INTRODUCTION

The Internet Lab is a miniature version of the Internet. Instead of millions of hosts and routers, the Internet Lab has only four routers, four personal computers (PCs), and a few Ethernet hubs. Still, the equipment in the Internet Lab is sufficient to study the protocols of the Internet. By keeping the number of devices in the Internet Lab small, it is feasible to complete non-trivial lab exercises in a reasonable amount of time.

This introduction provides an overview of the hardware and the software environment of the Internet Lab. In Section 1 we describe the different network devices and cables that will be used throughout the Internet Lab manual. All network exercises in the Internet Lab are run from computer systems that run the Linux operating system. In Section 2, we provide an overview of Linux, and discuss basic commands that are used throughout the Internet Lab manual. Section 3 discusses the traffic analysis tools *tcpdump* and *ethereal*. These tools are used to capture and display network traffic, and are essential for studying network protocols. Section 4 provides an overview of the Internet Operating System (IOS), the operating system of Cisco routers, and the covers basic features of the command language used to configure Cisco routers.

Version 6c, May 16th, 2003

Deleted: March

-updated numbering of Figures 28, 31

TABLE OF CONTENTS

Deleted:

1. Overview of the Internet Lab Hardware

The Internet Lab consists of a set of routers, PCs, and Ethernet hubs. The setup of the Internet Lab equipment should be similar to that in Figure 1. The Internet Lab is completely isolated from the Internet. By not connecting the Internet Lab to an operational network, the Internet Lab manual can ask you to perform tasks that would otherwise cause significant disruptions, such as unplugging network cables, or would raise security concerns, such as capturing and studying network traffic. In a real network, the majority of tasks that you perform in the lab exercises are restricted to network engineers and administrators.

1.1. Description of the Hardware

The Internet Lab has four PCs, which are labeled as *PC1, PC2, PC3*, and *PC4*. All PCs have the Linux Red Hat 7.3 operating system or a later version of Red Hat Linux installed. Each PC has a floppy drive, a serial port, and two 10/100 Mbps Ethernet interface cards (*NICs*). The back of each PC is similar as shown in Figure 2. Each PC has ports to connect a keyboard, a mouse, a monitor, a parallel port, one or more serial ports, and ports to connect audio devices. In addition, each PC has two network interface cards with Ethernet ports. In Figure 2, the network interface cards are labeled as *eth0* and *eth1*. The serial ports, assuming that the PC has two such ports, are labeled as *ttyS0* and *ttyS1*. These labels refer to the names that the Linux operating system uses to identify the Ethernet cards or serial ports. For example, when you assign an IP address to an Ethernet interface card in Linux, you need to specify the name of the interface.

The PCs are controlled from a single KVM (Keyboard-Video-Mouse) switch, which connects a keyboard, monitor, and mouse to the PCs. The KVM switch gives you control over all four PCs from single keyboard, monitor and mouse. The KVM switch has buttons that select to which PC the keyboard, monitor, and mouse are connected to. With the KVM switch, you can access only one PC at a time.

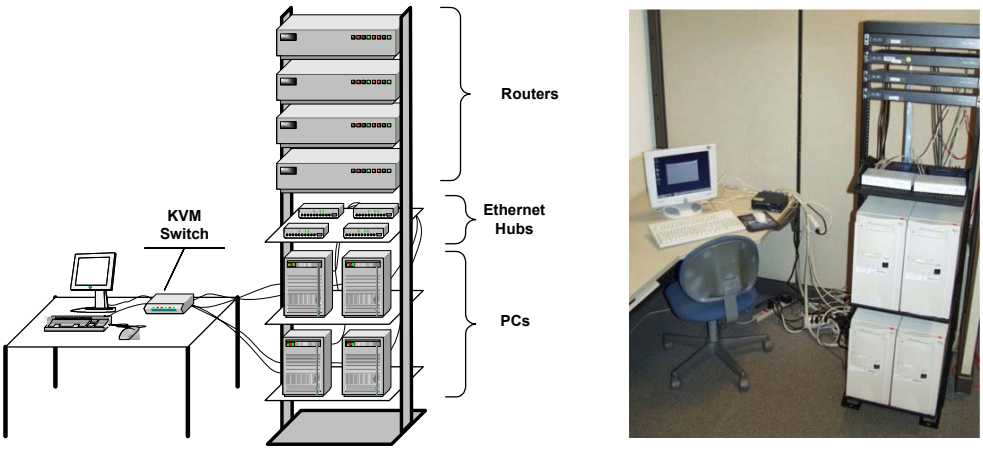


Figure 1. The Internet Lab equipment.

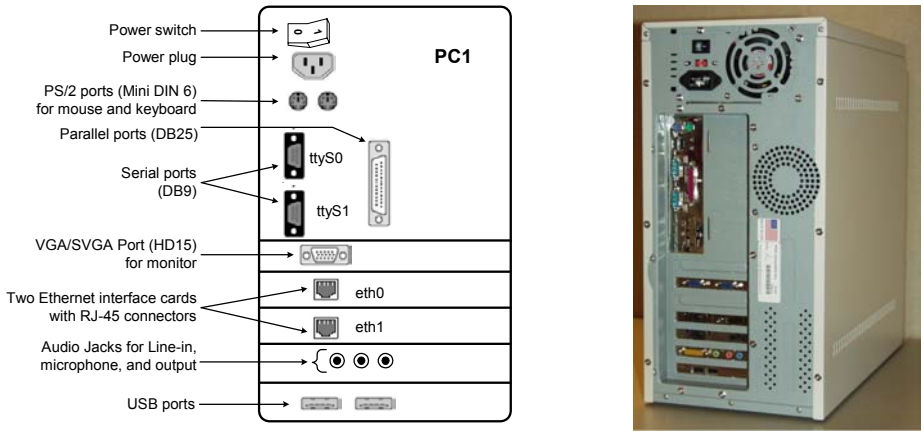


Figure 2. Ports in the back of a PC.

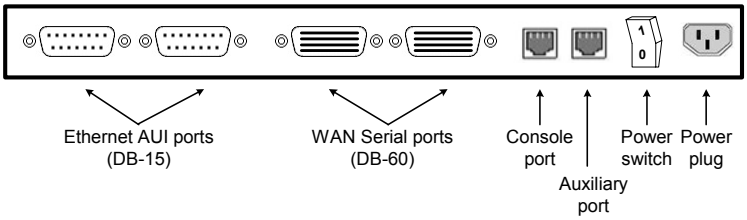
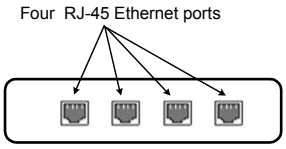


Figure 3. Cisco 2514 router with two Ethernet ports and two serial ports.



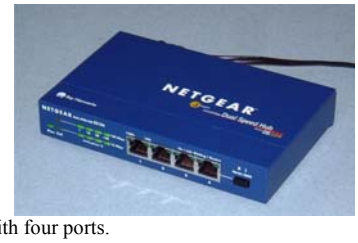


Figure 4. Ethernet Hub with four ports.

The Internet Lab has four Cisco routers, labeled as *Router1, Router2, Router3*, and *Router4*. Each router has two Ethernet interfaces and one or more serial wide area network (WAN) interface. The Ethernet interfaces operate at 10 or 100 Mbps and the serial WAN interfaces have a rate of up to 2 Mbps. Figure 3 shows the back of a Cisco 2514 router, two Ethernet ports and two serial WAN ports. The routers run the Internet Operating System (IOS), the operating system of Cisco routers, which has its own command and configuration language for routers. The version of the operating system used in the Internet Lab is IOS 12.0 or a more recent version.

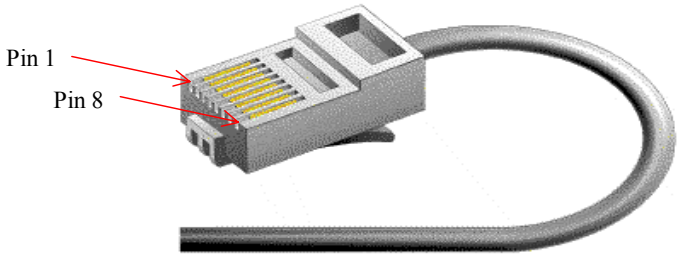
There are four Ethernet hubs in the lab, each with at least four ports (see Figure 0.4). The hubs should be dual-speed, that is, support data transmissions at both 10 and 100 Mbps.

PCs and routers that connect to the same Ethernet hub form an Ethernet local area network (LAN), also called an *Ethernet segment.*

The Internet Lab has many different types of cables and connectors. There are cables to connect the keyboard, video, and mouse ports of the PCs to the KVM switch. Then there are cables that connect the Ethernet interface cards of PCs and routers to the Ethernet hubs. Lastly, there are two types of serial cables. One type of serial cable is used to connect a PC to the console port of a router. The other type of serial cables connects two serial WAN interfaces of the Cisco routers.

1.2. Wiring a Twisted Pair Ethernet Network

Most network experiments in the Internet Lab are done over Ethernet networks, which is the dominant networking technology for local area networks. The Ethernet equipment in the Internet Lab uses the physical layers 10BaseT and 100BaseTX, which are currently the most widely used physical layers for Ethernet. 10BaseT and 100BaseTX have a data rate of 10 Mbps and 100 Mbps, respectively. The cables that connect 10BaseT or 10Base100TX Ethernet interface cards are unshielded twisted pair (UTP) cables with RJ-45 connectors, as shown in Figure 5. We will often to refer to these cables as *Ethernet cables*.



This is copied from the web.

Figure 5. UTP cable with RJ-45 connector.

Setting up a 10BaseT and 100BaseTX Ethernet network configuration is relatively easy. Given a set of PCs, routers, and Ethernet hubs, all that is needed is to connect PCs and hubs with Ethernet cables. However, there are a few things to consider when wiring an Ethernet network. First, there is the quality grade of the UTP cables used. For 100BaseTX Ethernet, cable must satisfy the Category 5 (Cat 5) rating of the Telecommunication Industry Association standard for wiring commercial buildings. A Cat 5 cable has four pairs of twisted wires, even though 10BaseT and 100BaseTX Ethernet only use two of the four wire pairs of a Cat 5 cable.

A second issue to consider when wiring an Ethernet network is that Ethernet cables come in two types: *straight-through cables* and *crossover cables.* The difference between these two types is the assignment of pins of the RJ45 connector to the wires of the Cat 5 UTP cable. In Figure 5, we show an RJ-45 connector with 8 pins. The assignment of pins to the wires is shown in Figure 6. In a straight through cable, the pins of the RJ-45 connectors at both ends of the cable are connected to the same wire. A crossover cable switches pins 1

and 6, and pins 2 and 3, which corresponds to switching the transmit and receiver pins. Since it is easy to confuse straight-through cables and crossover cables, it is good practice to clearly label the different types of cables.

To determine if an Ethernet cable is straight-through or crossover, simply hold the RJ-45 connectors of both cable ends next to each other (with the connector end pointing away from you) and compare how the colored wires are connected to the pins of the RJ-45 connectors. Since the plastic shielding of each wire has a different color, you can determine the difference between a straight-through cable and a crossover cable. In a straight-through cable, colors appear in the same sequence. In a crossover cable the positions of pins 1 and 3, and 2 and 6 are reversed.

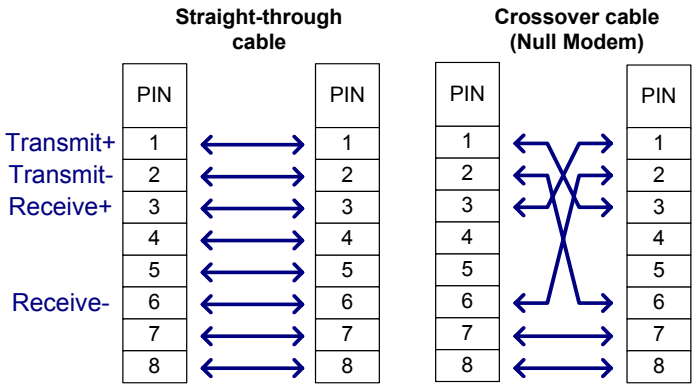


Figure 6. Pin connection for straight through and crossover Ethernet. Ethernet only uses pins 1, 2, 3, and 6.

To connect a PC or router to an Ethernet hub, you need to use a straight-through cable. When connecting two PCs, two routers, or two hubs with an Ethernet cable, you must use a crossover cable. Likewise, when connecting two hubs you must use a crossover cable. There is one special case: some hubs have a port which is labeled as *uplink port*. If you connect on uplink port of a hub to a regular port of another hub you need to use a straight-through cable. Figure 7 illustrates these scenarios. In Figure 7(a) devices are connected to a hub using a straight-through cable. In Figure 7(b) devices are connected directly with crossover cables, and in Figure 7(c), a configuration is shown where hubs are connected to each other, with and without an uplink port.

