Indira Gandhi National Open University
School of Computer and Information Sciences

MCS-022 OPERATING SYSTEM CONCEPTS & NETWORKING MANAGEMENT

Linux Operating System **2**

Indira Gandhi
National Open University
School of Computer and
Information Sciences

MCS - 022
OPERATING SYSTEM CONCEPTS & NETWORKING MANAGEMENT

Block

2

LINUX OPERATING SYSTEM

May, 2005

© Indira Gandhi National Open University, 2005

ISBN-81-266-

Printed and published on behalf of the Indira Gandhi National Open University, New Delhi by the Director, School of Computer & Information Sciences.

Laser Typeset : Rajshree Computers, 5A/177, W.E.A., Karol Bagh, New Delhi-110005.

Printed at :

BLOCK INTRODUCTION

The block introduces a large set of concepts related to GUI, operating system & computer network. It also briefly elaborates on the design description of GUI and operating system. The block comprises four units.

Unit 1: It introduces several GUI terms and its functionalities and what are the GUI design considerations. At the end it looks at the several GUI standards.

Unit 2: It describes the basic principles and concept of operating system. It provides a discussion on the evolution of operating system, several design approaches to operating system, classification and characteristics of operating system. The unit is independent of any operating system environment.

Unit 3 & **Unit 4** mainly focus on networking concepts: types of networks, topologies, transmission media (wired) connecting devices such as bridge, repeaters, gateways, etc.

UNIT 1 INTRODUCTION TO LINUX OPERATING SYSTEM

Structure **Page Nos.**

1.0 INTRODUCTION

Today's trends are towards development of free and platform independent software Java popularised this concept in the field of programming language and now it is being done by Linux in the operating system.

Linux started out as a Unix variant to run on an IBM PC platform but with one major difference — its source code was freely available under the auspices of Free Software Foundation (FSF). Due to this, it quickly positioned itself as an alternative to other Unix workstations such as those offered by Sun Microsystem, Compaq and Sillicon Graphics. Due to high quality designing of its kernel qualities such as stability, modularity and easy configurability — it is now dominating the corporate world significantly. For example, major banks, investment houses, retail establishments, educational institutions, etc., use it.

1.1 OBJECTIVES

After going through this unit you will be able to :

- list the features of Linux;
- list the drawbacks of Linux;
- understand the process of thread management, and
- understand the memory management features.

1.2 FEATURES OF LINUX

Linux has several strong features. In this section we will discuss and explain them.

Linux is inexpensive

The first benefit of Linux is cost. All versions of Linux may be freely downloaded from the web. If you don't want to download, prepackaged versions of Linux may be purchased online. In addition, the software may be legally shared with your friends. In addition, when the time comes to upgrade the operating system, the Linux upgrade would be free.

In addition to being inexpensive, Linux can run on the old system. Its products can run on Intel 386 microprocessors, which were popular in the late 1980s. The server has never slowed down despite increased use.

Linux is Fast

Linux runs respectably well on old computers, and it is even faster on newer, more powerful computers. This is because Linux programs are very efficient and lean. They use as few resources as possible, and unlike Windows, Linux programs use little, if any, graphics. Graphics can slow a system's response time, making it slower than it truly is. Linux may not be pretty, but it is fast.

Linux is Stable

The Linux code is well written. This both increases the speed at which Linux runs and improves the stability of the operating system. Linux is next to impossible to crash. If an application crashes, you can simply remove the program from memory to restart your computer. In older versions of Windows, a crashing program had the potential to take down the entire computer. This is one of the reasons why Linux is used on many web servers where stability is crucial. With Linux, web-hosting providers can guarantee 99.9 percent uptime.

Open-Source Software

Finally, Linux has open-source software. This means that users can read the source code and modify it as needed. This probably means little to the average user of the final version of a Linux kernel. However, during development, "beta" releases of the kernel are available to developers who will download the code and test it thoroughly. When possible, they will find any problems and correct the code. This process helps to ensure that the final release of the kernel is as well written as possible.

1.3 DRAWBACKS OF LINUX

Even though Unix and Linux operating systems are widely used on corporate servers, web sites, and large-scale networking environments, we still won't find many people using it on their desktop computers or workstations at home. There are several reasons for this.

Security

Because code is distributed with the Linux software, programmers are free to explore how the system works – for good or bad. Many security loopholes have been reported in the literature. Although most system vulnerabilities are detected before the product is released, clever programmers can still discover new ones.

Lack of Support

No system is 100 percent secure. Even Microsoft products have security vulnerabilities. However, Microsoft products do have extensive documentation and support. Microsoft releases service pack and updates frequently to fix discovered vulnerabilities.

Support and documentation for Linux can be spotty at best. A customer who downloads Linux from a server may receive only an electronic manual and access to online help pages.

Limited Software Selection Choice

People purchase computers to run software. Users of Windows computers have many software titles to choose from. Linux users have software in every category but are often limited in their choices. For example, consider Internet browsers. The two most popular browsers are Netscape Navigator and Microsoft Internet Explorer. Both are available for Linux, but only Netscape has created a Linux version of its latest browser, version 6.32. internet Explorer's most recent Linux version is 5, despite the fact that Windows XP ships with Internet Explorer version 6.

You are also limited in your choice of word processors. The most popular word processor is Microsoft Word. Chances are that every computer in your school uses Word. But Word is not available for Linux. Your best choice is **StarOffice**, a suite of applications that contains a word processor. However, StarOffice, although a very nice product, is not Word. A proficient Word user would have to learn some new skills to use StarOffice.

Limited Hardware Support

Just as not all popular software run on Linux, not all hardware products work with Linux. Red Hat and the other Linux vendors work very hard to support the more common devices. They provide drivers for hardware devices. A driver is a small program that allows the operating system to communicate with the peripheral devices. Having the correct driver is crucial. If you have a new or unusual device, you may be out of luck. For instance, many branded companies have not written a Linux compatible driver.

Complexity

Linux, like its predecessor Unix, assumes that you know what you are doing, and it assumes that you know the consequences of every command you type. In contrast, Windows XP asks you to verify everything, and then shows you a pretty animation to confirm the action.

For a beginning user, Linux can be frightening to use; entering the wrong command can have serious consequences. It dosen't help that Linux is also case sensitive, so you must enter the commands in lowercase, and be careful to use the correct case for each subcommand you use with a command. Upper — and lowercases are often different actions.

1.4 COMPONENTS OF LINUX

In this section we will take up the various component of Linux Operating System.

1.4.1 Memory Management Subsystem

Linux is made up of a number of functionally separate pieces that, together, comprise the operating system. One obvious part of Linux is the kernel itself; but even that would be useless without libraries or shells. In this section we will discusss the various components of Linux kernel.

One of the basic objectives of any operating system is to make one feel that there is a large amount of memory although it is having a small physical memory. This apparently large memory is known as virtual memory. The system divides the memory into easily handled pages (logical unit) and swaps these pages onto a hard disk as the system runs.

The memory management subsystem is one of the most important parts of the operating system. Since the early days of computing, there has been a need for more memory than exists physically in a system. Strategies have been developed to overcome this limitation and the most successful of these is virtual memory. Virtual memory makes the system appear to have more memory than it actually has by sharing it between competing processes as they need it.

Virtual memory does more than just make your computer's memory go further. The memory management subsystem includes:

Large Address Spaces: The operating system makes the system appear as if it has a larger amount of memory than it actually has. The virtual memory can be many times larger than the physical memory in the system.

Protection: Each process in the system has its own virtual address space. These virtual address spaces are completely separate from each other and so a process running one application cannot affect another. Also, the hardware virtual memory mechanisms allow areas of memory to be protected against writing. This protects code and data from being overwritten by rogue applications.

Memory Mapping: Memory mapping is used to map image and data files into a processes address space. In memory mapping, the contents of a file are linked directly into the virtual address space of a process.

