtle differences between each command. For example:

The whereis command does not allow you to configure the database of files that will be searched. The which command does everything whereis does, but has the additional benefit of allowing you to exclude or include various directories.

The locate command and the slocate command are very similar. The slocate command allows you to exert greater control over the database that users can search. It is considered more secure, as its name indicates: Secure locate. Systems administrators often use slocate to limit an end user's ability to learn about your system. Still, it is important to understand that many times, the slocate command gets aliased as the locate command in many newer systems.

The find command is the most sophisticated command of the bunch, and is therefore the most difficult one to use. Make sure that you learn the examples given above, and that you also practice using it. You will likely be asked to find a certain file by the exam. The exam will give you various parameters for finding a file, then ask you to write out the proper find command that solves the problem. For example, you will be told that you need write out a find command that finds any file created between two and three days ago on the /var/ partition.

1.110.1 Install & Configure X11

The X Window system has existed for more than 20 years. The two major providers of the X Window GUI are X.org (www.x.org) and XFree86 (www.xfree86.org). The LPI exam requires knowledge of both.

The X Window system uses the X11 protocol. As a result, using X involves you in a client-server relationship. Even when you begin an X Window session on a local system, your session is acting as a client to a server. This server must provide:

A login environment.

Fonts.

Mouse support.

A Window manager, which provides the look and feel of the environment (title bars, color schemes, the ability to tab between applications, focus schemes). Example window managers include Enlightenment, Sawfish, WindowManager, Window Maker, and Blackbox.

Note: It is possible for your X Window system to support remote clients, as well.

Many different X Window environments exist. Two of the most popular include:

Gnome.

KDE

These environments have their own default window managers, which can be changed if you prefer.

Using X.org

The list of relevant files commands used to install and configure the X Window system include:

xorgcfg: A GUI-based tool for the X.org system. This tool allows you to probe the system's sound card and monitor and establish the desired settings for your X Window implementation.

xorgconfig: Creates an xorg.conf file, which controls the system's GUI behavior.

/etc/X11/xorg.conf: Provides instructions for all input and output devices. Each section of this file is dedicated to a specific function or device, including the fonts used, the input devices (keyboard, mouse, touchpad), the video card, and the monitor.

Configuring X.org

The LPI exam focuses mostly on the /etc/xorg.conf file. Remember the following as you configure this file:

Each section is dedicated to a specific purpose, including:

Files: Specifies locations for the fonts used by the X server.

Module: Provides a listing of the types of kernel modules used by the X server.

InputDevice: Can specify any one of the following three devices:

Keyboard.

Mouse

Touchpad

Device: Specifies the video card (e.g., an ATI Radeon Mobility 9600/9700)

Monitor: Lists the monitor being used.

Screen: The resolution that the monitor will use. If multiple resolutions are entered, a default resolution will be specified, and the end user can then choose various resolutions by issuing various key combinations, discussed in Table 23, below.

ServerLayout: Tells the X server which resources to provide to the display.

DRI: An optional section. Provides options for the Direct Rendering Infrastructure (DRI), which is used to enhance video display on some X.org systems.

You can edit the file using any standard text editor (e.g., vi, emacs, pico, xedit). You can even use a word processor such as OpenOffice.org, but make sure that you save the file as a standard ASCII text file, and not in a word processing format (e.g., OpenDocument, Microsoft Word).