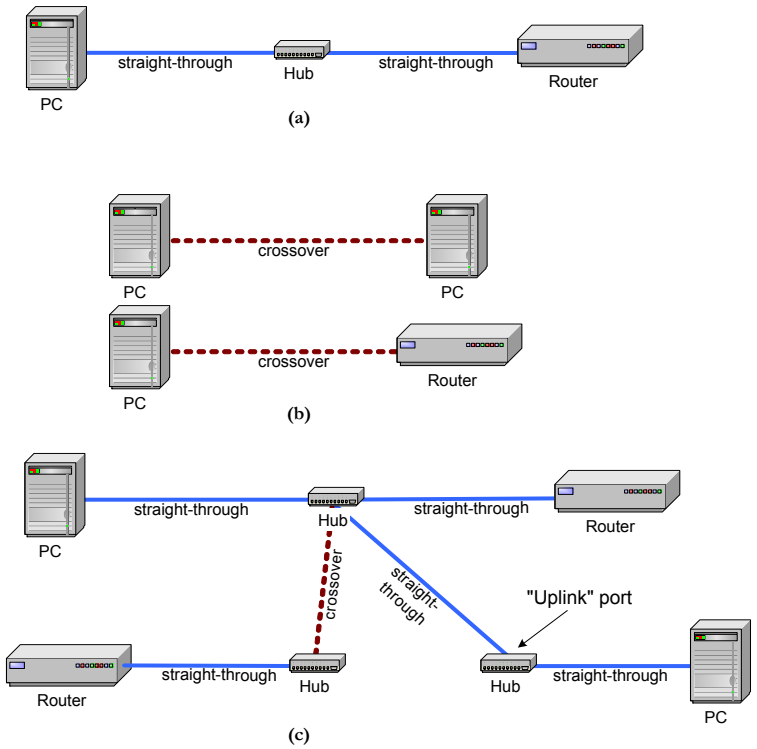


Figure 7. Wiring an Ethernet network with straight-through and crossover cables.

The last issue to consider when wiring an Ethernet network is that some Ethernet interfaces may not have RJ-45 connectors. Some Ethernet interfaces, e.g., those on Cisco 2500 series routers, have 15-pin AUI ports on their Ethernet interfaces. In that case, an AUI/RJ-45 transceiver, as shown in Figure 8, is needed. The transceiver is a small device with an AUI interface on one side and an RJ-45 port on the other side.

Figure 8. Two AUI/RJ-45 transceivers – the top one shows the RJ-45 end, the bottom one shows the AUI end.

1.3. Wiring a Serial Connection from a PC to a Router

Most routers have a *console port* which can be used to configure the router over a serial connection. The console port is an asynchronous serial port that provides an interface for sending and receiving ASCII characters. In the Internet Lab, routers are always accessed via the console port. This is done by connecting a serial cable from a serial port of a Linux PC to the console port of a router, as shown in Figure 9, and by executing a terminal emulation program on the PC which sends and receives characters over the serial port. On the Linux PCs, we use the terminal emulation program *kermit* to access the routers. *Kermit* can send commands to a router and can display the output from a router on a PC.

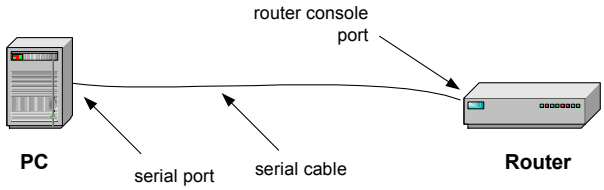


Figure 9. Connecting a PC to a router with a serial cable.

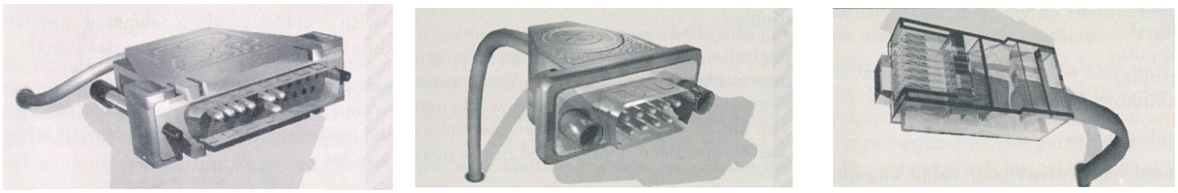
The majority of serial ports follow the EIA/TIA 232-C standard, which is commonly referred to as *RS-232*. The *RS-232* standard defines the electrical signaling, the connectors, cable requirements, and other properties of a serial connection. Devices that comply with this standard can establish a low-bandwidth connection over their serial ports without requiring any additional configuration. Still, establishing a serial connection can be an interestingly complex undertaking, due to the variety of available cables, connectors, and adaptors.

The RS-232 standard specifies a 25-pin connector, called DB-25. However, since most applications of serial connections only use a subset of the available 25 pins, RS-232 connectors with fewer pins are widely used. Among the most popular types of connectors, are the DB-9 connector with 9 pins and the RJ-45 connector with 8 pins. All three types are shown in (shown in Figure 10). Table 1 shows the type of serial connectors found on serial ports of PCs and on the console ports of some Cisco routers.

Machine	**Connector**
Recent PCs (post 1995)	DB-9
Cisco 2500, 2600	RJ-45
Cisco 7000,7200	DB-25

Table 1. Connectors used on various types of PCs and routers.

Most of the time, cables for RS-232 serial connections use a cable with eight wires, such as the cables used for Ethernet. There are three different types of wiring in use: straight-through, crossover, and rollover. The wiring options are illustrated in Figure 6 and Figure 11. Most recent Cisco routers require a rollover wiring to access the console port.



DB-25 DB-9 RJ-45

These are scanned from a book.

Figure 10. Serial connectors.

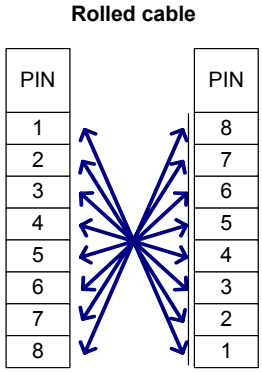


Figure 11. Wiring of a rollover serial cable.

1.4. Wiring a serial WAN connection between two routers

The routers in the Internet Lab have, in addition to Ethernet network interfaces, also have one or more serial wide area network (WAN) interfaces. These interfaces, which are shown in Figure 3, are synchronous serial ports that can operate in full-duplex mode at the rates of a T1 line (1.544 Mbps) or an E1 line (2.048 Mbps). The connectors of the interfaces have a Cisco proprietary format, and are called DB-60 connectors. Instead of Ethernet, traffic over these interfaces is transmitted using the *Point-to-Point Protocol (PPP)* or the *High Level Data Link (HDLC)* protocol.

In the Internet Lab, the serial WAN interfaces, which we will mostly refer to as *serial interfaces*, will be used as shown in Figure 12. Two serial interfaces are connected with a proprietary crossover cable with two DB-60 connectors.

In real networks, serial WAN interfaces are used as shown in Figure 13. Two routers in remote locations can be connected by leasing a T1 line from a telecommunications network provider, which provides a dedicated capacity of 1.544 Mbps. Typically, the T1 service is accessed via an UTP cable with RJ-45 connectors. A router sends traffic over the leased T1 line, by connecting one of its serial interfaces to a data service unit/channel service unit (DSU/CSU), which can be thought of as a modem for a serial WAN connection.

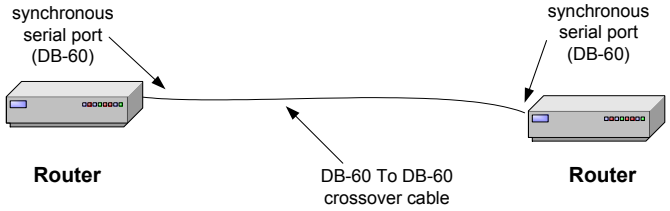


Figure 12. Connecting serial WAN interface cards.

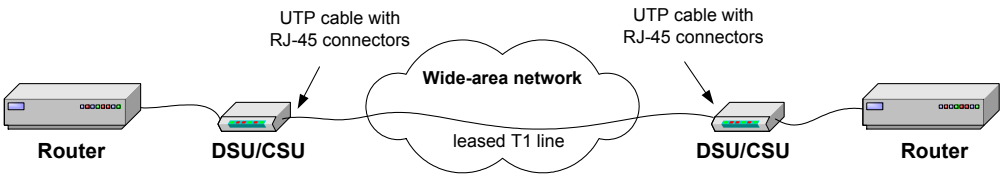


Figure 13. Use of serial WAN interfaces.

2. An Overview of the Internet Lab Software

All network experiments in the Internet Lab are controlled from the PCs. Since the PCs run the Red Hat Linux operating system, you need to become familiar with the Linux operating system. This section provides a brief overview for newcomers to Linux. If you have worked on Linux before, you may want to quickly browse this section, or skip it entirely.

At the end of this section, we discuss some Internet applications which are used extensively in the Internet Lab exercises. These are the applications *Telnet, FTP* and *ping*,

2.1. Linux and Unix

Linux is a clone of the Unix operating system. Unix was developed in the late 1960s at Bell Laboratories. A Unix operating system consists of a kernel and a set of common utility programs. The kernel is the core of the operating system, which manages the computer hardware, provides essential facilities, such as the control of program executions, memory management, a file systems, and mechanisms for exchanging data with attached devices, other programs, or a network. The utility programs include a *Shell,* a command line interface for Unix systems, and numerous commands, compilers, and other programs.

Since its inception, many different versions of Unix-like operating systems have been developed, including AIX (developed by IBM), HP-UX (by Hewlett Packard), SunOS and Solaris (by Sun Microsystems), Ultrix (by DEC), and many others. By the early 1990s, when PCs had become fast enough to run a full Unix-like operating systems, Unix versions for PCs started to emerge, including, FreeBSD, NetBSD, OpenBSD, and Linux. Linux is a new branch of Unix, whose development was initiated by the Finnish computer science student Linus Torvalds. Linux software is distributed freely. Even the source code for Linux is available. Bundled with other free software, particularly, the *GNU* software, which includes editors, compilers, and applications, and the *X Windows* graphical user interface, Linux has become an alternative to the Microsoft Windows as operating system platform for PCs.

The Linux operating system is distributed by organizations that package Linux with other, sometimes proprietary, software. Popular Linux distributions include Red Hat Linux, SuSE, Slackware, Mandrake, and Debian. For the most part, the different distributions of Linux are quite similar. However, there are differences in the configuration files, that is, the files that contain system parameters, which are read when the system or a server on the system are started. Since many lab exercises deal with changing configuration files, the lab experiments are bound to a certain distribution. We have selected Red Hat release version 7.3, which runs the Linux kernel version 2.4. The Internet Lab will also run on more recent versions of Red Hat.

On most Unix systems, a user interacts with the operating system via a graphical window system. Virtually all window systems for Unix systems are based on the *X Windows system*, sometimes simply called *X* or *X11*. In the lab manual, we use the *Gnome desktop* with the *Enlightenment window manager*, one of the popular desktop environments for

Linux, that is based on *X11*. However, all labs can be completed with any window manager or desktop environment for *X11*.

Next we describe some features of Linux, and show how to perform a set of basic tasks. The description of Linux applies to all Unix-like operating systems.

2.1.1. Logging in

Linux is a multiuser operating system where multiple users can work on the same system at the same time. Linux uses accounts to administer access to the system. Before you can work on a Linux system you must provide an account name (*login name*) and a password. This process is referred to as "logging in". Each Linux system has a special account, with login name *root*. The root account is reserved for administrative tasks. The root user, often called *root*, can access all files and all programs, delete all files, create or delete accounts, and change configuration files. In short, the root user can do anything on a Linux system.

Most lab exercises require that you make changes to the configuration of the Linux system, or run programs which require the privileges of the root user. Therefore, whenever you access the PCs in the Internet Lab, you log in as the root user. A risk of logging in as root is that a single inadvertent action may render the system useless and may require a new installation of the operating system. Some actions may even damage the hardware of the system. Therefore, whenever you are logged in as root, exercise caution when deleting files, so that you do not delete a file that is needed by the Linux kernel.

When you log in to a Linux Red Hat that has the Gnome desktop environment installed, you see a window as shown in Figure 14. Other X11 window managers or desktop environments would show a similar window. To login as root, type `*root'* in the login field and press the enter key. When prompted for the password, type the root password and press the enter key. If the password is correct, you will see a desktop similar to the desktop shown in Figure 15. You must know the root password to log in as root user. If you have done the installation of Linux yourself, you have selected a root password during the installation procedure. Otherwise, someone must have given the root password to you.