Fair Physical Memory Allocation: The memory management subsystem allows each running process in the system a fair share of the physical memory of the system.

Shared Virtual Memory: Although virtual memory allows processes to have separate (virtual) addresses spaces, there are times when you need processes to share memory. For example, there could be several processes in the system running concurrently and simultaneously depending upon the number of processors residing in the system but might be using the common file, e.g., C-amplifier.

Therefore, it is better to have only one copy in physical memory and all of the processes running sharing it. Dynamic libraries are another common example of executing code shared between several processes.

Another example of shared memory is that it can also be used as an Inter Process Communication (IPC) mechanism, with two or more processes exchanging information via memory common to all of them. Linux supports the Unix^{TM} System V shared memory IPC.

An Abstract Model of Virtual Memory

Before considering the methods that Linux uses to support virtual memory it is useful to consider an abstract model which is applicable to a large number of systems.

As the processor executes a program it reads an instruction from memory and decodes it. In decoding the instruction it may need to fetch or store the contents of a location in memory. The processor then executes the instruction and moves onto the next instruction in the program. In this way the processor is always accessing memory either to fetch instructions or to fetch and stored data.

In a virtual memory system all of these addresses are virtual addresses and not physical addresses. These virtual addresses are converted into physical addresses by the processor through a mapping scheme using a set of tables maintained by the operating system.

To make this translation easier, virtual and physical memory are divided into handy sized chunks called pages. These pages are all of the same size. They need not be, but if they were not the system would be very hard to administer. The size of a page varies from one system to another. Each of these pages is given a unique number; the frame number (FN). In this paged model, a virtual address comprises two parts; virtual page frame number (VPFN) and offset within the frame. Each time the processor encounters a virtual address it must extract the virtual page frame number and the offset. The processor must translate the virtual page frame number into a physical one (address of RAM) and then access the location at the correct offset into that physical page. To do this the processor uses page tables. The size of a page table varies from one process to another and the number of page tables depends upon the number of processes residing in a system.

Figure 1 shows the virtual addresses spaces of two processes, process P1 and process P2, each with its own page tables. These page tables map each processes virtual pages into physical pages in memory. This shows that process P_1's virtual page frame number 0 is mapped into memory in physical page frame number 1 and that process P_2's virtual page frame number 1 is mapped into physical page frame number 4. Entry in the theoretical page table contains the following information:

Figure 1: Abstract model of Virtual to Physical address mapping

The processor uses the virtual page frame number as an index into the processes page table to retrieve its page table entry. If the page table entry at that offset is valid, the processor takes the physical page frame number from this entry. If the entry is invalid, the process has accessed a non-existent area of its virtual memory. In this case, the processor cannot resolve the address and must pass control to the operating system so that it can fix things up.

In case the required page frame is not found, the processor generates a page fault and then the required page is brought from the hard disk.

Mapping of virtual address to physical address can be done in any order. For example, in process P_1 virtual page frame number 0 is mapped to physical page frame number 1 whereas virtual page frame number 7 is mapped to physical page frame number 0 even though it is higher in virtual memory than virtual page frame number 0. This demonstrates an interesting byproduct of virtual memory; the pages of virtual memory do not have to be present in physical memory in any particular order.

Linux shares many of the characteristics of the memory management schemes of other Unix implementations but has its own unique features. Overall, the Linux memory management scheme is quite complex. Here, we give a brief overview.

Linux Virtual Memory

Linux makes use of a three-level page table structure, consisting of the following types of tables (each individual table is the size of one page):

- **Page Directory:** An active process has a single page directory that is the size of one page. Each entry in the page directory points to one page of the page middle directory. The page directory must be in main memory for an active process.
- **Page Middle Directory:** The page middle directory may span multiple pages. Each entry in the page middle directory points to one page in the page table.
- **Page Table:** The page table may also span multiple pages. Each page table entry refers to one virtual page of the process.

To use this three-level page table structure, a virtual address in Linux is viewed as consisting of four fields. The leftmost (most significant) field is used as an index into the page directory. The next field serves as an index into the page middle directory. The third field serves as an index into the page table. The fourth field gives the offset within the selected page of memory.

Page Allocation

To enhance the efficiency of reading in and writing out pages to and from main memory, Linux defines a mechanism for dealing with contiguous blocks of pages mapped into contiguous blocks of page frames. For this purpose, the buddy system is used. The kernel maintains a list of contiguous page frame groups of fixed size; a group may consist of 1, 2, 4, 16, or 32 page frames. As pages are allocated and deallocated in main memory, the available groups are split and merged using the buddy algorithm.

Page Replacement Algorithm

The Linux page replacement algorithm is based on the clock algorithm in which a use bit and a modify bit are associated with each page in main memory. In the Linux scheme, the use bit is replaced with an 8-bit age variable. Each time that a page is accessed, the age variable is incremented. In the background, Linux periodically sweeps through the global page pool and decrements the age variable for each page as it rotates through all the pages in main memory. A page with an age of 0 is an "old" page that has not been referenced in some time and is the best candidate for replacement. The larger the value of age, the more frequently a page has been used in recent times and the less eligible it is for replacement. Thus, the Linux algorithm is a form of least frequently used policy.

1.4.2 Linux Process and Thread Management

Processes carry out tasks within the operating system. A program is a set of machine code instructions and data stored in an executable image on disk and is, as such, a passive entity; a process can be thought of as a computer program in running state. It is a dynamic entity, constantly changing as the machine code instructions are executed by the processor. As well as the program's instructions and data, the process also includes the program counter and all of the CPU's registers as well as the process stacks containing temporary data such as routine parameters, return addresses and saved variables. Linux is a multiprocessing operating system which can support many processes running in parallel. Processes are separate tasks each with their own rights and responsibilities and also running in their own address spaces. If one process crashes it will not cause another process in the system to crash. Each individual process runs in its own virtual address space and is not capable of interacting with another process except through secure mechanisms to be managed by kernel.

The most precious resource in the system is the CPU, usually there is only one except in a multi-processors based system. Linux is a multiprocessing operating system, its objective is to have a process running on each CPU in the system at all times, to maximize CPU utilization. If there are more processes than CPUs (and there usually are), the rest of the processes must wait before a CPU becomes free until they can be run. In a multiprocessing system many processes are kept in memory at the same time. Whenever a process has to wait, the operating system takes the CPU away from that process and gives it to another, more deserving process. It is the scheduler which chooses which is the most appropriate process to run next and Linux uses a number of scheduling strategies to ensure fairness.

Linux supports a number of different executable file formats, ELF (Executably and linkable format) is one, Java is another and these must be managed transparently.

Data structure for Linux Processes

So that Linux can manage the processes in the system, each process is represented by a task — struct data structure (task and process are terms that Linux uses Interchangeably). The task vector is an array of pointers to every task-struct data structure in the system. This means that the maximum number of processes in the system is limited by the size of the task vector; by default it has 512 entries. As processes are created, a new task_struct is allocated from system memory and added into the task vector. To make it easy to find, the current, running process is pointed to by the current pointer.

As well as the normal type of process, Linux supports real time processes. These processes have to react very quickly to external events (hence the term "real time") and they are treated differently from normal user processes by the scheduler. Although the task-struct data structure is quite large and complex, its fields can be divided into a number of functional areas:

State: As a process executes its changes state according to its circumstances. Linux processes have the following states:

Running: The process is either running (it is the current process in the system) or it is ready to run (it is waiting to be assigned to one of the system's CPUs).

Waiting: The process is waiting for an event or for a resource. Linux differentiates between two types of waiting process; interruptible and uninterruptible. Interruptible waiting processes can be interrupted by signals whereas uninterruptible waiting processes are waiting directly on hardware conditions and cannot be interrupted under any circumstances.