A sample xorg.conf file

```
Section "Files"
    FontPath    "unix/:7100"            # local font server
    # if the local font server has problems, we can fall back on these
    FontPath    "/usr/lib/X11/fonts/misc"
    FontPath    "/usr/lib/X11/fonts/cyrillic"
    FontPath    "/usr/lib/X11/fonts/100dpi/:unscaled"
    FontPath    "/usr/lib/X11/fonts/75dpi/:unscaled"
    FontPath    "/usr/lib/X11/fonts/Type1"
    FontPath    "/usr/lib/X11/fonts/CID"
    FontPath    "/usr/lib/X11/fonts/100dpi"
    FontPath    "/usr/lib/X11/fonts/75dpi"
    # paths to defoma fonts
    FontPath    "/var/lib/defoma/x-ttcidfont-conf.d/dirs/TrueType"
    FontPath    "/var/lib/defoma/x-ttcidfont-conf.d/dirs/CID"
EndSection

Section "Module"
    Load  "bitmap"
    Load  "dbe"
    Load  "ddc"
    Load  "dri"
    Load  "extmod"
    Load  "freetype"
    Load  "glx"
    Load  "int10"
    Load  "record"
    Load  "type1"
    Load  "vbe"
EndSection
```

```
Section "InputDevice"
        Identifier      "Generic Keyboard"
        Driver          "keyboard"
        Option          "CoreKeyboard"
        Option          "XkbRules"      "xorg"
        Option          "XkbModel"      "pc104"
        Option          "XkbLayout"     "us"
EndSection

Section "InputDevice"
        Identifier      "Configured Mouse"
        Driver          "mouse"
        Option          "CorePointer"
        Option          "Device"                "/dev/input/mice"
        Option          "Protocol"              "ImPS/2"
        Option          "Emulate3Buttons"       "true"
        Option          "ZAxisMapping"          "4 5"
#       Option          "MaxTapTime"            "0"
#       Option          "SHMConfig"             "on"

EndSection

Section "InputDevice"
        Identifier      "Synaptics Touchpad"
        Driver          "synaptics"
        Option          "SendCoreEvents"        "true"
        Option          "Device"                "/dev/psaux"
        Option          "Protocol"              "auto-dev"
        Option          "HorizScrollDelta"      "0"
#       Option          "SHMConfig"             "on"

EndSection



Section "Device"
        Identifier      "ATI Technologies, Inc. Radeon Mobility 9600/9700 M10/M11 (RV350 NP)"
```

```
#       Driver          "ati"
        Driver          "fglrx"
        BusID           "PCI:1:0:0"
EndSection

Section "Monitor"
        Identifier      "Generic Monitor"
        Option          "DPMS"
EndSection

Section "Screen"
        Identifier      "Default Screen"
        Device          "ATI Technologies, Inc. Radeon Mobility 9600/9700 M10/M11 (RV350 NP)"
        Monitor         "Generic Monitor"
        DefaultDepth    24
        SubSection "Display"
                Depth           1
                Modes           "1440x900"
        EndSubSection
        SubSection "Display"
                Depth           4
                Modes           "1440x900"
        EndSubSection
        SubSection "Display"
                Depth           8
                Modes           "1440x900"
        EndSubSection
        SubSection "Display"
                Depth           15
                Modes           "1440x900"
        EndSubSection
        SubSection "Display"
                Depth           16
                Modes           "1440x900"
                Modes           "1440x900"
        EndSubSection
        SubSection "Display"
```

```
                Depth           24
                Modes           "1440x900"
        EndSubSection
EndSection

Section "ServerLayout"
        Identifier      "Default Layout"
        Screen          "Default Screen"
        InputDevice     "Generic Keyboard"
        InputDevice     "Configured Mouse"
        InputDevice     "Synaptics Touchpad"
EndSection

Section "DRI"
        Mode    0666
EndSection
```

For the LPI exam, it is recommended that you:

Study the xorg.conf man page.

Configure a few options in the xorg.conf file. For example, enable and disable the mouse. Remember that when you make changes to the file, you will have to restart the X server completely. A simple restart will not make the changes active. To restart the system, kill the entire X subsystem.

Using XFree86

The XFree86 project is currently less popular, but you still need to know how to use it, mainly because it still has a large installed base from older Red Hat systems. Following is a quick discussion of the most relevant applications:

XF86Setup: The standard graphical configuration utility. Probes your system and finds all the input and output devices (keyboard, mouse, video card and monitor). It then gives you choices. While it will make recommendations, you can also choose your own settings. You can also use XF86Setup to modify existing settings.

xf86config: A text-based configuration application. Useful if troubleshooting a monitor that even XF86Setup cannot work with.

xvidtune: Allows you to fine-tune your XF86 configuration. Alterations include:

Making the screen wider.

Making the screen more narrow.

Adjusting the screen so that it is shorter or taller.

Moving the entire screen left, right, up and down.

/etc/X11/XF86Config: The primary configuration file for your X Window implementation. As with xorg.conf, it is divided into sections (e.g., keyboard, mouse, video card, display).

The .Xresources file: A hidden file in a user's home directory that allows the system administrator to customize how XFree86 works at login. You can include any command-line option to the X command you want to automatically run, including:

The display number to use: The default display is 0, which is the first display, because X uses a zero-based count, like many Linux applications.

Access control settings: You can require various types of authentication, from standard to Kerberos.

Geometry settings: You can change the left and right, bottom and top settings for the screen.