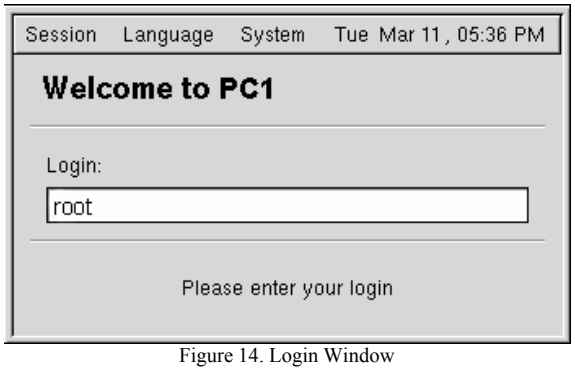


Figure 14. Login Window

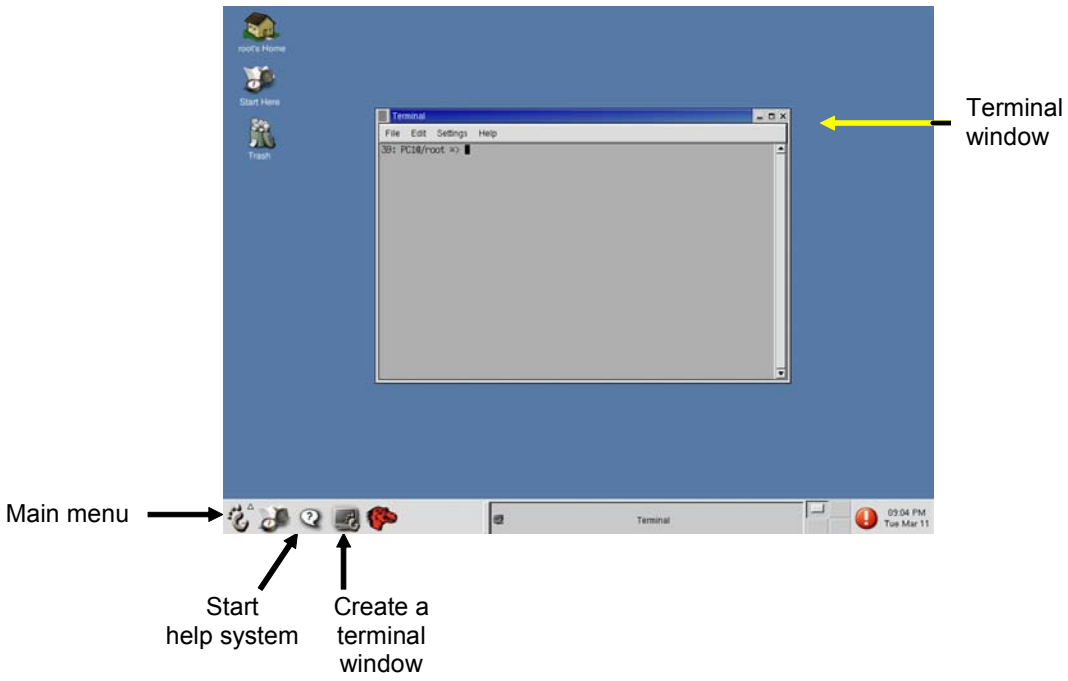


Figure 15. Snapshot of a Gnome Desktop

2.1.2. Navigating the Desktop

The following are a few tasks that you need to be able to perform in the Gnome desktop environment:

- **Opening a terminal window:** To use the command line interface of Linux you need to create a *terminal window*. A new terminal window is created by clicking a button in the panel at the bottom on the desktop, as shown in Figure 15. The center of the desktop in Figure 15 shows a terminal window.

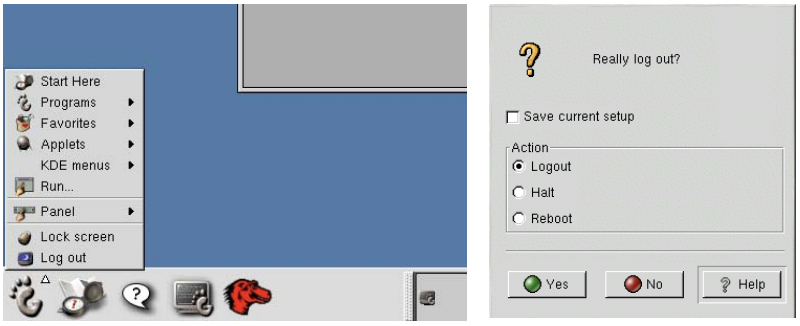
- **Working with windows on the desktop:** You move a window on the desktop by selecting the top bar of a window and dragging the window to its new position. You can hide, maximize, and destroy a window by clicking one of the buttons in the top bar of a window.

- **Cutting and pasting text:** Most X11 windows managers have a simple feature for copying and pasting text. Select text with the left mouse button, move the mouse to the desired position in the same or a different window, and paste the copied text by

pressing the middle mouse button. With a two-button mouse, you can paste by pushing both buttons simultaneously.

- **Logging out:** At the end of each lab session, you must log out of the root account. In the Gnome desktop, you log out by clicking on the main menu button (see Figure 15). In the displayed menu, shown in Figure 16(a), select *Log out.* This will display a window on the desktop as shown in Figure 16(b). Selecting the *Logout* button logs you out of the root account.

- **Getting help:** The Gnome desktop has an online help system. The system is started by clicking on the question mark button in the panel of the Gnome desktop (see Figure 15).



(a) (b)

Figure 16. Logging out in Gnome.

2.1.3. The Linux File System

Like most operating systems, Linux organizes files as a hierarchical tree of directories. Figure 17 shows a snapshot of the directory hierarchy of Linux. The directory at the top of the hierarchy, which is denoted by `/', is called the *root directory*.

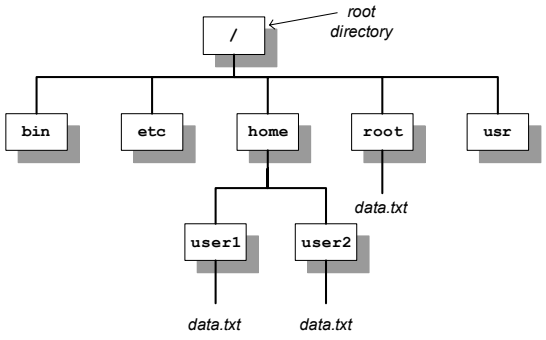


Figure 17. Snapshot of a Linux directory hierarchy.

Each file and directory in a Linux file system is uniquely identified by a *pathname*. Pathnames can be absolute or relative. Absolute pathnames start at the root directory. The absolute pathname of the root directory is a slash (*/*). In the file hierarchy in Figure 17, the absolute pathname of directory *home* in the root directory is */home*, that of directory *user1* in */home* is */home/user1*, and the absolute pathname of file *data.txt* in */home/user1* is */home/user1/data.txt*.

Pathnames that do not begin with a slash are relative pathnames and are interpreted relative to a *current (working) directory*. For example, if the current directory is */home*, then the pathname *user1/data.txt* refers to the absolute pathname */home/user1/data.txt*.

When using relative pathnames, a single dot (.) denotes the *current directory* and two dots (..) denote the *parent directory*, which is the directory immediately above the current directory. With the parent directory, it is feasible to identify each file with relative pathnames. In Figure 17, if the current directory is */home/user1*, the relative pathname .. refers to directory */home,* the pathname ../.. refers to the root directory, and the pathname *../user2/data.txt* refers to the file */home/user2/data.txt.*

Each Linux account has a *home directory*. For regular accounts, that is, accounts which are different from the root account, the home directories are located in */home*. So, */home/user1* is the home directory for an account with login *user1*. The home directory of the root account is */root*. When a new terminal window is created, the current directory in the terminal window is the home directory. Since you log in as root in the Internet Lab, this is directory */root*.

A more complete list of the top levels in the Linux file system hierarchy is shown in Figure 18. Linux configuration files are located in directories */etc*, */usr/etc*, */var* and their subdirectories. Whenever you modify the configuration of a Linux system, you will work on files in these directories.

Each file and each directory has an owner. A regular user only owns the home directory and all files created by the user. The root is the owner of all other files on the system.

In Linux, each file has a set of access permissions. The permissions are *read ("r"), write ("w"),* and *execute ("x"),* and give, respectively, permission to read the contents of a file, modify the file, or execute the file as a program. Permissions are set by the owner of a file. Linux specifies access permissions separately for the owner of the file, a user group which is associated the file, and the set of all users. So, the owner of a file can set the permissions so that all users can read the files, but only the owner can modify the file. The root user can ignore all access permissions and can even change the ownership of files. Since the exercises in the Internet Lab are done from the root account, access permissions are not important for the Internet Lab. The downside of not having to worry about access permissions is that there is no protection against accidentally deleting or corrupting files.

When using a floppy disk or a CD-ROM on a Linux system, the media can be attached to the directory tree of the Linux system. Linux expects that the external media has been formatted with a hierarchical file system that is recognized by Linux, complete with root

directory. The process of adding an external file is illustrated in Figure 19. The figure shows a file system on a floppy disk which is mounted to an existing Linux file system. After mounting the floppy disk, the files on the floppy disk are available through the pathname */mnt/floppy*. The commands for mounting a floppy disk are discussed in Lab 1.

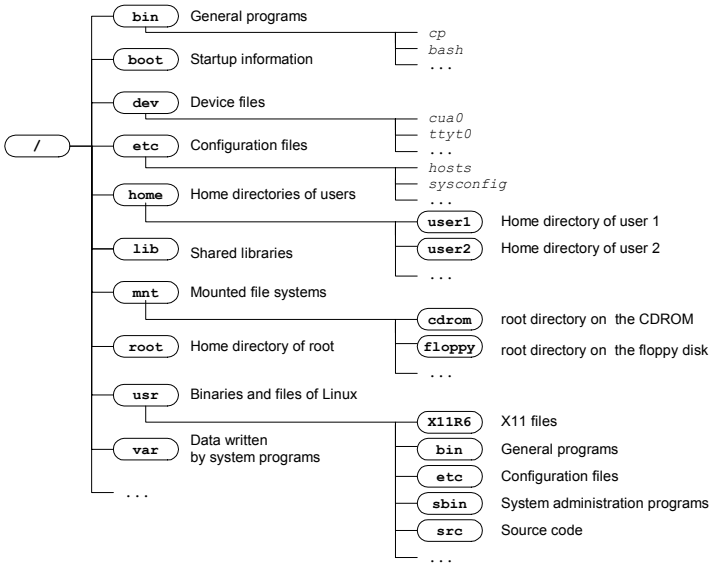


Figure 18. Main directories of the hierarchical Linux directory structure. Directories are shown as boxes and file names are written in italics.

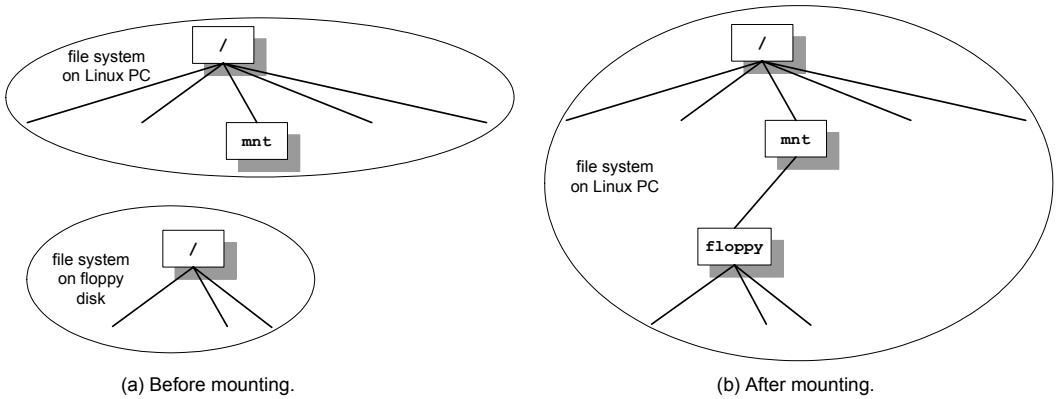


Figure 19. Mounting a file system on a floppy disk.

2.1.4. Linux Devices and Network Interfaces

In Linux, hardware devices, such as, disks, keyboard, and the mouse, are represented by *device files* which reside in the directory */dev*. For example, the mouse of a PC is represented by the device file */dev/mouse*. With device files, communication with an external device is similar to reading and writing from and to a file. When data is written to or read from a device file, Linux communicates with a device driver that is associated with the device file. The device driver communicates and controls a hardware device. In the Internet Lab, you will work with a number of different device files. For example, you will access the serial ports of the PCs via device files */dev/ttyS0* or */dev/ttyS1* and the floppy disk drive will be accessed via */dev/fd0*.

The software abstraction through which the Linux kernel accesses networking hardware is that of a *network interface.* For example, when assigning an IP address to an Ethernet interface cards, one manipulates the configuration parameters of the network interface which represents the Ethernet card. Just like other devices, each network interface is associated with a device driver. In most Unix-like operating systems, a network interface is implemented as a device files. This is different in Linux, where network interfaces are internally defined in the kernel. As a result, networking hardware is handled slightly differently from other hardware. In Linux, the names of network interfaces for Ethernet hardware are *eth0* for the first Ethernet interface card, and *eth1* for the second Ethernet interface card. There is a special network interface, the *loopback interface*, with name *lo*. The loopback interface is not connected to a real device, but is a virtual interface, which allows a PC to send messages to itself.