Stopped: The process has been stopped, usually by receiving a signal. A process that is being debugged can be in a stopped state.

Zombie: This is a halted process which, for some reason, still has a task-struct data structure in the task vector. It is what it sounds like, a dead process.

Scheduling Information: The scheduler needs this information in order to fairly decide which process in the system most deserves to run,

Identifiers: Every process in the system has a process identifier. The process identifier is not an index into the task vector, it is simply a number. Each process also has User and group identifiers, these are used to control this processes access to the files and devices in the system.

Inter-Process Communication (IPC): Linux supports the classic Unix™ IPC mechanisms of signals, pipes and semaphores and also the System V IPC mechanisms of shared memory, semaphores and message queues to allow processes to communicate with each other and with the kernel to coordinate their activities.

Links: In a Linux system no process is independent of any other process. Every process in the system, except the initial process has a parent process. In Unix operating system the initial process is known as init. New processes are not created, they are copied, or rather cloned from previous processes. Every task-struct representing a process keeps pointers to its parent process and to its siblings (those processes with the same parent process) as well as to its own child processes.

Times and Timers: The kernel keeps track of a processes creation time as well as the CPU time that it consumes during its lifetime. Each clock tick, the kernel updates the amount of time in jiffies that the current process has spent in system and in user mode. Linux also supports process specific interval timers, processes can use system calls to set up timers to send signals to themselves when the timers expire. These timers can be single-shot or periodic timers.

File System: Processes can open and close files as they includes pointers to any files opened by this process.

Virtual memory: Most processes have some virtual memory (kernel threads and daemons do not) and the Linux kernel must track how that virtual memory is mapped onto the system's physical memory.

Processor Specific Context: A process could be thought of as the sum total of the system's current state. Whenever a process is running it is using the processor's registers, stacks and so on. This is the processes context and, when a process is suspended, all of that CPU specific context must be saved in the task_struct for the process. When a process is restarted by the scheduler its context is restored from here.

Linux Threads

A new process is created in Linux by copying the attributes of the current process. A new process can be cloned so that it shares resources, such as files, signal handlers, and virtual memory. When the two processes share the same virtual memory, they function as threads within a single process. However, no separate type of data structure is defined for a thread. Thus, Linux makes no distinction between a thread and a process.

1.4.3 File Management Subsystem

In Linux, as it is for Unix, the separate filesystems that the system may use are not accessed by device identifiers (such as a drive number or a drive name) but instead they are combined into a single hierarchical tree structure that represents the filesystem as a single entity. Linux adds each new filesystem into this single filesystem tree as they are mounted onto a mount directory, for example / mnt/cdrom. One of the most important features of Linux is its support for many different filesystems. This makes it very flexible and well able to coexist with other operating systems. The most popular filesystem for Linux is the EXT2 filesystem and this is the filesystem supported by most of the Linux distributions.

A filesystem gives the user a sensible view of files and directories held on the hard disks of the system regardless of the filesystem type or the characteristics of the underlying physical device. Linux transparently supports many different filesystems (for example MS-DOS and EXT2) and presents all of the mounted files and filesystems as one integrated virtual filesystem. So, in general, users and processes do not need to know what sort of filesystem that any file is part of, they just use them.

The block device drivers hide the differences between the physical block device types (for example, IDE and SCSI) and, so far as each filesystem is concerned, the physical devices are just linear collections of blocks of data. The block sizes may vary between devices, for example 512 bytes is common for floppy devices whereas 1024 bytes is common for IDE devices and, again, this is hidden from the users of the system. An EXT2 filesystem looks the same no matter what device holds it.

1.4.4 Device Drivers

Device drivers make up the major part of the Linux kernel. Like other parts of the operating system, they operate in a highly privileged environment and can cause disaster if they get things wrong. Device drivers control the interaction between the operating system and the peripheral devices that they are controlling. For example, the filesystem makes use of a general block device interface when writing blocks to a disk. The driver takes care of the details and makes device specific things happen. Device drivers are specific to the controller chip that they are driving.

Check Your Progress 1

1) What are the features of Memory Management subsystem?

....

....

....

2) What are the different states of a Linux operating system?

....

....

....

3) What is the purpose of a file system?

....

....

....

1.5 SUMMARY

Linux is an Unix like operating system, with the major difference that its source code is freely available. In this unit we have described the strong features of the operating system and also highlighted its drawbacks. Linux like any other operating system is made up of a number of functionally separate précis. The kernel is the most important component of the system. It deals Memory Manager, File Manager, Process Manager and I/O Manager. Device Drivers make up the major part of the Linux kernel. They control the interaction between the Operating system and peripheral devices. One of the main objective of Memory Management is to make available to the process a large amount of Memory although it is having a small physical memory through virtual memory concept. Although virtual memory allows processes to have separate address space it also provides shared space among many processes. The unit also describes the process and Thread Management.

1.6 SOLUTIONS/ANSWERS

1) The following are the important features:

   - Protection of a process
   - Address mapping from virtual to physical
   - Swapping of a process between Hard disk & Physical memory.
   - Providing a shared space among many processes.

2) Linux processes have the following states:

   - Running
   - Waiting
   - Stopped
   - Zombie

3) The file system provides a user a complete view of files & directories held on the hard disk of the system regardless of the file system type or characteristics of underlying physical device. Linux system supports many different file systems and presents all of the mounted files & filesystems as one integrated virtual filesystem.

1.7 FURTHER READING

Operating Systems, 4th Edition, William Stallings, PHI.

UNIT 2 LINUX COMMANDS AND UTILITIES

Structure **Page Nos.**

2.0 INTRODUCTION

This unit introduces you to Red Hat Linux 9 (hereinafter referred to as Linux) and tells you how to start working on your Linux computer. A few elementary commands are all you need to get the feel of what working in a Linux environment is like. The attempt is to let you see enough Linux features to allow you to walk up to a computer running Linux, login and start working on it. However, in this block we will not touch upon the design of Linux. This unit is oriented towards acquainting you with the richness of the system and making you comfortable with the Linux environment. Apart from the academic, theoretical and sociological aspects of the development of Linux, it is undoubtedly a rich, open and otherwise convenient operating system that gives you a bewildering array of tools to be productive. Unlike the earlier versions of UNIX, Linux is not open to the charge of being arcane and difficult to use. It is growing in popularity all over the world and is all set to enter the mainstream of corporate computing. If you are accustomed to an operating system like Microsoft Windows, then you might find Linux a bit different in terms of look and feel and as far as the commands go. However, these days the actual operating system you use is less at the forefront than it used to be, say, a decade ago, unless you are a software developer. The focus has now shifted more and more to the user and the facilities that the system can give him.

2.1 OBJECTIVES

After going through this unit you will be able:

- to start and carry on a login session under Linux;
- to change your account password;
- to understand basic Linux concepts like the hierarchical directory structure;
- to understand the various types of files under Linux, and
- to close a login session.

2.2 ENTERING THE MACHINE

We will now start learning how to use a Linux computer. This unit will talk about the basic steps involved in logging on to a system running Linux and also what you can do once you have gained ingress. But remember that you cannot learn Linux by reading this unit or even this block. That might at best allow you some familiarity with the terminology used and might tell you something about its organisation. You might even come to know something about its features and the tools available under it. But you will not be able to work expertly on a Linux computer or feel comfortable in a Linux environment, much less be productive in it. You will not be able to appreciate the power and beauty of Linux, nor marvel at the collaborative approach that produced it. The only way to learn Linux is by working on a real Linux machine. This unit, this block and other supplementary reading material, together with the Linux documentation and the many excellent books on the subject, will be a valuable aid in your voyage of discovery. So you must gain access to a working Linux machine and try out whatever you feel like. Do not be afraid of exploring or making mistakes.