Make sure you know how to do the following:

Verify that a video card and monitor can work with your X server: Issue the following command:

For XFree386: sudo X :3 -scanpci

For X.org: sudo Xorg :3 -scanpci

Specifying a remote server: You can connect to remote X Window systems using the X command. Following are two examples:

X :1 -query gandalf.company.com: Begins an X Window session on the second display that leads to a login screen on the remote host. The first display in this case would be busy **providing your local X Window display**. On the second display, your local system would act as a client to the system named gandalf.company.com. You would be able to log in and access resources, just as if you were sitting in front of the system.

X :1 -indirect gandalf.company.com: Begins an X Window session, but instead of receiving a login screen, you receive a listing of hosts that are willing to accept remote X Window connections.

Generic X commands

Once you have entered an X session, it is important to know various key combinations that are generic from one version of X to another. In other words, they will work in Xorg and XFree86. Table 23 lists these combinations.

Table 23: Common X Window key combinations

Key combination	***Description***
Ctrl, Alt , Backspace	Restarts the X server.
Ctrl, Alt, + (on the numeric keypad)	Cycles through the available resolution settings, starting from existing resolution to higher resolution. If you keep pressing this sequence long enough, you will begin again at the lowest resolution and proceed to the highest again.
Ctrl, Alt, - (on the numeric keypad)	Cycles through available resolution settings, starting from the existing resolution to the lowest (the exact opposite of Ctrl, Alt, +).
Ctrl, Alt F1-F6	Switches between text terminals (one that doesn't show an X session). If you press Ctrl, Alt F2, you will be given a second text terminal on the system. Usually, systems boot showing the first terminal by default; press Ctrl, Alt F1 to return to it.
Ctrl, Alt F12	Switches betwee X window terminals. If you press Ctrl, Alt F8, you will be shown a second X display. Usually, this display is empty, because the default X display is found in the first terminal, available by pressing Ctrl, Alt F1.

The above commands are only operative once an X window session is active; they will not work at a command line. If these commands do not work, it is most likely that they have been disabled in the XF86Config file.

Installing and configuring an X font server.

A font server by default whenever any X Window system is installed. However, you can install a dedicated font server. Using a remote font server can help create consistent login environments from one system to the next. Remote font servers also make it possible for clients to save disk space.

Installing a font server involves the following steps:

Obtain the font server software. Most of the time, it is bundled in with the X Window software. A font server may have to be installed, such as xfs (www.xfree86.org).

Compile (if necessary) and install.

Edit your X Window configuration file (e.g., xorg.conf) to recognize the font server.

Start the server using the initialization scripts (e.g., /etc/init.d/xfs start).

You can specify either a local or a remote font server. When troubleshooting problems font server problems, remember the following:

Check the FontPath setting in the XF86Config or xorg.conf file.

Make sure that the specified font server is on the network. If the remote system is down, contact the system administrator. If the remote system is up, make sure that the correct name is specified in your configuration file.

Use the netstat command to make sure that your local font server (if it is being used) is active. The font server usually listens on the xdcmp port, which is UDP port 177.

Restart the font server, regardless of whether it is a local or remote server.

Restart your X server.

Make sure that your system can access the network.

Manually editing the X Window configuration file

Sometimes you cannot just use xorgcfg or XF86Config to obtain the right settings. You will need to know how to edit each text file using a standard text editor, preferably vi.

For example, to use a remote font server, you will have to specify it in the X Window configuration file. **Notice the following section, either your** /etc/X11/xorg.conf or /etc/X11/XF86config file:

```
Section "Files"
     FontPath      "unix/:7100"            # local font server
```

You simply change the above entry to a remote font server. The syntax for the change is as follows:

```
transport/hostname:port
```

You want to use the same transport, but now also want to specify a remote host name. Suppose that the remote font server is called fs1.company.com: You would make the following change in your /etc/XF86 or /etc/X11/xorg.conf file:

```
     FontPath      "unix/fs1:7100"            # remote font server
```

When editing this file, you can use any standard text editor.

Note: The numbering scheme for this objective changes, due to an internal LPI issue. You can verify that this topic is part of the LPI 101 exam by going to the following URL: http://www.lpi.org/en/obj_101.html.

1.110.2 Setup a display manager

A display manager is what appears when you log on to a local or remote X Window system. The display manager will provide:

A login screen.

Information about the system (e.g, the host name, the IP address, a user policy).

Sometimes, a listing of other systems that have display managers running.

When configuring a display manager, you will be asked to demonstrate how to:

Activate and deactivate the display manager.

Customize the environment, including changing the display manager greeting, and the default color depth.

Configure a display manager.