2.1.5. Linux Shell and commands

The command line interface of the Linux operating system is called a *Shell.* A Shell is a program that interprets and executes Linux commands which are typed in a window terminal. Whenever you create a new terminal window, a Shell is started. The Shell displays a prompt at which the user can type commands. The prompt can be as simple as

```
%
```

or the prompt can be set to provide additional information. For example, in the terminal in Figure 15, the prompt

```
39: PC1@/root =>
```

displays the name of the computer and the current directory. Throughout this book, we use the prompts "%", or "PC1%" if we want to indicate that this is a Shell prompt at PC1. When you type a command at the prompt, and press the enter key, the Shell interprets the command, and, if it is a valid Linux command, executes the command. A Shell is terminated by typing *exit* at the command prompt. If the Shell is running in a terminal window, the terminal window disappears. Linux offers a variety of Shell programs with names such as *sh, csh, ksh, tcsh*, or *bash*. For the purposes of the material covered here, the differences between these Shell programs are not relevant.

Next, we review some basic Linux commands that are typed in at a Shell prompt. Commands in Linux have a common format: a command name, which may be followed by a set of options and arguments. For example, in the command *`ls –l data.txt'*, *`ls'* is the command, *`-l'* is an option which further specifies the command, and *`data.txt'* is an argument. Options are generally preceded by a – (dash), and multiple options can be specified in a single command.

The only built-in help feature of a Linux system are the online manual pages for Linux commands, called the *man pages*. The man pages offer detailed information on a command, however, they provide a lot of detail which are not always helpful for new users of Linux. Desktop environments, such as *Gnome*, provide additional help information.

Getting help

man *cmd*

Displays the on-line manual pages. For example, the command *man ls* displays the manual pages of the command *ls*.

When you login to a PC in the Internet Lab you may find that changes to the Linux system from a previous lab are still in effect. Restarting ("*rebooting*") the operating system removes all temporary configuration changes. Therefore, at the beginning of each lab you should always reboot the Linux PCs.

Rebooting Linux

reboot

Stops and restarts the Linux operating system. Rebooting removes all temporary changes to the operating system. When system configuration files have been modified, the changes are effective after the system reboots.

Do not reboot Linux by powering the PC off and on. This can leave the file system in an inconsistent state.

halt

Stops Linux without restarting.

Note: In the Gnome desktop environment, you can reboot the system following the instructions for logging out. When you arrive at the window shown in Figure 16, simply select *halt* or *reboot*.

Formatted: Font: Not Italic

Deleted: Figure 16

Since all files in Linux are organized as a tree of directories, you need to become familiar with navigating and manipulating the directory tree. The following are commands that relate to Linux directories.

Directory commands:

pwd Prints the absolute path of the current directory.

cd *dirpath*
cd

Changes the current directory to the relative or absolute pathname of the directory *dirpath*. If no directory is given, the command changes the current directory to the home directory. For example, the command *cd /usr/bin* changes to directory */*usr*/bin*, the command *`cd ..'* changes to the parent directory, and the command *cd* without a parameter changes, if you are logged in as root, to directory */root*.

mkdir *dirname*

Creates a new directory with name *dirname* in the current directory. For example, the command *mkdir xyz* creates a new directory with name *xyz*.

rmdir *dirname*

Deletes the directory *dirname* from the current directory. A directory cannot be deleted when it still contains files or subdirectories. Thus, before deleting a directory, all files and subdirectories must be deleted first.

Before discussing the commands to list and manipulate files, we introduce the *wildcard characters* * (star) and ? (question mark). The wildcard character * matches any sequence of zero or more characters, and ? matches any single character. Wildcard characters are useful to describe multiple files in a concise manner. For example, The text string *A*.txt* matches all file names that start with an *`A'* and end with *`.txt',* e.g., *ABC.txt, A.txt,* and *Ab.txt* The text string *A?.txt* matches all filenames that are two characters long and start with *`A',* e.g., *Ab.txt* and *A1.txt.*

File commands:

ls
ls ***dirname***

Lists information about files and directories in the current directory. If the command has a directory name as argument, then the command lists the files in that directory. The *ls* command has several options. The most important is `*ls –l*', which includes extensive information on each file, including, the access permissions, owner, file size, and the time when the file was last modified.

For example, `*ls /'* lists all files and directories in the root directory; *`ls AB*'* lists all files and directories in the current directory that start with *AB*; `*ls –l ..'* prints detailed information on each file and directory in the parent directory of the current directory.

mv ***fname newfile***
mv ***fname dirname***

Renames a file or directory with name *fname* as *newfile*, or moves a file or directory to the directory *dirname.* If the destination file (*newfile*) exists, then the content of the file is overwritten, and the old content of *newfile* is lost. If the first argument is a file name and the second argument is a directory name (*dirname*), the file is moved to the specified directory.

For example, *`mv data.txt text.txt'* simply renames file *data.txt*, and `*mv * /root'* moves all files from the current directory to directory */root* (and gives an error message if the current directory is */root*).

cp ***fname newfile***
cp ***fname dirname***

Copies the content of file *fname* to *newfile*. If a file with name *newfile* exists, the content of that file is overwritten. If the second argument is a directory, then a copy of *fname* is created in directory *dirname*.

For example, *`cp *.txt /tmp'* creates a copy of all files that end with *`.txt*' in directory */tmp*.

rm ***fname***

Removes a file. Once removed, the file cannot be recovered. For example, `*rm *'* removes all files in the current directory.

Note: Linux may not issue a warning when a file is overwritten or when a file is removed. When you use the option *`–i'*, Linux asks for confirmation before overwriting or deleting files. It is strongly recommend that you use *`cp –I'* instead of *cp*, *`mv – i'* instead of *mv*, and *`rm –i'* instead of *rm*. Many Shells are configured to always use the *`-i'* option.
An important thing to have in mind is that Linux does not have an *undo* command that reverses the effects of a previously issued command.

In many lab exercises you need to modify the content of configuration files. For modifying files, the following commands are helpful.

Commands to view and modify the text files:

more ***fname***

Displays the contents of file *fname*, one page at a time. The display can be scrolled with the 'Page Up' and 'Page Down' keys. Keyboard controls are space bar or *'f'* for the next page, '*b*' for the previous page, and *'q'* to end the display.

cat ***fname***

This is similar to the more command, but the file is displayed without stopping at the end of each page.

gedit ***fname***
gedit

This command opens the file *fname* in the text editor *gedit*. A new *textfile* can be written by running *gedit* without an argument. A text editor is used to view or modify the content of a text file.

In addition to *gedit*, Linux has a wide variety of editors that can be used to modify text files. Widely used text editors in Linux include *vi, emacs,* and *pico* . We recommend the *gedit* editor if you have never worked with a text editor on an Unix-like system, since it has an intuitive graphical user interface. To edit the file */etc/hosts* with *gedit*, simply type:

```
PC1% gedit /etc/hosts
```

The user interface of *gedit* is shown in Figure 20. To modify the file simply click on a location in the text window and type text. You can type *Ctrl-c* for copying highlighted text and *Ctrl-v* for pasting text[1]. To save the changes, press the *Save* button. To terminate the exit the application press the *Exit* button. If you push *Exit* and have not saved changes to the file, the editor will ask whether you want to save the changes.

[1] To enter *Ctrl-c*, hold down the `*Ctrl'* key and then press the `*c*' key.

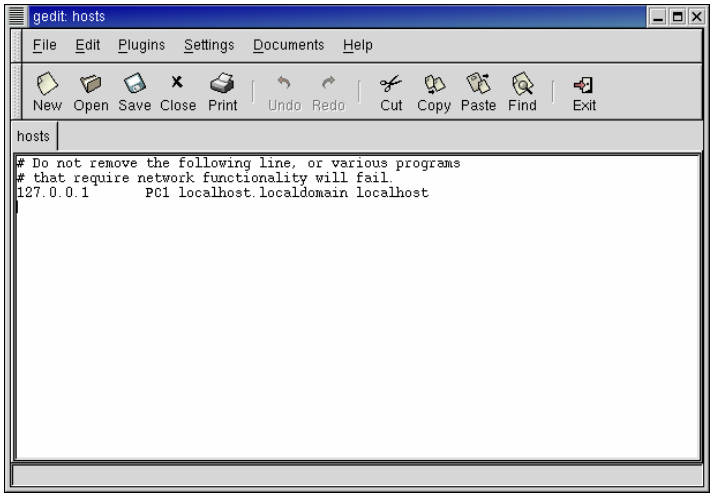


Figure 20.The *gedit* text editor.

Many lab experiments ask you to save data that is displayed in a terminal window to a file. The following commands show how to redirect the output of a terminal window to a file.

Redirecting the output of programs

cmd > fname

The output of *cmd* is written to file *fname*. The file is created if it doesn't already exist, and its contents is overwritten if the file exists. For example, the command *ls > mlist.txt* writes a listing of the current directory in file *mylist.txt*.

cmd >> fname

The **>>** operator *appends* the output of command *cmd* to the end of file *fname*.

For example, the command, *`ls* **>>** *mlist.txt'* appends a listing of the current directory to file *mylist.txt*.

cmd* | tee *fname
cmd* > *fname* & tail –f *fname

Both commands have the effect that the output of command *cmd* is displayed in the terminal window, and also written to file *fname*. The file is created if it does not exist, or the content is overwritten if the file exists.

For example, the command *`ls | tee mylist.txt'* displays the listing of the current directory on the screen, and writes a listing of the current directory to file *mylist.txt.*

In Linux, each terminal window can run multiple commands at the same time. Also, it is possible to stop a command temporarily and resume it at a later time. In each terminal

window, one command can be run as a *foreground process* and multiple command can be run as *background processes*. When a command is issued from the prompt, say

```
% gedit
```

the command *gedit* is started in the foreground. When a command is running in the foreground, no Shell prompt is displayed until the command is finished. The same command can be run in the *background* by adding an & (ampersand) at the end of the command, as follows:

```
% gedit &
```

If a command is executed in the background, the Shell prints a prompt for the next command without any delay. Using background commands, you can run multiple commands from a single terminal window.

You can switch a command that is running in the foreground to the background and vice-versa. Switching a command from the foreground to the background is done as follows:

```
% gedit
Ctrl-z
% bg
%
```

Here, *gedit* is the command in the foreground. Typing *Ctrl-z* stops the command, and *bg* resumes the stopped command in the background. To switch a command from the background to the foreground, type

```
% jobs
```

The command *jobs* lists all commands that are currently running in the background or are stopped, e.g., with *Ctrl-z*. The command

```
% fg %1
```

resumes the first command in the foreground. The following are a set of Linux commands that control the execution of commands.

Control of commands:

Ctrl-c

Pressing *Ctrl-c* terminates the command running in the foreground.

Ctrl-z

Pressing the *Ctrl-z* stops the command running in the foreground.

***cmd* &**

Executes the command *cmd* in the background.

jobs

Lists all background and stopped commands of the current user, and assigns a number to each command.

fg *%n*

fg Resumes the *n*-th command of the user (as listed by the command *jobs*). If no number is given, the command refers to the command that was last running, started, or stopped.

bg %*n*

bg Resumes the *n*-th command of the user that is stopped or running in the background. If no number is given the command refers to the command that was last running, started, or stopped.

kill %*n*

Terminates the *n*-th command of the user.

pkill *cmd*

Terminates a process that executes the command with name *cmd.*

2.2. Applications

We next describe some of the software tools and applications that are used throughout the Internet Lab manual. These are the remote terminal application *Telnet*, the file transfer protocol *FTP*, and the *ping* command, which is a simple, but powerful debugging tool.

2.2.1. Running a Telnet Session

Telnet is a remote login protocol for executing commands on a remote host. To establish a *Telnet* session to a host with name PC2 at IP address 10.0.1.12, simply type the command *telnet 10.0.1.2***.** Assuming that a *Telnet* server is running at PC2, you are prompted for a login name and a password. If the login is successful, you see a Shell prompt from PC2, and can issue Linux commands. A *Telnet* session is terminated by typing *exit* at the command prompt. Figure 21 shows the output from a short *Telnet* session from PC1 to PC2.

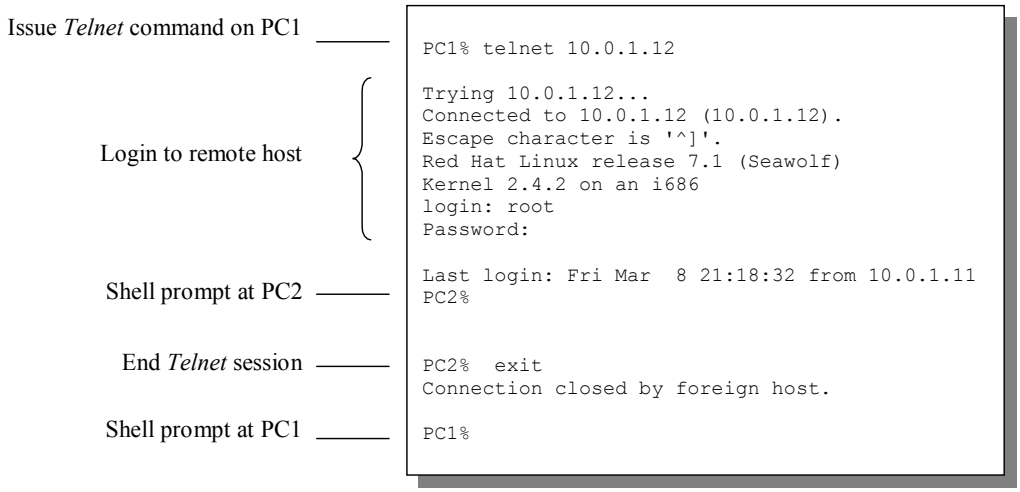


Figure 21. Telnet Session.