Whenever you learn about a command or any other feature, do not hesitate to try out all its variations. Do not confine yourself to only what is mentioned here. This block is of necessity brutally brief and can only serve as an incomplete introduction. Use any other material to which you have access and experiment to your heart's content. You will learn as much from your mistakes and from seeing unexpected outcomes as from things you do by the book.

2.2.1 User Names and Groups

Every Linux user is given a name when she is allowed access to a Linux system. This is also called an account, as in commercial arrangements an account is kept of the usage of the machine by each user. The user name need not have any relation to the actual name of the user, though it quite often is some abbreviation of the name. For example, a person called Ram Kumar might be given a name kumarr on a Linux system. Here the account name is formed by taking the surname (abbreviated if it is too long) and the first letter of the first name. It is quite possible for a person to have more than one account on a single machine (in a different name), especially if the person uses the machine in more than one capacity.

Another way of making account names is based on the role being played by the person. For example, there can be several people working on some programming projects. Ram Kumar could be working on two projects with different teams. In such a case he might have an account name like crypt02 for his cryptography project, while for his natural language processing project his account might be nlp04. Such an arrangement helps to keep people in teams part of the same group. One of the motivations for Linux was to allow easy sharing of information, consistent with the needs of security and privacy. So Linux allows account names to be grouped under a common group name. All users belonging to the same group can share group privileges. If Ram Kumar leaves the organization and Zafar Khan takes his place, he might be assigned to continue work on crypt02 while somebody else might be assigned to nlp04.

Some user names are reserved by Linux for its use, for example, bin and uucp. So you cannot use those names for yourself. There is also a special kind of user on every Linux system who has all possible access rights on the system. This user is called the superuser, the system administrator or simply root because that is the user name conventionally allotted to her. For administrative convenience large systems can have more than one superuser account. The superuser is the one who can create new user accounts, shutdown the system and perform other maintenance tasks.

You would by now be wondering why everybody cannot access the machine as root. The reason is that when you are granted access to a machine you are given a password as well as an account or user name. You are free to choose your password though there are usually certain constraints imposed in the interests of security. So you cannot enter the system as root unless you know the root password. On any well maintained installation the root password is guarded very carefully, as public knowledge of this would jeopardize the security of the installation.

While root can access all user files and override any system protection meant for mere mortals, nobody can figure out what your password is. However, root can change or remove your password. A user can also change her password though there are usually some constraints on this too. There can be a minimum period before which you cannot change your password. After a certain maximum period you might be forced to change your password. There can also be constraints on what your password can be. There can be rules that force you to have a minimum password length, include special characters, disallow your previous five passwords and so on.

All the above constraints are meant to make your password difficult to guess. This is needed to make it hard for anyone else to masquerade as you and gain unauthorized access to the machine. Computer security is a matter of great concern to all of us as we become more and more dependent on them for performing our day-to-day tasks. Remember that on the machine your identity is determined by your user name and the machine cannot usually distinguish between physical individuals. However, for high security applications such as military work, access to a machine might be through the use of biometric methods like retinal scans or fingerprints. For most daily applications, however, passwords are still the only means of authentication.

☞ **Check Your Progress 1**

1) Can more than one person use the same user account on a Linux system?

....

....

....

2) Can there be more than one account with the same name on a Linux system?

....

....

....

3) Can more than one account have the same password?

....

....

....

2.2.2 Logging In

You will now see how to enter a Linux system so that you can start to use its facilities. This process of gaining access to a system is called logging in. For this you must have a valid user account on the machine and also know your password. In an organizational environment, this set up would have been performed for you by the system administrator of the machine when you were granted permission to use it. In this block we will assume you are working in some organization. If you are running Linux on a personal machine, then you would have to do all the set up activity yourself, or get somebody to do it for you. Of course, it does mean that you can grant yourself all authority and permissions on the machine, something that is usually not possible in an organizational context.

There are different ways of connecting to a Linux machine. You could go to the console of a Linux machine, such as a personal computer (PC), start it up and log in. You could be working on a Microsoft Windows machine or even another Linux machine and could connect over the network (whether local or a wide area network) to a Linux machine. Having once gained access you have all the facilities allowed to you by the administrator. There are only a few situations where physical access to the machine makes a difference. In this block we will not dwell on those matters.

When you see the console of your Linux machine or connect to it over the network, you see a message like:

```
IGNOU Linux machine
Kernel 2.4.20-8 on an i686
login:
```

The actual message on these lines could be different or absent depending on the installation. It could be longer, shorter or even be absent. This does not affect anything else that you do in any way.

You should now type in your user name and press the RETURN or ENTER key. In most cases you have to press the RETURN key, sometimes labelled as ENTER, to register what you have typed. You will also find that as you type on the console you will be able to see whatever you have typed. This is because Linux usually echoes whatever you type on the terminal. So your screen should now look like this:

```
IGNOU Linux machine

Kernel 2.4.20-8 on an i686

login: kumarr

Password:
```

You must type in your user name, also called the login name, exactly as allocated by your system administrator. This is because Linux is case sensitive, that is, it distinguishes between lower and upper case letters. In this respect it differs from operating systems like VMS or Microsoft Windows. So be careful of small and capital letters when working on Linux.

When Linux asks you for your password, key it in carefully. Notice that your password is not echoed as you type. In fact, the cursor does not move at all. This is to prevent somebody from reading your password over your shoulder, as that would enable that person to masquerade as you by logging into the computer in your name and using it.

Linux now checks whether you are a valid user and whether you entered the correct password. If there is any mistake you get a message saying:

```
Login incorrect

login:
```

This means you can try to login again. There can be other reasons why you might not be able to login even though you are a valid user and did not make any typing mistake. The messages you get in those situations will however be different.

Why be so pessimistic? Let us assume you have managed to login successfully. The system may then display some messages and finally give you a sign that it is now ready to obey your commands. The messages you see depend on how the system has been configured or set up by the system administrator and by you. So you might not even see any messages. However, usually there is a message indicating when you logged in last. This is useful because if the date and time differ from what you remember about your last login, it could mean that somebody else is using your account.

Let us now look at some of the other common types of messages you see on most systems as you login. These usually give some information about the system like the space available on the machine, news about the system and whether you have any mail. The news is called the message of the day and appears whenever you login. The message:

```
You have mail.
```

Means someone has sent you mail using the user-to-user communication facilities available in Linux.

After the login messages you see a prompt, which is the sign that Linux is ready for your commands. The prompt can be changed to whatever you like but the default prompt also depends on what shell you have been assigned. The usual default in Linux is the Bourne again shell, called bash, which is normally set to have the following prompt:

```
[kumarr@linux kumarr]$
```

This is the prompt we will use throughout the block unless some other prompt is explicitly called for. When you see the prompt on your terminal it usually means that Linux has finished executing the last command you gave it and is ready for your next command. Here kumarr is your login name, Linux is the name of the machine and the home refers to the directory in which you are located after you login. This is usually your home directory.

There can be limits on the number of attempts, say five, that you can make at logging in. The action taken depends on the installation but can be alerting the system administrator or deactivating the terminal, perhaps for a short time only. So you should be careful not to make too many typing mistakes. In particular be careful not to forget or mistype your password and avoid passwords with certain characters like #.

Now you are still at your console that is black and white. If you want to use the graphical features of Linux you need to get the X Window system up. For this you need to issue the command:

```
[kumarr@linux kumarr]$ startx
```

whereupon the X Window system will start up and you will see a coloured screen with a bar containing several icons at the bottom. The background and the colour of the screen as well as the icons and facilities available in the tray depend on the configuration of Linux that has been performed during installation or later. You will usually want to work in the X Window system rather than directly on the console as you can then use the mouse and other graphical facilities available in Linux. At this point you are presented with your desktop.