Activating and deactivating a display manager

You have two choices when activating and deactivating a display manager. First, you can use a supplied GUI application. Second, you can edit the text files directly. Often, you will have to take the second choice. When editing the text files directly, make sure you:

Verify the correct display manager that you will use: Many systems will have multiple display managers installed. Look for the appropriate Gnome Display Manager (gdm) or KDE display manager (kdm) directories.

Make backups of the text files before editing.

Use comments to retain old settings, even if you do use backups.

To deactivate a display manager, find the text files and comment out the settings that open ports. You may also need to edit the text to add instructions, such as display = no. Files and directories to consider include:

/etc/X11/xdm/*

/etc/X11/kdm/*

/etc/X11/gdm/*

Finally, remember the /etc/inittab file. This file will help you determine whether or not a system boots into the X Window system or not by default at system startup. Learn the contents of this file carefully, especially the initdefault line:

id:5:initdefault:

In the above line, init level 5 will be the default level, which for many Linux systems implies both a display manager and a local X Window login screen.

Customizing the environment

When customizing your environment, use the scripts associated with the login manager. Make sure you familiarize yourself with how to:

Change the display manager greeting.

Change the default color depth to one that can be viewed by systems with various capabilities. The lower the depth, the more compatible your display.

1.110.4 Install & Customize a Window Manager Environment

A window manager is a part of the overall X Window system.

Table 24 describes the most essential files to know.

Table 24: Display manager files and settings

Command	***Description***
.xinitrc	Automatically runs applications at login.
.Xdefaults	Determines the look and feel of the current session.
xhost	Used to allow or disallow logins on the local system. Make sure you understand the following: xhost + xhost -
DISPLAY environment variable	Provides information about the current display. If you type $DISPLAY and press ENTER at a command prompt, you will see the current display (e.g., 0:0 for the first display).

When customizing settings, consider the following:

The ability to edit X Window configuration files, such as .xinitrc, so that terminals automatically appear (e.g., xterm, rxvt, aterm).

Configuring a default terminal.

Use the /var/log/messages and /var/log/syslog files to determine the cause of problems with X Window display. You can frequently learn that a particular dependency has not been met. If this is the case, you will have to edit the relevant files in the /etc/X11/ directory to solve the problem.

Working in remote X sessions

When working with remote X sessions, make sure you understand how to export a session. You do this by logging on to the remote system and typing the following:

```
$ export DISPLAY=mycomputer.company.com:0.0
```

You can then start the application you want to use.

The application will then connect to your local system, and the remote application will display to your local screen.

It is also important to conduct secure X sessions. Ways to do this include:

Tunneling the X session in an SSH session.

Using SSL.

Using SSH is the best option. Make sure that you understand that you must first establish the SSH session, then begin the X Window session.

Tunneling X connections using SSH

SSH encrypts data packets automatically. With additional configuration, you can configure SSH to authenticate using public keys. However, public key authentication is beyond the scope of the LPI 101 exam,. If you are using OpenSSH version 3.7 and below, you would issue the following command:

```
ssh -X host1.company.com -l username
```

The -X option tells SSH to tunnel (i.e., encrypt) any X Window applications that you run on the remote connection. The -l option allows you to specify the user name you wish to use as you log on to the remote computer. If you don't specify this user name, the ssh command will automatically apply the user name you are currently logged in under.

If you are using a more modern version of SSH, you would issue the following command:

```
ssh -Y host1.company.com -l username
```

The -Y option is necessary for newer versions of X, which enable an authentication mechanism. This mechanism cannot be negotiated by older SSH clients. The newer SSH clients allow you to use the older -X option, but also the -Y option.

Configuring the SSH server to forward X sessions

You need to make sure that the remote system is configured to tunnel X sessions. Usually, they are not configured to do so. As root, you have to edit the /etc/ssh/sshd_config file. You would then alter its contents so that it contains the following lines:

```
X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes
```

If these lines are commented out, each line will be preceded by a hash mark (#). Remove the comment. You will have to restart the SSH daemon so that it re-reads this configuration file and enables X Window forwarding.

Remember, SSH is a client-server protocol, as with most that you will use in a modern network environment. As a result, you need to make sure that the SSH server on your local system is configured to forward X connections. As root, edit the client configuration file, called /etc/ssh/ssh_config file, and make sure that the following lines are not commented out:

```
ForwardAgent yes
ForwardX11 yes
```

Once these lines are uncommented (if necessary), restart the SSH daemon. You will then be able to forward connections between your system and the remote system. For the remote system to do the same thing to your system, you would have to repeat the same process for the server and client SSH configuration files.