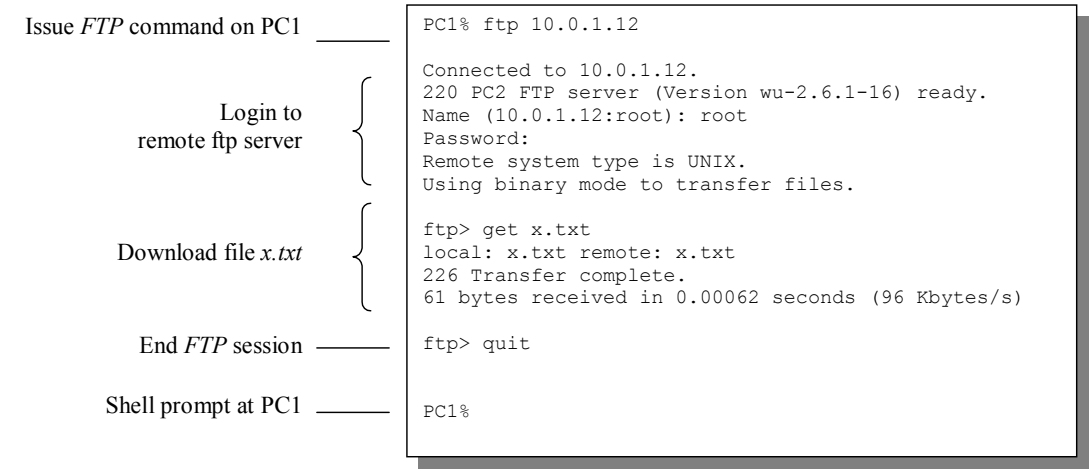


Figure 22. FTP session. (Some system messages are omitted.)

2.2.2. Running an FTP Session

The *File Transfer Protocol (FTP)* is used for copying files between computer systems. An *FTP* session from PC1 to PC2, where PC2 has IP address 10.0.1.12, is initiated by typing the command *ftp 10.0.1.2* (see Figure 22)**.** Similar as in *Telnet*, the user at PC1 is prompted by PC2 for a login name and a password. If the login is successful, the user at PC1 sees a command prompt: *`ftp>'*. The *FTP* prompt accepts a limited set of commands, which can be used to download files from PC2 to PC1 or to upload files from PC1 to PC2. The

command *get* is used to download a file, and the command *put* is used to upload a file. The most important *FTP* commands are summarized as follows.

FTP commands:

ls

Lists the content of the current directory on the remote FTP server. After logging in, the current directory is the home directory of the user.

!ls

Lists the content of the current directory on the local system.

cd *dirname*

Changes the current directory at the remote system to *dirname.*

lcd *dirname*

Changes the current directory at the local system to *dirname.*

binary
ascii

FTP transfers files either as text files or as binary files. The default mode of data transfer is ASCII, which is suitable for transferring text files. Before transferring a file that is not a text file, e.g., a JPEG image or a compiled program, the transfer mode must be switched to binary mode with the command *binary*. The command *ascii* switches back to text mode.

get *fname*
get *fname fname2*

Downloads the file with name *fname* from the current remote directory to the current local directory. If a file with name *fname* exists in the local directory, it is overwritten without issuing a warning. If the command has a second filename as argument (*fname2*), the downloaded file is renamed as *fname2* on the local system.

mget *fname*

Downloads multiple files if *fname* uses wildcard characters. For example, the command *`mget *.txt'* downloads all files that end with *.txt*.

put *fname fname2*

Uploads file *fname* from the current local directory to the current remote directory. If a file with name *fname* exists in the remote directory, it is overwritten without issuing a warning. If the command has a second filename as argument (*fname2*), the downloaded file is renamed as *fname2* on the remote system.

mput *fname*

Uploads multiple files if *fname* uses wildcard characters. For example, the command `*mput *.txt'* uploads all files that end with *.txt.*

quit

Ends the FTP session.

help

Lists all available commands.

2.2.3. Ping

One of the most simple, but also most effective, tools to debug IP networks is the *ping* command. *Ping* tests whether a given IP address is reachable. *Ping* sends a short packet to an IP address and waits for a response from that IP address (see Figure 23). The packets that are issued during a *ping* are *ICMP Echo Request* and an *ICMP Echo Response* messages.[2] The *ping* command sends an *ICMP Echo Request* message to an interface with the specified IP address, and expects an *ICMP Echo Response* message in return.

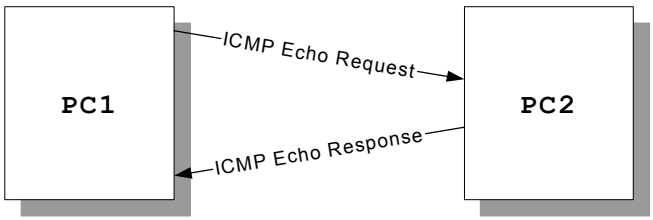


Figure 23. *Ping* Command.

When issuing a *ping* command, a Linux system measures and displays the time between the transmission of the ICMP Echo Request and the return of the ICMP Echo Response. However, the main information provided by *ping* is not the time to receive a response, but whether a certain host is reachable at all. In most cases, if a *ping* command between two machines is successful, most Internet applications are likely to run without problems.

Ping is the single most important tool of a network engineer to troubleshoot problems in a network configuration. Whenever you change the network setup in the Internet Lab, the *ping* command can useful to verify correctness of or troubleshoot the network configuration.

ping *IPaddress*

Issues a *ping* command for the host with the given IP address. The system will

[2] ICMP packets are covered in Lab 2.

issues one *ICMP Echo Request* packet with a size of 56 bytes every second. The command is stopped by typing *Ctrl-c.*

ping -c *num IPaddress*

The command stops after sending *num* ICMP Echo Requests and *receiving num* ICMP Echo Response packets, where *num* is a number.

ping -f *IPaddress*

The sender transmits *ICMP Echo Response* messages as quickly as possible.

ping -i *num IPaddress*

The sender waits for *num* seconds between transmissions of *ICMP Echo Request* messages. The default value is 1 second.

ping -n *IPaddress*

With this option, the output uses numeric IP addresses and does not display symbolic names of hosts.

ping -R *IPaddress*

With this option, the traversed route of the ICMP messages is displayed. The display is limited to the IP addresses of the first nine hops of the route.

ping -s *num IPaddress*

The number of data bytes in the *ICMP Echo Request* is set to *num* bytes. The default value is 56 bytes.

ping -v IPaddress

Displays a verbose output.

3. Network Protocol Analyzers

To make observations of the behavior of network protocols, we need to have tools that can monitor network traffic, and present the traffic in a human readable form. Tools that capture and display traffic on a network interface card are referred to as *network protocol analyzers* or *packet sniffers*. In the Internet Lab we extensively use two network protocol analyzers: *tcpdump* and *ethereal*.

Network protocol analyzers set the network interface card into a mode, called *promiscuous mode*, where the card captures all traffic that passes by the interface card. An Ethernet interface in promiscuous mode can capture the traffic transmitted by all systems that are connected to the same Ethernet hub. Because of the involved security issues, the use of network protocol analyzers is generally restricted to the root user.

The software architecture of a network protocol analyzer on a Linux system with an Ethernet card is shown in Figure 24. The network protocol analyzer is running as an application that communicates with a component in the Linux kernel, called the *Linux socket filter*. The Linux socket filter acts as an agent between the protocol analyzer and the Ethernet device driver. It sets the Ethernet device driver in promiscuous mode and obtains a copy of all incoming traffic from the network and all outgoing traffic to the network. The socket filter processes the traffic and passes the traffic to the network protocol analyzer, which displays the traffic to the user.

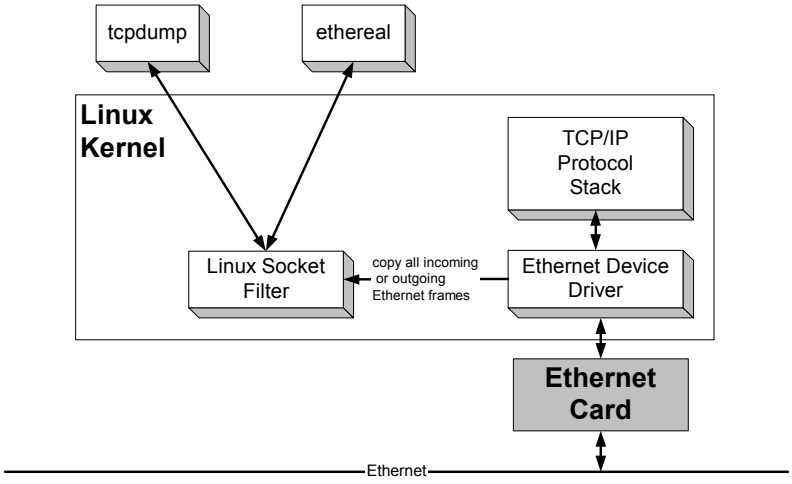


Figure 24. Implementation of network protocol analyzers in Linux.

3.1. Tcpdump

Tcpdump, which was developed in the early 1990s at the Lawrence Berkeley National Laboratory, is started by running the command

```
% tcpdump
```

An example of the output of *tcpdump* is shown in Figure 0.22. The figure depicts the output for an FTP session. The output from *tcpdump* is displayed in the terminal window where the program was started.

Tcpdump displays one line for each transmitted or received Ethernet frame. In each line, *tcpdump* displays a timestamp and information that is derived from the protocol headers contained in the Ethernet frame. The timestamp "*16:54:51.340712*" corresponds to 4:54 PM and 51.340712 seconds. The fractions of second after the second digit may not be very accurate, since the system clocks on most PCs are reliable only for times that exceed 10-50 milliseconds. If the Ethernet frame is an IP datagram with UDP or TCP payload, then *tcpdump* displays information on the source and the destination of the frame. For example, the entry in Line 1 of Figure 25*"128.143.137.144.1555 > 128.143.137.11.53"* indicates that the sender of the IP datagram is IP address 128.143.137.144 at port 1555 and the destination is *128.143.137.11* at port *53.* Even if a frame does not contain an IP datagram, *tcpdump* attempts to interpret the payload. For example, in Figure 25, Lines 7 and 8 show that the payload of the frame is an ARP packet. In addition to IP addresses, *tcpdump* displays information from other protocol headers, such as, TCP, UDP, routing protocols, and other protocols. In Figure 25, Lines 1–6 dispaly information from DNS messages, and Lines 9–13 display information from TCP segment headers.

```
    %tcpdump
1.  16:54:51.340712 128.143.137.144.1555 > 128.143.137.11.53: 1+ A? neon.cs. (25)
2.  16:54:51.341749 128.143.137.11.53 > 128.143.137.144.1555: 1 NXDomain* 0/1/0 (98) (DF)
3.  16:54:51.342539 128.143.137.144.1556 > 128.143.137.11.53: 2+ (41)
4.  16:54:51.343436 128.143.137.11.53 > 128.143.137.144.1556: 2 NXDomain* 0/1/0 (109) (DF)
5.  16:54:51.344147 128.143.137.144.1557 > 128.143.137.11.53: 3+ (38)
6.  16:54:51.345220 128.143.137.11.53 > 128.143.137.144.1557: 3* 1/1/2 (122) (DF)
7.  16:54:51.350996 arp who-has 128.143.71.21 tell 128.143.137.144
8.  16:54:51.351614 arp reply 128.143.71.21 is-at 0:e0:f9:23:a8:20
9.  16:54:51.351712 128.143.137.144.1558 > 128.143.71.21.21: S 607568:607568(0)
                                                          win 8192 <mss 1460> (DF)
10. 16:54:51.352895 128.143.71.21.21 > 128.143.137.144.1558: S 3964010655:3964010655(0)
                                                          ack 607569 win 17520 <mss 1460> (DF)
11. 16:54:51.353007 128.143.137.144.1558 > 128.143.71.21.21: . ack 1 win 8760 (DF)
12. 16:54:51.365603 128.143.71.21.21 > 128.143.137.144.1558: P 1:60(59)
                                                          ack 1 win 17520 (DF) [tos 0x10]
13. 16:54:51.507399 128.143.137.144.1558 > 128.143.71.21.21: . ack 60 win 8701 (DF)
```

Timestamp | Source IP address and destination IP address | Packet headers from other protocols

Figure 25. Output of *tcpdump*.