Once here you can open up a terminal window by right clicking the mouse and selecting the appropriate option. You can have as many windows as you like and you can be doing different tasks in different windows. It is not that only terminal windows can be opened up. You can click on the icons on the desktop or in the tray and run any applications, such as a browser, office productivity tools and so on.

You can have several desktops. In your tray you will see four rectangles that represent four available desktops to you. All the windows and applications that you are running are associated with the desktop in which you open them. If you want a clean slate where you are doing some other related tasks, you can click on another rectangle and go to that desktop. This is very convenient if you want to do groups of tasks and do not want to clutter up one desktop with too many unrelated windows.

☞ **Check Your Progress 2**

1) What happens if you type in your login name all in upper case?

....

....

....

2) Try logging in by using a friend’s account (do not ask him for the password). Are you able to get the prompt?

....

....

....

3) Try logging in using an account which does not exist on your system (confirm this from your system administrator). Is there any difference in the computer’s response from that in the last exercise? Why do you think this is so?

....

....

....

4) Try using your mouse just after logging in at the console when you are in text mode. What happens? Are you able to perform any operation with the mouse?

....

....

....

2.2.3 Correcting Typing Mistakes

Many of us are not professional typists and we make a lot of mistakes while typing. In any case all of us are human beings and are prone to error. Whether you are a one or two finger expert or know touch typewriting, you are going to mistype your commands some time or the other. What do you do when you want to find out in a session whether you have any Sundays left in the month? Normally you would use the cal command thus:

```
[kumarr@linux kumarr]$ cal
```

Suppose now that by mistake you type:

```
[kumarr@linux kumarr]$ csl
```

After you press the RETURN key Linux will say:

```
-bash: csl: command not found
```

if you are lucky and a command csl does not exist. If it does it will be executed and you could well be in deep trouble depending on what csl does.

You would therefore do better to cancel your command or correct your mistake. These actions can be accomplished by using the kill and erase characters respectively. The kill character cancels the entire line you typed while the erase character erases or rubs out the last character.

Usually the erase character will be the character ^H. This means that you have to type H while holding down the CONTROL key, often abbreviated to Ctrl. This key is normally located on both sides of the keyboard near the Shift keys. In this block we will use the convention of writing ^H to mean Ctrl-H and you must be careful not to confuse this with the two separate characters ^ (circumflex) and H. The backspace key generates the sequence ^H on the keyboard and you can use it to delete the character preceding the cursor.

But Linux offers you other facilities to make typing easier. What if you want to have typed 8 characters and want to change the 5th? Just use the left arrow key to go to the character you want to change, press the Delete key and type in the correct character. You can use the right arrow key to go to the right in a command. If you want to repeat a command you issued 3 commands earlier, you can use the up arrow key. Pressing it once brings up the last command you issued, so press it thrice to get the command you issued 3 commands earlier. Then use the RETURN key as usual to invoke the command, that is, to ask the computer to execute it. You can likewise use the down arrow key to go forward through your command history.

What is more, once you have reached one of your old commands you can edit it by using the left and right arrow keys together with the backspace or delete keys. This is useful if you want to rerun a command with some other options. For example,

```
[kumarr@linux kumarr]$ cal 2004
```

will give you the calendar for the year 2004. If after issuing a few more commands you want to see the calendar for the year 2003, you do not need to type in the full command again. Just use the up arrow keys to locate the command and to change the 4 in 2004 to 3. Then hit the RETURN key. What a boon for typing challenged people like this author!

You can use the command:

```
[kumarr@linux kumarr]$ history
```

to see the previous commands that you entered. To run a command entered a long time ago, instead of using the up arrow key, you can say !<*no*> where *no* stands for the command number. To run a command *n* commands previously, say

```
[kumarr@linux kumarr]$ !-n
```

☞ **Check Your Progress 3**

1) What are the characters you need to use to correct typing mistakes while logging in? How do you correct typing mistakes while entering the password?

....

....

....

2) How many previous commands can you invoke by using the arrow keys?

....

....

....

3) How will you enter a '\' in the command that you invoke?

....

....

....

2.2.4 Format of Linux Commands

We will now be at the general format of Linux commands and take the opportunity to study some simple commands. Let us go back to the bash shell and the cal command that we mentioned earlier.

```
[kumarr@linux kumarr]$ cal
```

This gives the calendar for the current month and year (of course this will depend on what the system date has been set to - that is what the computer believes the current month and year to be), and you can use it to, say, find out how many sundays are left in the month. Another simple command is

```
[kumarr@linux kumarr]$ date

Wed Oct 13 09:20:59 IST 2004
```

which displays the current system date and time. You will realize that the computer has no way of knowing what the current date and time really are, so what it can tell you is whatever it thinks is the current time and date. This can be set by the system administrator to be almost anything, but in most installations, care is taken to see that the date is set correctly. This is because today computers are in general networked with other computers and many system utilities depend on the correct date and time. In Linux when we say date, we mean both the date and time, which is why the output shown above includes the time as well.

Notice that the time zone is part of the output. This is significant when you are on a network spanning time zones.

Another simple one word command is:

```
[kumarr@linux kumarr]$ who

kumarr  tty1         Oct 13 08:41

kumarr  pts/0        Oct 13 08:54 (:0.0)

khanz tty2           Oct 13 07:39
```

This tells you the names of the users currently logged in to the system, their terminal numbers and the date they logged in. You will find that you will always be listed as one of the users, since you usually run the commands only when you are logged in to the machine. There is another form of the who command that you can now try out.

```
[kumarr@linux kumarr]$ who am i

kumarr   pts/0        Oct 13 08:55 (:0.0)
```

This time we have given the arguments am i to the actual command who. The result is similar to that obtained earlier, but now you are the only user listed. This command has the effect of telling you the login name of the user currently logged in at that terminal, the number of the terminal and the date the user logged in. Other users of the system at that time are not listed.

Arguments to the command are separated from the command by one or more spaces. It might seem silly to ask the computer who you are, but if the previous user has not terminated his session, you can find who it was by this command. But you would do well never to leave your terminal unattended while you are logged in, as it would be a security lapse. Actually Linux also provides the command:

```
[kumarr@linux kumarr]$ who are you

kumarr   pts/0        Oct 13 08:56 (:0.0)
```

which is synonymous with who am i, but sounds much more intelligent.

You have now seen the general format of Linux commands, which comprises the basic command followed by zero or more arguments. The command and the various arguments are separated by one or more spaces and the whole sequence is terminated by the newline character, which is produced when the ENTER key is pressed.

You can enter more than one command on the same line by separating the commands from one another with semicolons like this:

```
[kumarr@linux kumarr]$ date; who

Thu Oct 14 00:08:28 IST 2004

kumarr   tty1         Oct 13 23:51

kumarr   pts/0        Oct 13 23:59 (:0.0)
```

The commands are executed one after the other in the order they were specified on the command line. After the last command is over you get the prompt again.

Arguments to commands should not contain spaces otherwise the different words of the argument would be interpreted as different arguments by the computer. If for some reason the argument needs to contain a space, you must enclose the argument in double quotes (“) or in single quotes (’).

Most arguments to commands are filenames (discussed later in this unit), options or expressions. All of these could occur in the same command. The exact order in which these arguments are listed can depend on the command and should be ascertained by examining the documentation for that command. Usually options immediately follow the command with the expressions and filenames coming next. You will see details of such cases later when we study more complex commands than the ones we have looked at so far.

If an argument itself contains quotes of one kind you can enclose it in quotes of the other kind. Thus:

```
[kumarr@linux kumarr]$ grep -n "Ram Kumar's Salary" employee payroll
```

looks for the expression Ram Kumar's Salary in the files employee and payroll and prints the line numbers of the lines in which the expression is found.

Sometimes the bash shell places restrictions on the use of certain characters because it interprets them in some special way. To use these characters in arguments, you have to use quotes. The details of bash, the shell, are discussed in Unit 4.

☞ **Check Your Progress 4**

1) How can you get the calendar for some other month in some other year?

....

....

....

2) Get the calendar for the year 1752 and look at it. Is anything the matter?

....

....

....

3) Find out how to set the system date. Why do you think only the super user is allowed to do this?

....

....

....

4) Study the who command and use it to find the date the machine was started up, and also how many users are currently using your system.

....

....

....

5) Open up terminal windows and applications on all four desktops and use the who command. What do you see? Are you considered to be the same user on all desktops?

....

....

....

2.2.5 Changing Your Password

You saw earlier that your password was the only way of preventing somebody from using your account on the system. Without it anybody who knew your login name could walk up to the machine and start using your account. This would be really serious in the case of the superuser or root.

When you are first given your account you are told what your password is. This is usually some default convention used by the system administrator, though not a good practice. It should be a different password generated for each new user, otherwise you are inviting a security breach. You are then invited to change your password when you first log in. This can be done with the command:

```
[kumarr@linux kumarr]$ passwd

Changing password for user kumarr.

Changing password for kumarr

(current) UNIX password:

passwd: Authentication token manipulation error
```

Notice that unlike the commands you saw so far, the passwd command is interactive. It asks you to enter some information rather than doing all the work by itself. The first thing it asks for is your current (or old) password. This is to make sure that somebody else cannot change your password while you have left your terminal unattended. If the wrong password is entered here, you get a message as shown above and you get back the prompt. You can then of course try again:

```
[kumarr@linux kumarr]$ passwd

Changing password for user kumarr.

Changing password for kumarr

(current) UNIX password:

New password:

Retype new password:

passwd: all authentication tokens updated successfully.
```

If you now enter the old password correctly you are asked to type in your new password. After that you are asked to enter it again. If you type two different things here, the system tells you that they do not match and asks you to try again. If you keep getting mismatches, the command terminates with the message passwd: Authentication information cannot be recovered

The number of retries allowed is configurable by the system administrator and is usually kept at 3. This is because if you are unable to change your password you are unlikely to be able to enter it correctly later to login. Also notice that none of the passwords is echoed. Your system will probably have restrictions on what passwords you can choose. The password should not be too short or too long to remember. You should change it periodically so that if someone has been using your account by laying hands on your password, they cannot continue to do so indefinitely.

If you are wondering how the passwords are stored on the machine such that even the superuser cannot find out what your password is, the answer is that Linux encrypts your password before storing it. This means that what is stored on the computer bears no resemblance at all to what you typed in as your password. When you try to login the next time, Linux again encrypts the password you type in and compares it with what has been stored. If the two are the same, you are allowed to login, otherwise your attempt is blocked. No ordinary user can even read the encrypted password – only the superuser can do so. So no one can find out what your actual password is – at least, not easily.

In Linux, you cannot change somebody else's password, even if you know what it is, from within your account. If you try to do so, you are told:

passwd: Only root can specify a user name.

How then can root change a user's password without already knowing it? Ah! When the user executing the passwd command is root or the superuser, Linux does not ask it to supply the old password. That is why root can set your password to anything without knowing what it is currently.

Since your password is the only way of protecting your account, you must take care to choose passwords well, that is, choose one which cannot be easily guessed. As a general rule, do not write down your password anywhere and let it be locked up in your head. Your Linux installation will probably enforce some rules on what you can set the password to be, or even the intervals at which you can change it.

Here is a short excerpt from the Linux manual entry for the passwd command about choosing paswords.

Don't write down your password - memorise it. In particular, don't write it down and leave it anywhere, and don't place it inan unencrypted file! Use unrelated passwords for systems controlled by different organisations. Don't give or share your password, in particular to someone claiming to be from computer support or a vendor. Don't let anyone watch you enter your password. Don't enter your password to a computer you don't trust. Use the password for a limited time and change it periodically.

Your password should be hard to guess, and so you should not use information about yourself that is easily available. This includes your name, that of your family, or anything to do with your vehicle, credit card and so on. Also avoid dictionary words. It would be reasonably safe to use a combination of upper and lower case letters and special characters, with a length of at least 8 characters.

☞ Check Your Progress 5

1) Can you run the passwd command again and set your password to what it already is?

....

....

....

2) Can a friend (not the superuser) help if you have forgotten your password?

....

....

....

3) Find out any restrictions in force at your installation on what you can set the password to be.

....

....

....

4) What does Linux have to say about choosing passwords? Find out from the documentation.

....

....

....

2.2.6 Characters with Special Meaning

Some characters are interpreted in a special way by the shell. These meanings will be discussed in detail in the next unit of this block, but that apart, there are certain characters you will find to be useful.

For example, suppose you start a command that takes a long time to execute, and you change your mind and do not want to wait for the command to finish. You can abandon or break a command in between by pressing the BREAK character, which is set to ^C. Again, consider a command which produces a lot of screen output. This could happen if you were typing out a long file, for example. The output will probably be dumped on your terminal far too fast for you to read. To stop output on the screen temporarily, hit ^S. You can press any other key to restart the output.

A special character that you can use to erase a command line that you have typed is ^U. This is useful when you just want to start over rather than correcting some small mistake.

Another special character can be used to terminate your login session. While you can do so using the exit command, you could also try the special character ^D. This indicates to the shell that there will not be any more input from you. So Linux logs you out and again displays the login message on the console for another user, or you again, to start another login session. If you are in the X-Window mode and are in a graphical terminal window, the window is closed if you logout.

☞ **Check Your Progress 6**

1) The commands you have learnt so far produce only a small amount of screen output. How will you produce output which does not fit into one screen, using only the commands you have learnt so far?

....

....

....

2) How much control does the ^S command allow you over the output? Can you read a long file in sequence easily by this method?

....

....

....

2.2.7 Linux Documentation

Linux comes with copious documentation that is mostly available on line. You should, as a user, learn how to use the Linux manuals and other resources. While we will not be able to discuss this topic in detail here, you will have to acquire this skill if you want to obtain a good understanding of Linux. This is because in this block we do not have the space to consider any but the most basic commands, and even those only briefly. We will not even be able to consider all the options available with many of the commands that we do discuss. The only way for you to master them will be by consulting the documentation.

Usually the documentation will have been installed on your machine when Linux was set up. In that case you can look up the manual entry for a command by using the `man` command. For example, to learn more about the `who` command than what we have talked of, saying:

```
[HYPERLINK "mailto:kumarr@linux"kumarr@linux kumarr]$ man who
```

You can similarly learn more about the date, cal or any other command. So to learn more about the man command itself, say:

```
[kumarr@linux kumarr]$ man man
```

For many commands you can use the `info` command to get more complete information. There are several other resources available, for example you can look at the URL:

https://www.redhat.com/docs/

that has a host of documentation on Red Hat Linux. You can use a search engine to look for Linux documentation on the Internet, to find user groups and so on. These days it is less common to use printed manuals, primarily because they tend to get outdated so quickly! Besides user level documentation, there are many resources that you can find for system administrators and programmers, as you get to a more advanced level in Linux with practice and experience. So use this block as a quick introduction to get your feet wet and then move onto the more detailed documentation available.

☞ **Check Your Progress 7**

1) Look up the manual entries for all the commands we have studied so far. What do you feel about the number of options available with each command?

....

....

....

2.3 THE FILE SYSTEM

In this section we will explore the file and directory structures of Linux. Just as a paper file is something into which you can put papers and bunch a group of papers together, a Linux file is something into which you can put data. A file has a name, and this name is the property of the file rather than the data present in it at any given time. It is possible to change the data in a file. This act does not affect the name of the file. Thus Linux commands can be made to operate on the data in a file as a group.