The following list shows different uses of the *tcpdump* command.

tcpdump -i *interface*

Specifies that *tcpdump* is started on the given interface. This option should be used on systems with multiple network interfaces. For example, *tcpdump –i eth0* starts *tcpdump* on interface *eth0*.

tcpdump -n

With the *`–n'* option, *tcpdump* does not print host names, but prints the IP addresses in the packet. We recommend to always set the *`–n'* option, since resolving host names from the IP addresses may have the undesirable effect that *tcpdump* sends DNS messages, that is, *tcpdump* may generate traffic on its own.

tcpdump -x

With this option, the first 68 bytes of the captured packet are displayed in hexadecimal form.

tcpdump –l

Buffers the output to the terminal window and enables to save output to a file. When saving the output of *tcpdump* to file *fname* use the command
tcpdump -l | tee *fname*
or
tcpdump -l > *fname* & tail -f *fname*

Note: *Multiple options can be used in the same command line. For example, the command `tcpdump –i eth0 –n –x – t –vv' enables all of the above options.*

When the *tcpdump* tool is started with the command

```
% tcpdump –n eth0
```

it displays all packets that are captured on network interface *eth0*. Instead of capturing all traffic, and then searching through the output for the data of interest, one can limit the amount of traffic captured by *tcpdump* by specifying a filter expression in the command line. With a filter expression, only the traffic that matches the filter expression is captured and displayed. For example, the command

```
% tcpdump –n eth0 host 10.0.1.12
```

captures IP datagrams from or to IP address 10.0.1.12, and ignores traffic with different addresses. A list of filter expressions which may be useful in the exercises of the Internet Lab is shown in Table 2. The filter expressions can be combined using negation (*not*), concatenation (*and*), or alternation (*or*) to form complex filter expressions. In filter expressions with multiple operators, negation has the highest precedence. Concatenation and alternation have equal precedence and are interpreted from left to right. For example, the command

```
% tcpdump –n eth0 not \icmp or src host 10.0.1.12 and ip multicast
```

displays IP datagrams that are not ICMP messages or come from host 10.0.1.12 and, in addition, do not have an IP multicast destination address. A different precedence of the operators can be enforced with parentheses. For example, each of the following three filter expressions yields a different result:

```
not \icmp or host 10.0.1.2  and \tcp
not (\icmp or host 10.0.1.2)  and \tcp
not \icmp or (host 10.0.1.2  and \tcp).
```

If an address or number is not specified by a keyword, then the most recent keyword is assumed. For example,

```
host 10.0.1.2  and 10.0.1.3
```

is short for

```
host 10.0.1.2  and  host 10.0.1.2
```

It is possible to access specific fields in protocol headers, and select packets based on the values of protocol header fields. This is done with expressions of the form *proto[offset : size]* which select bytes *offset+1, offset+2, ..., offset+size* from the header of protocol *proto*. For example, *`ip[2:2]'* selects the third and fourth byte in the IP header the total length field. The expression *`ip[2:2]>576'* selects IP datagrams that are longer than 576 bytes. The *tcpdump* expression that displays these IP datagrams is

```
% tcpdump –n 'ip[2:2]>576'
```

Note that the expression is put in quotes (' '). If a selection specifies a protocol header, packets that do not have such a protocol header are simply ignored. Table 3 shows examples for selecting packets based on the contents of protocol headers. Single bits can be tested using a *bitwise and* operator (*&*) and a *comparison* operator (*>, <, >=, <=, =, !=*). For example *`ip[0] & 0x80 > 0'* selects packets where the first bit of the IP header is set. Also, a selection can be combined with any other filter expression. For example,

```
% tcpdump -n 'ip[2:2]>576' and not host 10.0.1.2
```

selects all IP datagrams longer than 576 bytes that do not have IP address 10.0.1.2 as source or destination IP address.

`dst host 10.0.1.2`	IP destination address field is 10.0.1.2
`src host 10.0.1.2`	IP source address field is 10.0.1.2
`host 10.0.1.2`	IP source or destination address field is 10.0.1.2
`src net 10.0.1.0/24`	IP source address matches the network address 10.0.1.0/24
`dst net 10.0.1.0/24`	IP destination address matches the network address 10.0.1.0/24
`net 10.0.1.0/24`	IP source or destination address matches the network address 10.0.1.0/24
`dst port 80`	Destination port is 80 in TCP segment or UDP datagram
`src port 80`	Source port is 80 in TCP segment or UDP datagram
`port 80`	Destination or source port is 80 in TCP segment or UDP datagram
`src and dst port 80`	Destination and source port is 80 in TCP segment or UDP datagram
`tcp port 80` `udp port 80`	Destination or source port is 80 in TCP segment Destination or source port is 80 in UDP datagram
`len <= 200`	Packet size is not longer than 200 bytes

`icmp`
`tcp`
`udp`
	IP protocol field is either set to the number for

`ospf`	ICMP or TCP or UDP or OSPF
`ip proto 17`	IP protocol number is set to 17
`broadcast`	Ethernet broadcast packet
`ip broadcast`	IP broadcast packet
`multicast`	Ethernet multicast packet
`ip multicast`	IP multicast packet
`ip` `arp`	Ethernet payload is IP or ARP

Table 2. Filter expressions for *tcpdump* filters.

`tcp[0]  > 4`	The first byte of the TCP header is greater than 4
`ip[2] <= 0xf`	The third byte of the IP header does not exceed 15
`udp[0:2] == 1023`	The first two bytes of the UDP header are equal to 1023
`ip[0] & 0xf > 5`	IP headers that are longer than 20 bytes, i.e., IP headers with options. The expression *`ip[0]'* selects the first byte from the IP packet and *`ip[0]& 0xf'* filters the four lower order bits in the first byte. The expression *`ip[0] & 0xf > 5'* selects all IP packets where the IP header length is larger than five. Since the IP header field expresses multiples of four bytes, these are packets where the IP header is longer than 20 bytes
`ip[6:2] & 0x1fff == 0`	IP packets where the fragment offset field is zero, i.e., unfragmented IP packets or the first fragment of a fragmented IP packet. The expression *`ip[6:2]'* selects the seventh and eight byte from an IP header, and *`ip[6:2] & 0x1fff'* filters the last 13 bits from these two bytes
`tcp[13] & 3 != 0`	TCP headers with the SYN flag or the FIN flag set. The expression *`tcp[13] & 3'* selects the two least significant bits from the 14-th byte in the TCP header. These bits hold the SYN flag and the FIN flag. The expression is not 0, if at least one of the bits is set

Table 3. Selection of packets based on protocol header contents.

3.2. Ethereal

Ethereal is a protocol analyzer with a graphical user interface, which recognizes a large number of protocols. *Ethereal* is the main tool for capturing traffic in the Internet Lab. *Ethereal* is started from a terminal window with the command

```
% ethereal
```

The command displays a window as shown in Figure 26. The traffic capture is started by selecting *Capture*:*Start* in the main menu of the window. Once the traffic capture is started, the *ethereal* window displays the traffic in three different views. The first view shows a summary of the captured packets, which shows one line for each packet. One of these packets can be highlighted, by clicking on the corresponding line. In Figure 26, the first packet is highlighted. The second view shows the protocol header details from the highlighted packet. Packet headers can be expanded and hidden. In Figure 26, the Ethernet header is expanded and the other headers are hidden. The third view shows the hexadecimal and ASCII representation of the packet header in the second view. The traffic captured by *ethereal* can be saved to a file by selecting *File:Print* in the main menu.

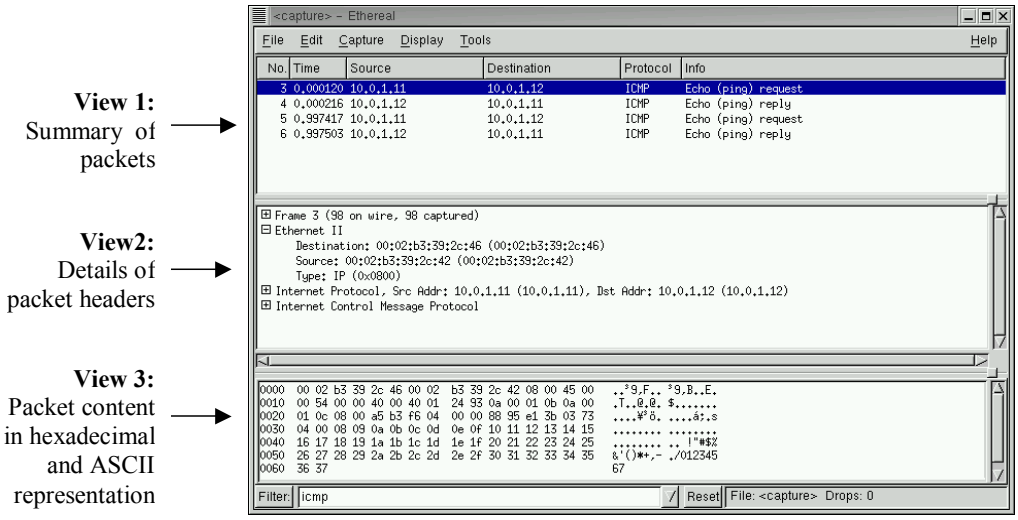


Figure 26. Ethereal window.

As in *tcpdump*, a user can limit the traffic to be captured. In *ethereal*, this is done by setting a *capture filter*. In addition, for traffic that is already captured, a user can display a subset of the captured traffic by specifying a *display filter*.

Capture Filters: A capture filter specifies the type of traffic that is captured by *ethereal*, similarly to filters in *tcpdump*. In fact, capture filters in *ethereal* are written using the same syntax as *tcpdump* filters. A capture filter can be set in the command line when *ethereal* is started or in the capture window before a traffic capture is initiated. The following command is used to set a capture filter from the command line:

```
% ethereal -i interface -f filter
```

where *interface* is a network interface and *filter* is a capture filter expression. The capture filter expression is written using the same syntax as for *tcpdump* filters. If no capture filter is specified, *ethereal* captures all traffic.

Alternatively, the interface and the capture filter can be set from the *Capture Preferences* window of *ethereal*, which is opened by *selecting Capture:Start* in the main window, and by typing in the interface name and the desired filter expression in the appropriate boxes. The *Capture Preferences* window of *ethereal* is shown Figure 27. Here, the interface is set to *eth0* and the capture filter is set to *host 10.0.1.12* .

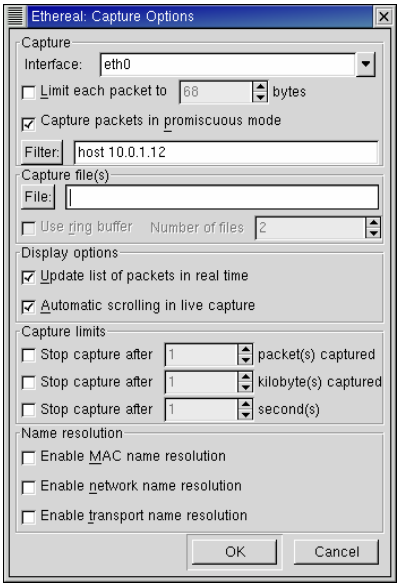


Figure 27. Setting a capture filter in *ethereal*.

Display Filters: A display filter specifies the type of traffic that is displayed in the main window of *ethereal*, but does not restrict the amount of traffic that is captured. An advantage of using display filters is that it is possible to change display filters after packets have been captured. The syntax for setting display filters is different from the syntax for setting capture filters. Also, display filters cannot be set from the command line. A display filter is set by typing a display filter expression at the bottom of main window in *ethereal*, next to the label *Filter*. At the bottom of Figure 26, the display filter expression *icmp* restricts the display of traffic to ICMP messages. In Figure 28, we see that the display filter is set to *ip.dst==10.0.1.12*, which selects all IP packets with the destination IP address *10.0.1.12*. When the filter is applied, only those packets that match the filter are displayed in the main window. The *Reset* button next to the *Filter* box removes the filter.

Figure 28. Setting a display filter in *ethereal*.

The syntax for display filters is different from that for capture filters. In Table 4 and Table 5 we show *ethereal* display filters that correspond to the *tcpdump* filters (and *ethereal* capture filters) from Table 2 and Table 3. Display filters have a separate keyword for each header field of a protocol, and are generally easier to read. The keywords for a particular header field can be obtained from the manual page of *ethereal*.

Ethereal offers interactive help for writing display filters. The help is activated by clicking on the *Filter* button, located in the bottom left corner of the *ethereal* main window (see Figure 26 and Figure 28). This pops up the *Display Filter* window (see Figure 29). Then, clicking on the *Add Expression* button pops up the *Filter Expression* window (see Figure 30). Now, the desired filter expression can be built by selecting a protocol and a protocol field. Once a filter expression is constructed, it is displayed in the main window of *ethereal*.