A file usually exists on the hard disk(s) of the computer. This will be the case when you are logged in to the machine and are engaged in a session. The actual areas of the hard disk used by a file can change as the file is increased and decreased in size. As you will see later, the size of a file in Linux has a precise technical meaning, and the size of a file does not necessarily tell you the actual amount of data in it.

Linux has three kinds of files – ordinary, directory and special. You have already got an idea of what ordinary files are. Special files will be discussed in Unit 5 of this block.

Directory files contain information about other files, including directories or special files. A directory groups its contents together hierarchically under itself, and a directory within a directory is called a subdirectory of the directory at the higher level, also called the parent directory. Thus a Linux file system is like an inverted tree of directories, starting at a root and going down to an arbitrary depth of hierarchically arranged levels.

We will now look at some of the files in Linux, learn how to navigate the file structure and how to make use of it.

2.3.1 Current Directory

Every user who is given an account on a Linux system is also given a directory where he reaches on logging in. This directory is also called the home directory. The current, working or current working directory is the directory in which you are currently located. On logging in, your current directory is normally your home directory. You can find out what your current directory is at any time by using the command:

```
[kumarr@linux kumarr]$ pwd
/home/kumarr
```

This means that your current directory is called kumarr and is located under the directory home, which is in turn located under the root directory. Of course the actual home directory you are allotted will depend on the installation. By the way `pwd` is one of the few Linux commands that do not take any arguments or options.

The output that `pwd` displays is called the full pathname of your current working directory. This is also known as the complete or absolute pathname, that is, the pathname starting from root. You can refer to your directory by just saying `kumarr`. But this is not unambiguous as there can be another directory called `kumarr` under some other directory as well. But no two directories or files on the same Linux machine can have the same complete or full pathname. The various components of the path are separated from one another by slashes (‘/’).

We have not yet talked of what a valid filename can be. Actually in Linux there are no restrictions and a filename can have any characters and any length. The same rules apply to directories as well. In practice it is best to avoid certain characters in filenames because they have special meaning to the shell.

☞ **Check Your Progress 8**

1) What would happen if your home directory did not exist and you tried to login?

....

....

....

2) How would you put a space into a filename?

....

....

....

2.3.2 Looking at the Directory Contents

We will now see how to look at the contents of a directory. The command is:

```
[HYPERLINK “mailto:kumarr@linux”kumarr@linux kumarr]$ ls
```

This gives you a listing of all files in the current directory. If you have just been allotted your account and are logging in for the first time, you will be in your home directory and that directory will be empty, that is, there will be no files in it.

Over the years the `ls` command has accumulated a lot of options and it takes some time and experimentation to understand them all. The first option we look at is:

```
[kumarr@linux kumarr]$ ls -a
```

Here `a` means all. This is your first taste of Linux options, so look at the command carefully. The command ls is followed by at least one space after which the hyphen or minus sign introduces the option letter. The -a option tells Linux to list all files including those that are "hidden". Hidden files are those that start with the '.' character. Unless the -a option is used, ls never lists such files in its output. The output of ls is always sorted in some order, the default order being alphabetical. This sort order can be altered by other options to ls which we will take up later. This is why the file (actually a directory) '.' is listed before '..' in the output.

```
.                 .emacs              .gtkrc-1.2-gnome2    .netscape6
..                .esd_auth           .ICEauthority        .openoffice
abc               _files              ignou                .qt
abc\d             .fonts.cache-1      .kde                 .recently-used
abc d e f         .gconf              .mailcap             .rhn-applet.conf
.bash_history     .gconfd             .mcop                .sversionrc
.bash_logout      .gnome              .metacity            .user60.rdb
.bash_profile     .gnome2             .mime.types          .Xauthority
.bashrc           .gnome2_private     .mozilla
.chromium         .gnome-desktop      .nautilus
.chromium-score   .gtkrc              .netscape
```

You will usually find that directories are listed in a different colour such as blue. This helps you locate them quickly. The '.' refers to the current directory and '..' to its parent. These are pronounced dot and dot dot respectively. In this case '.' refers to /home/kumarr and '..' to /home. The directory '/' or root is its own parent. This output is of course not very interesting because your home is devoid of files created by you and you do not yet know how to create any. The only files you see are hidden files made by the software itself.

To get around this, let us look at some other directory. You can get the listing of any directory by supplying its name as an argument to ls. Thus to look at the directory listing of the root directory, use the command:

```
[kumarr@linux kumarr]$ ls /

bin     dev     home    lib           misc  opt   root  tftpboot usr
boot    etc     initrd  lost+found    mnt   proc  sbin  tmp      var
```

Here the output is sorted column wise. We must caution you that it is quite likely that you will see a different listing than the one shown here. It is self evident that the listing will depend completely on the machine you are working on. However there are some files that will surely exist on the root directory of a working installation. The above directories are such files. To sort the output according to rows, from left to right, say:

```
[kumarr@linux kumarr]$ ls -x /
```

As you have seen the ls command lists several columns in its output. This is easy to change. If you want 1 column in the output, say:

```
[kumarr@linux kumarr]$ ls -1

bin

boot

dev
etc
...
```

Another variation of the command is the -C option that might produce the same output as the command without an option. If that happens it means that the actual command has been configured to use the -C option by default.

Besides using the colour of the file to identify directories, you can use the -p option or the -F option to append a '/' to every file that is a directory. The -F option also appends a '*' to every file that is an executable file, that is, a command. Check this out and understand how the -F and -p options differ.

```
[kumarr@linux kumarr]$ ls -Cp /

bin/   dev/  home/    lib/         misc/  opt/   root/  tftpboot/  usr/

boot/  etc/  initrd/  lost+found/  mnt/   proc/  sbin/  tmp/       var/
```

If you repeat this command on the bin directory, you will see something different.

```
[kumarr@linux kumarr]$ ls -Cp /bin

arch            df              hostname  nice            su

ash             dmesg           igawk     nisdomainname@  sync

ash.static      dnsdomainname@  ipcalc    pgawk           tar

aumix-minimal   doexec          kbd_mode  ping            tcsh
```

You see some files in a different colour. These are executable files, or commands that you can run. You can use the -F option to verify that they are so because each of them will have a * appended to the name in the listing.

When you have a very large directory, you might want to use an option of ls that gives a very compact listing.

```
[kumarr@linux kumarr]$ ls -m /bin

arch, ash, ash.static, aumix-minimal, awk, basename, bash, bash2, bsh, cat,

chgrp, chmod, chown, cp, cpio, csh, cut, date, dd, df, dmesg, dnsdomainname
```

The colours of the entries are as usual but each entry is separated from the other by a comma.

The above examples show you that the root directory consists of both directories and ordinary files. In Linux everything is considered to be a file. The directories here, or anywhere else, can themselves contain other directories to any depth. To see the contents of /home, you can say:

```
[kumarr@linux kumarr]$ ls -xp /home

kumarr/ khanz/
```

You might surmise that you are seeing the home directories of users who have accounts on the machine. You also see your own home directory here. You can tell that they are directories by the colour as well as the / that is appended to the name. But there is one thing that you need to note. When you had logged in and checked your current directory, the result had been:

```
[kumarr@linux kumarr]$ pwd

/home/kumarr
```

which is different from what you see here. Why is this so? We have seen in the last section that the pwd command tells us the full, absolute pathname of the current working directory. When we look at the contents of /home, kumarr is merely one of the directories under it and is shown as such. To get the complete pathname we must specify the preceding portion which is /home. Thus the full or complete pathname is / home/kumarr.

If you look at the directory listing of /usr, you will find a bin directory under it too. By now you will have understood that this bin directory is different from the one you saw under the root directory. The first has the full pathname /usr/bin, while the second has the full pathname /bin. You should now look at the contents of the other directories and try specifying their complete pathnames. You can also try looking at their contents by specifying relative pathnames. We will look at complete and relative pathnames again in the next section. You would do well to understand pathnames, relative and absolute, thoroughly as that will be useful in navigating around the directory tree.