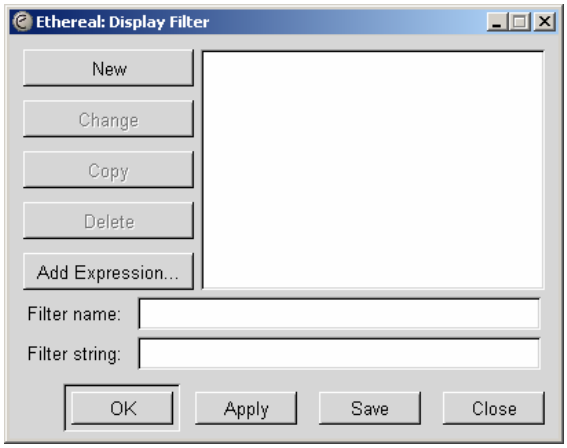


Figure 29. Display filter window of *ethereal*.

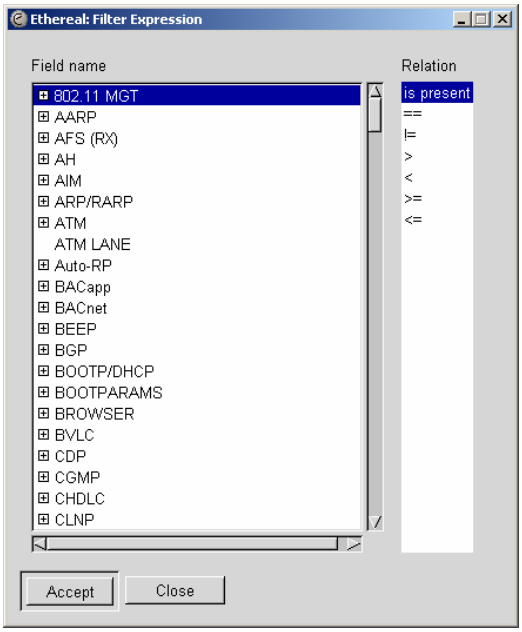


Figure 30. Creating display filter expressions in *ethereal*.

`ip.dst==10.0.1.2`	IP destination address field is 10.0.1.2
`ip.src==10.0.1.2`	IP source address field is 10.0.1.2
`ip.addr==10.0.1.2`	IP source or destination address field is 10.0.1.2
`ip.src==10.0.1.0/24`	IP source address matches the network address 10.0.1.0/24
`ip.dst==10.0.1.0/24`	IP destination address matches the network address 10.0.1.0/24
`ip.addr== 10.0.1.0/24`	IP source or destination address matches the network address 10.0.1.0/24
`tcp.dstport == 80 or udp.dstport == 80`	Destination port is 80 in TCP segment or UDP datagram
`tcp.srcport==80 or`	Source port is 80 in TCP segment or UDP datagram

`udp.srcport==80`	
`tcp.port==80 or udp.port==80`	Destination or source port is 80 in TCP segment or UDP datagram
`(tcp.srcport==80 and tcp.dstport==80) or (udp.srcport==80 and udp.dstport==80)`	Destination and source port is 80 in TCP segment or UDP datagram
`tcp.port==80 udp.port==80`	Destination or source port is 80 in TCP segment Destination or source port is 80 in UDP datagram
`eth.len <= 200`	Packet size is not longer than 200 bytes
`icmp tcp udp ospf`	IP protocol field is either set to the number for ICMP or TCP or UDP or OSPF (see example below) or one can use the protocol name. (Since icmp, tcp, and udp are keywords in *tcpdump,* therefore an escape character (`\') must be placed in front of these keywords.)
`ip.proto==17`	IP protocol number is set to 17
`eth.dst==ff:ff:ff:ff:ff:ff`	Ethernet broadcast packet
`ip.dst[==`	No general expression exists for IP broadcast packet
`eth.dst[0]==1`	Ethernet multicast packet
`ip.dst==224.0.0.0/4`	IP multicast packet.
`ip arp`	Ethernet payload is IP or ARP

Table 4. Display filter expressions in *ethereal* (compare with Table 2).

`tcp[0] > 4`	The first byte of the TCP header is greater than 4
`ip[2] <= f`	The third byte of the IP header does not exceed 15
`udp[0:2] == 3:ff`	The first two bytes of the UDP header are equal to 1023 Note: When selecting bytes from a header, each byte

	is written as a hexadecimal number, and bytes are separated by a colon
`ip.hdr_len > 20`	IP headers that are longer than 20 bytes, i.e., IP headers with options
`ip.frag_offset == 0`	IP packets where the fragment offset field is zero, i.e., unfragmented IP packets or the first fragment of a fragmented IP packet
`tcp.flags.syn==1 or tcp.flags.fin==1`	TCP headers with the SYN flag or the FIN flag set

Table 5. More display filter expressions in *ethereal* (compare with Table 3).

4. Cisco Internet Operating System (IOS)

Just like a general purpose computer, routers run an operating system. The operating system generally is started (*booted*) when a router is powered up. Since routers do not have hard disk drives, the operating system is stored on a flash memory card or nonvolatile RAM (NVRAM). This section gives an overview of the *Internet Operating System (IOS),* the operating system of Cisco routers. The Cisco routers in the Internet Lab run IOS version 12.0 or higher.

In the Internet Lab, routers are always accessed from the PC via the console port, as discussed in Section 1.2. Once the connection is made, the terminal emulation program *kermit* can be started on a PC to send commands to and receive the output from the router. In the Internet Lab, PC1 is connected to Router1, PC2 to Router2, and so on.

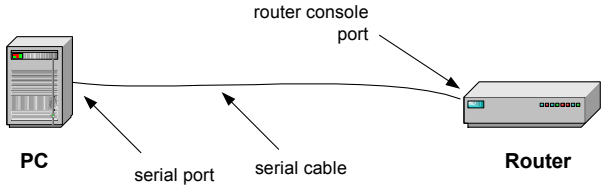


Figure 31. Connecting a cable to a router.

Formatted: Caption, Don't keep with next

If one of the interfaces of a router has an IP address configured, an alternative method to access a router is to *telnet* or *secure Shell* (*ssh)* to the IP address of the configured interface. However, this only works if the router has an interface with a valid IP address.

Deleted: Figure 0.28. Connecting a cable to a router.¶

Once a connection is established, a router shows a command prompt or asks for a login password. After a successful login, a user types commands, similarly as in a Linux Shell. Each router manufacturer has its own command line interface, and the syntax for router commands can be very different across different types of routers. Here, we discuss the command line interface of Cisco IOS.

4.1. The Cisco IOS Command Modes

The command line interface of IOS has a rich syntax. There are hundreds of configuration commands, and some commands have numerous options. Different from a Linux Shell, the command line interface of IOS runs in different modes, and each command requires a certain mode. The Internet Lab features only the most common command modes and, for each command mode, uses only a small subset of available commands. The command modes used in the Internet Lab are the *user EXEC mode*, the *privileged EXEC mode*, the *global configuration mode*, the *interface configuration mode*, and the *router configuration mode.*

Each command mode has a different prompt, and a user can derive the current command mode from the command prompt. The user EXEC Mode is indicated by an angle bracket (>), the privileged EXEC mode by the pound sign (#), and the configuration modes are indicated by an abbreviation of the configuration mode, followed by the pound sign, for example, *(config)#, (config-if)#,* and *(config-router)#.* Typing a question mark (*?*) in any command mode generates a list of all available commands in the current mode.

Table 6 presents a summary of the command modes. Figure 32 illustrates the available transitions between different command modes, and which commands need to be issued. For example, changing from the privileged mode to the global configuration mode is done with the command *configure terminal*. Typing *exit* in this mode returns to the privileged mode. As shown in Figure 32, it is not feasible to switch arbitrarily from one command mode to another. For example, the global configuration mode cannot be entered from the user EXEC mode.

Deleted: Figure 31

Deleted: Figure 31

IOS command mode	Role of command mode	Command prompt
User EXEC mode	• Limited command set, e.g., ping, telnet, traceroute • No change of system parameters	`Router1 >`
Privileged EXEC mode	• Manage configuration files examine state of router • Access control with password (enable secret)	`Router1#`
Global configuration mode	• Change system wide configuration parameters	`Router1(config)#`
Interface configuration mode	• Modify configuration of a specific interface	`Router1(config-if)#`
Router configuration mode	• Modify the configuration of a specific routing protocol	`Router1(config-router)#`

Table 6. Cisco IOS Command Modes.

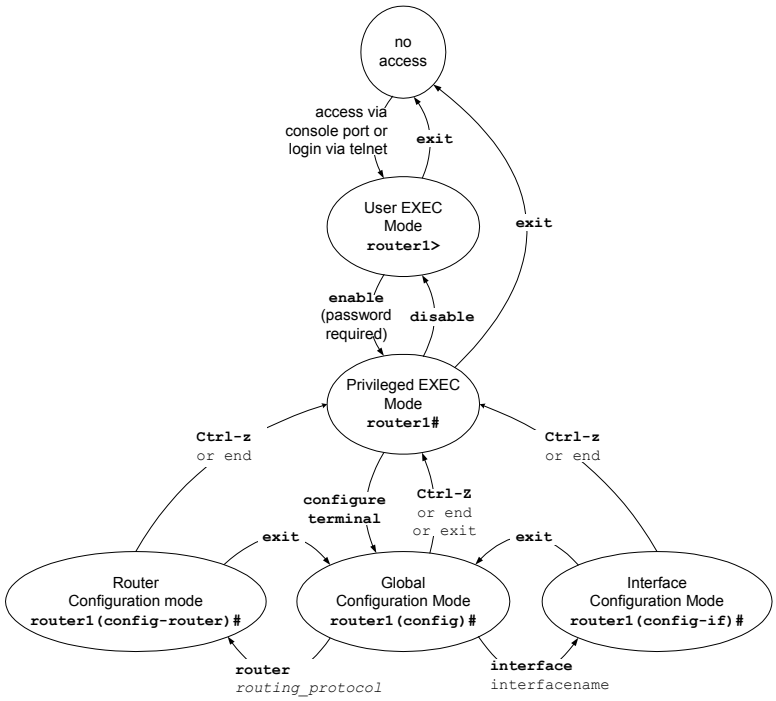


Figure 32. The Cisco IOS Command Modes.

Deleted: 31

4.1.1. User EXEC Mode

The user EXEC mode is entered when the router is accessed via a serial connection or when accessing the router via *telnet*. [3] The command prompt of the user EXEC mode is

```
Router1>
```

where *Router1* is the name that is assigned to the router. The user EXEC mode only offers a small set of commands, such as *ping, telnet*, and *traceroute*. Configuration parameters cannot be read or modified in this mode. Typing

```
Router1>exit
```

logs the user off.

[3] Entering the user EXEC mode over a serial connection may require a login password and entering this mode with *telnet* always requires a login password.

4.1.2. Privileged EXEC Mode

To change or view configuration information of a Cisco router, a user must enter a system administrator mode. In IOS, the system administrator mode is called the *privileged EXEC mode*. In the privileged EXEC mode, a user has rights similar to the root account on a Linux system. The privileged EXEC mode is used to read configuration files, reboot the router, and set operating parameters. To modify the configuration of a router, a user must proceed from the privileged EXEC mode to the global configuration mode, and, from there, to other configuration modes.

Entering the privileged EXEC mode requires to type a password, called the *enable secret*. The privileged EXEC mode is entered from the user EXEC mode by typing the command

```
Router1>enable
Password : <enable secret>
```

Typing the correct password displays the following command prompt:

```
Router1#
```

To change the command mode back to the user EXEC mode, the user types

```
Router1#disable
```

Typing *exit* logs the user off.

4.1.3. Global Configuration Mode

The global configuration mode is used to modify system wide configuration parameters, such as routing algorithms and routing tables. The global configuration mode can only be entered from the privileged EXEC mode. This is done by typing

```
Router1#configure terminal
```

No additional password is required to enter this mode. The argument *terminal* tells the router that the configuration commands will be entered from a terminal. The alternatives are to issue configuration commands from a configuration file or from a remote machine via a file transfer. The command prompt in the global configuration mode is

```
Router1(config)#
```

Global configuration commands include commands that enable or disable IP forwarding and that set static routing table entries. For example, the command

```
Router1(config)#ip routing
```

enables IP forwarding on the router, and the command

```
Router1(config)#ip route 20.0.1.0/24 10.1.1.1
```

adds a network route for destination address 20.0.1.0/24 via gateway 10.1.1.1 to the routing table. Typing *CTRL-z* as in

```
Router1(config)#CTRL-z
```

changes from the global configuration to the privileged EXEC mode.

4.1.4. Interface Configuration Mode

To modify the configuration parameters of a specific interface, for example, the IP address, a user must enter the interface configuration mode. The interface configuration mode, which can only be entered from the global configuration mode, for a network interface is entered by typing the keyword *interface* followed by the interface name.

In IOS, each network interface is associated with a name, which specifies an interface type, a slot number, and a port number. Examples of interface types that are used in the Internet Lab are serial WAN interface (*Serial*), 10 Mbps Ethernet (*Ethernet*), and 100 Mbps Ethernet (*FastEthernet*). Other types of interfaces are FDDI Token Ring (*FDDI*), and Asynchronous Transfer Mode (*ATM*). The slot number indicates the slot into which the interface card is inserted. The port number identifies a port on the interface card. For example, on a Cisco 2611, the interface name Ethernet0/0 identifies port 0 on a 10 Mbps Ethernet card, which is located in slot 0 of the router. Ethernet0/1 identifies port 1 on the same card. On routers which have a fixed number of interfaces and which do not have a slotted chassis, the slot number is omitted. For example, on a Cisco 2514 router, which has two Ethernet interfaces and two serial WAN interfaces, the interface names are *Ethernet0*, *Ethernet1*, *Serial0*, and *Serial1*. Throughout this book, we use the syntax for interface cards for slotted router chassis. For other routers, such as Cisco 2500 series routers, the names of the interfaces need to be changed appropriately. IOS assigns interface names automatically without intervention by a user. The privileged EXEC commands *show protocols* or *show interfaces* lists the names of all interfaces on a router.

The interface configuration mode for the network interface on port 1 of a 10 Mbps Ethernet card inserted in slot 0 of the router is entered with the command

```
Router1(config)#interface Ethernet0/1
```

The command prompt of the interface configuration mode is

```
Router1(config-if)#
```

To return to the global configuration mode one types

```
Router1(config-if)#exit
```

When a global configuration command is typed in the interface configuration mode, then IOS changes to the global configuration command.

4.1.5. Router Configuration Mode

The router configuration mode is used to configure the parameters for a specific routing protocol. When entering the router configuration mode, the name of the routing protocol must be specified as an argument. IOS supports numerous routing protocols, including the *Routing Information Protocol* (*RIP*), *Open Shortest Path First (OSPF),* and *Border Gateway Protocol (BGP)*, and many more. The command to enter the routing router configuration mode for the routing protocol RIP from the global configuration mode is

```
Router1(config)#router rip
```

The command prompt for the router configuration protocol is

```
Router1(config-router)#
```

Typing

```
Router1(config-if)#exit
```

changes to the global configuration mode.

4.2. IOS Commands for Interface Configuration

We next discuss the IP configuration of a network interface in IOS. Consider a router with a 10 Mbps Ethernet (*Ethernet*) interface card with two ports which is located in slot 0 of the router, with names *Ethernet0/0* and *Ethernet0/1*. The following sequence of IOS commands configures port 0 with IP address 10.0.2.1/24 and port 1 with IP address 10.0.3.1/24. In addition, the commands enable IP forwarding on the router.

```
Router1> enable
Password: <enable secret>
Router1# configure terminal
Router1(config)# no ip routing
Router1(config)# ip routing
Router1(config)# interface Ethernet0/0
Router1(config-if)# no shutdown
Router1(config-if)# ip address 10.0.2.1 255.255.255.0
Router1(config-if)# interface Ethernet0/1
Router1(config-if)# no shutdown
Router1(config-if)# ip address 10.0.3.1 255.255.255.0
Router1(config-if)# end
```

The first two commands change to the privileged EXEC mode and, from there, to the global configuration mode. The command *no ip routing*, which is the command to disable IP forwarding, is used to reset the contents of the routing table. The next command**,** *ip routing*, enables IP forwarding on the router. Then, the interface configuration mode is entered for interface *Ethernet0/0*. The command *no shutdown* enables the interface, and the

command *ip address 10.0.3.1 255.255.255.0* sets the IP address to 10.0.3.1/24. The commands to configure the second interface are similar. Note that the interface configuration mode for interface *Ethernet0/1* is entered without returning to the global configuration mode. The last command (*end*) returns to the privileged EXEC mode.

The following list summarizes the IOS commands for enabling IP forwarding and for configuring IP addresses.

IOS mode: Global configuration

ip routing

Enables IP forwarding.

no ip routing

Disables IP forwarding. This command also deletes the content of the routing table.

IOS: Interface configuration

no shutdown

Disables network interface.

shutdown

Enables a network interface.

ip address *IPaddress mask*

Sets the IP address and netmask of an interface to *IPaddress* and *netmask*.

bandwidth ***bw***

Assigns the bandwidth *bw* to an interface. The bandwidth is used as a cost metric by some routing protocols. The bandwidth does not impose a limit on the transmission rate of a network interface.

The routers in the Internet Lab have two Ethernet interfaces, of type *Ethernet* or *FastEthernet*, and one or more serial WAN interfaces of type *Serial*. The names of the interface depend on the types of routers used. On routers with a slotted chassis, the names of the interfaces additionally depend on the slot location of the interface card. The interface names of a router are displayed with the privileged EXEC commands *show interfaces* or *show protocols*. For example, on a Cisco 2611 router, where a *FastEthernet* card with two ports is inserted in slot 0 and a serial card with two ports is inserted in slot 1, the interface names are:

Interfaces on a Cisco 2611 router (with a FastEthernet interface card in slot 0 and a

WAN serial interface card in slot 1):

```
FastEthernet0/0, FastEthernet0/1, Serial1/0, Serial1/1
```

On routers with fixed interfaces cards, the interface names do not list a slot number. For example, on a Cisco 2514, the interface names are:

Interfaces on a Cisco 2514 router:

```
Ethernet0, Ethernet1, Serial0, Serial1
```

4.3. IOS Commands to Display the Configuration and other Information

IOS maintains two configuration files, which are called *startup configuration* and *running configuration*. The configuration files consists of a sequence of IOS commands. The startup configuration is kept in a file on Nonvolatile RAM (NVRAM), and contains the IOS commands that are executed when IOS is booted. To reboot IOS, one can turn the power switch off and then on again. Alternatively, a reboot of IOS is enforced when typing the privileged EXEC command *reload*. When IOS is booted up, the running configuration is set to the startup configuration. The running configuration stores the currently active configuration of the router, and issuing IOS configuration commands modifies the running configuration. The running configuration is kept in RAM and is lost when the router is powered off or when IOS is rebooted. To make changes to the running configuration permanent, the command *copy running-config starting-config* can be used to save the running configuration as the startup configuration.

The commands that display the configuration files are entered from the privileged EXEC mode, and are as given below.

IOS mode: Privileged EXEC

write term
show running-config
Displays the current configuration of the router. Both commands are identical.

show config
show startup-config
Displays the startup configuration of the router. Both commands are identical.

reload
Forces a reboot of IOS. This command discards the running configuration and reloads the startup configuration.

copy running-config starting-config
Saves the current configuration as the startup configuration. The new startup configuration will be used the next time IOS is rebooted.

In Section 4.5, we show the output of the *show startup-config* command for a Cisco 2514 router. In addition to configuration files, various commands are available to display information about the router. Below we list some frequently used commands.

IOS mode: Privileged EXEC

show version

Displays the version of IOS.

show protocols

Displays the IP configuration of the interfaces of the router. Also, indicates if IP forwarding is enabled or disabled.

show ip route

Displays the routing table.

show ip cache

Displays the routing cache.

show interfaces
show interfaces *interfacename*

Displays information about all network interfaces. When an interface name is given as argument, for example, *Ethernet0/1*, information is displayed only for the specified interface.

show ip arp

Displays the contents of the ARP cache.

The *show protocols* command gives a concise overview of the IP configuration of the interfaces of the router.

```
router1#show protocols
Global values:
  Internet Protocol routing is enabled
Ethernet0 is up, line protocol is up
  Internet address is 10.0.2.1/24
Ethernet1 is up, line protocol is up
  Internet address is 10.0.3.1/24
Serial0 is administratively down, line protocol is down
Serial1 is administratively down, line protocol is down
```

From this output, we can tell that IP forwarding is enabled on the router, that the Ethernet interfaces *Ethernet0* and *Ethernet1* are configured with IP addresses, and that the serial

interfaces are currently not used. More extensive information about the interfaces can be displayed with the *show interfaces* command. The output of this and other commands is shown in Section 4.5.

4.4. Navigating the IOS Command Line Interface

IOS provides a few features that make typing commands more convenient. We already mentioned that typing a question mark (*?*) in a given command mode generates a list of all available commands in the current command mode. For example,

```
Router1(config-if)#?
```

lists the available commands in the interface configuration mode. Since IOS commands can only be executed in a certain command mode, this command helps to determine if a command can be executed in the current mode. The question mark can also be used to determine the list of available options of a command. For example,

```
Router1#configure ?
```

lists all options that are available for the command *configure*.

When typing commands or the names of network interfaces, it is sufficient to type just enough characters so that IOS can interpret the input without ambiguity. The following shows how some abbreviations are interpreted.

```
conf                        configure
w t                         write terminal
int e0/0                    interface Ethernet0/0
```

When the Tab key (<*Tab*>) is typed in the command line interface, IOS attempts to complete the command. Command completion is successful only if enough characters are typed so that the prefix can be completed without ambiguity. Here are some examples of command line completions.

```
conf <Tab>                  configure
conf <Tab> t <Tab>          configure terminal
```

An interesting feature of IOS, is that putting a "*no*" in front of same command often creates a valid command. For example, if a certain command enables a feature of a router than adding a "*no*" in front of that command disables the same feature. Sometimes it is the other way around, that is, the command to enable a feature uses the command to disable the feature preceded by a "*no*". The following are a set of examples.

Enable IP forwarding :	**ip routing**
Disable IP forwarding:	**no ip routing**
Add a routing table entry:	**ip route 10.0.2.0 255.255.255.0 10.0.3.1**
Delete a routing table entry:	**no ip route 10.0.2.0 255.255.255.0 10.0.3.1**

Disable a network interface: **shutdown**
Enable a network interface: **no shutdown**

4.5. Displaying IOS Configuration Information

This section shows the output of configuration information from a Cisco 2514 router.

```
router1#show version
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-D-L), Version 12.0(17), RELEASE
SOFTWARE (fc1)
Copyright (c) 1986-2001 by cisco Systems, Inc.
Compiled Mon 16-Apr-01 17:52 by nmasa
Image text-base: 0x03038A64, data-base: 0x00001000

ROM: System Bootstrap, Version 11.0(10c), SOFTWARE
BOOTFLASH: 3000 Bootstrap Software (IGS-BOOT-R), Version
11.0(10c), RELEASE SOFTWARE (fc1)

Router1 uptime is 3 weeks, 6 days, 9 hours, 15 minutes
System restarted by power-on
System image file is "flash:c2500-d-l.120-17"

cisco 2500 (68030) processor (revision L) with 14336K/2048K
bytes of memory.
Processor board ID 07668449, with hardware revision 00000000
Bridging software.
X.25 software, Version 3.0.0.
2 Ethernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
32K bytes of non-volatile configuration memory.
8192K bytes of processor board System flash (Read ONLY)

Configuration register is 0x2102
```

```
router1#show interfaces Ethernet0
Ethernet0 is up, line protocol is up
  Hardware is Lance, address is 00e0.b06a.4eb8 (bia
00e0.b06a.4eb8)
  Internet address is 10.0.2.1/24
  MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255,
load 1/255
  Encapsulation ARPA, loopback not set, keepalive set (10 sec)
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:28, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 1 packets/sec
```

```
  5 minute output rate 0 bits/sec, 0 packets/sec
     5557 packets input, 1540509 bytes, 0 no buffer
     Received 5035 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0
abort
     0 input packets with dribble condition detected
     247651 packets output, 16613143 bytes, 0 underruns
     187763 output errors, 0 collisions, 77 interface resets
     0 babbles, 0 late collision, 0 deferred
     187763 lost carrier, 0 no carrier
     0 output buffer failures, 0 output buffers swapped out
```

```
router1#show interfaces Serial0
Serial0 is administratively down, line protocol is down
  Hardware is HD64570
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load
1/255
  Encapsulation HDLC, loopback not set, keepalive set (10 sec)
  Last input never, output never, output hang never
  Last clearing of "show interface" counters never
  Queueing strategy: fifo
  Output queue 0/40, 0 drops; input queue 0/75, 0 drops
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     0 packets input, 0 bytes, 0 no buffer
     Received 0 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     0 packets output, 0 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 output buffer failures, 0 output buffers swapped out
     0 carrier transitions
     DCD=down  DSR=down  DTR=down  RTS=down  CTS=down
```

```
router1#show startup-config
Using 825 out of 32762 bytes
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname router1
!
enable secret 5 $1$94pd$J.oTq5ujOU6Zko00pndrA/
enable password rootroot
!
ip subnet-zero
!
!
```

```
!
interface Ethernet0
 ip address 10.0.1.1 255.255.255.0
 no ip directed-broadcast
 no ip mroute-cache
!
interface Ethernet1
 ip address 10.0.2.1 255.255.255.0
 ip helper-address 10.0.1.21
 no ip directed-broadcast
 no ip mroute-cache
!
interface Serial0
 no ip address
 no ip directed-broadcast
 no ip mroute-cache
 shutdown
 no fair-queue
!
interface Serial1
 no ip address
 no ip directed-broadcast
 no ip mroute-cache
 shutdown
!
ip classless
!
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
line con 0
 transport input none
line aux 0
line vty 0 4
 password rootroot
 login
!
end
```

Deleted: uva