But let us now get back to our friend the ls command. One of the most useful and often used options is -l, which gives the so called long listing of the directories asked for.

```
[kumarr@linux kumarr]$ ls -l

total 24

-rw-rw-r—     1 kumarr   kumarr          4 Oct 15 01:18 abc

-rw-rw-r—     1 kumarr   kumarr          5 Oct 15 01:17 abc\d

-rw-rw-r—     1 kumarr   kumarr          5 Oct 15 01:17 abc d e f

drwxr-xr-x    2 kumarr   kumarr       4096 Aug 21 20:33 _files

drwxrwxr-x  3 kumarr   kumarr         4096 Oct  9 18:01 ignou

-rw-rw-r—     1 kumarr   kumarr         27 Oct 11 22:51 xx
```

Now this is a complicated looking output, so let us try and understand the meaning of this listing. The first column of the output tells you whether the file is a directory or not. A '-' means that it is an ordinary file while a directory has a 'd' in that position. So you now know another way of telling whether a file is a directory, apart from the -p or -F options and the colour of the listing that you have already looked at. The other 9 columns in the first field tell you about the permissions on that file. We will look at these in detail in a later section.

The next field in the output is a number indicating the number of links to the file. For a file this shows the number of names it has. In Linux the same physical data may have several names, although it must have at least one. Each name is a link to the file. Usually ordinary files have only one link, but if there are more it does not mean that there are that many copies of the data in the file. There is only one physical copy of the data that can be referenced using any of its names. In the case of directories the number of links tells you the number of subdirectories that it has.

The third field of the output shows the owner of the file. Root and bin are names reserved by Linux for its use as we have seen earlier. In some cases you might see a number like 207 instead of the user name.

The next field is the group name and in certain situations can be a number in the display. The user is a part of the group shown here.

The fifth field is the size of the file in bytes. You already know that the size of a file in Linux has a precise meaning which is unrelated to the amount of data in it. However, do not be alarmed because in most cases the intuitive meaning of size does hold good and the figures you see usually do represent the number of bytes of data in the file in question. For large files the size is difficult to comprehend when represented in bytes. For such cases, you can use the -h option that gives the size with a K, M or G following the number for kilo, mega and giga bytes respectively. These are true kB, MB or GB as the multiplier used is 1024. Often we colloquially use K to mean 1000 times. If you want the size represented with a multiplier of 1000, use the –si option. The size is then shown as kB, MB or GB after the number to help you distinguish which multiplier has been used.

The next item of information is the date the file was last modified, and in the end the name of the file is shown.

With the long listing of ls, you can find out many useful things about the file. Try looking at the directory long listing of the various system and other directories on your machine. In the course of this exploration, when you look at directories like /bin, /sbin and /usr/bin, you will see some familiar names such as who, pwd or ls. Thus ls is itself to be found in the /bin directory. The three directories we just mentioned are the ones where most of the binaries or executables of the system commands are to be found. There are some to be found under /etc. as well.

We will now look briefly at three other options to the ls command. When a directory is given as an argument to ls you get to see the contents of the directory. But suppose you want to check the permissions on a directory, say /home/kumarr. If you try:

```
[kumarr@linux kumarr]$ ls -l /home/kumarr
```

you will see nothing of what you need because ls tries to list the contents of the directory and at present there is no file in your home directory. In the examples above, we had created some files in the home directory of kumarr so as to be able to show you some output, and these files were then deleted. So to see the permissions you could say:

```
[kumarr@linux kumarr]$ ls -l /home
```

whereupon kumarr would be one of the entries. But this is awkward as you will have to wade through potentially several entries before you can locate the one you are interested in. The answer to this is:

```
[kumarr@linux kumarr]$ ls -ld /home/kumarr

drwx———   19 kumarr  kumarr      4096 Oct 16 16:48 /home/kumarr
```

which lists /home/kumarr as a directory and shows all the information about it.

By now you would have also realised that subdirectories are normally shown as single entries and any files inside them are not shown. To look at the contents of a directory and recursively of all subdirectories within it, use -R.

```
[kumarr@linux kumarr]$ ls -lR /usr
```

will show the contents of /usr and also recursively of every subdirectory inside it, down to ordinary files. Thus using:

```
[kumarr@linux kumarr]$ ls -lR /
```

you can see each and every file and directory on your system, though not hidden files.

Another option you might need sometimes is the -r option, for reverse. This reverses the sort order of files displayed by ls. You can try this with any option, such as:

```
[kumarr@linux kumarr]$ ls -lRr /
```

So far you have only given directories as arguments to ls, but you can use ordinary files as well. It then lists only that file if it exists. Moreover you can give any number of files or directories as arguments to ls and it will list whichever ones exist. You can also use wild cards here. The '?' character matches any single character, while the '*' matches any number of characters except a leading '.'.

If you feel out of breath after looking at these options, there are many more we have not looked at! You are encouraged to look up the documentation for ls and experiment with them. Many Linux commands have zillions of options – getting used to them all requires time and effort. But you will find that you soon get to know the options you use often. It is probably best, when learning a new command, to concentrate on a few useful looking options only. As you use them frequently, you will get to know them well. Then you can spend some time deepening your knowledge of the command by trying out the other options.

Most beginners get overwhelmed by the large number of options and do not know where to start or when to stop. You will have to work out a method that suits you. Maybe you are the type who likes to learn everything about a command at one go. But many people, including the author, find that building on a solid foundation of already known options is easiest.

☞ **Check Your Progress 9**

1) Read up on and try out the other options to ls. What is the output of ls -lm? Of ls -ml? Which option takes precedence? What is the result of ls -d?

....

....

....

2) If your system has the -x or -C option set by default, how can you get the standard ls listing?

....

....

....

3) How can you control whether you see different file types in different colours?

....

....

....

4) Find out how to sort the output on time rather than alphabetically.

....

....

....

5) Since you can have filenames that are of arbitrary length, would you prefer to store information in the filename or in the file?

....

....

....

2.3.3 Absolute and Relative Pathnames

You saw in the last section how pathnames could be relative or absolute. Since the Linux file system is logically structured like an inverted tree, it is important to understand how to specify pathnames. Both methods can be used and in Linux it does not matter which approach you use in identifying the file you mean, as long as you are careful about specifying it correctly. However there are situations where one or the other approach is more convenient. So you should take the trouble to assimilate the concept and learn how to navigate around the system with felicity. Let us look at a typical directory hierarchy on a Linux machine.

At the top is the root directory, under which are directories such as bin, boot, dev, home, lib, opt, home, etc, usr and so on. Under /dev you would have some 18 directories besides the ordinary files. Similarly /etc might have 63 odd directories under it and /home would have the home directories of different users. /usr could have about 13 directories that include bin, etc. and others.

The exact layout of the directory hierarchy on your machine is likely to be different. We will soon be looking at some of the important directories and files on a Linux system. For the moment, though, you would do well to just concentrate on learning how to move around. You already understand what is meant by the current directory. This is the directory in which you are located at any given time. If you issue the command ls, it is the files in the current directory that are brought up for you to see. If you login as kumarr, you will probably end up in /home/kumarr when you get your prompt unless things have been arranged otherwise.

Now suppose /home/kumarr has a directory called nlp that has a file called augcfg.C. Suppose you want to see the size of this file alone. For this you need to use the ls command and provide the filename as an argument to it. In Linux you can provide a pathname (relative or absolute) as an argument to a command wherever you could otherwise provide a bare filename. So you now have three ways of accomplishing what you want (we will assume that you have the required permissions— this will, in fact, be the usual situation) to do.

Let us first use an absolute pathname. So you have to specify the filename starting from root or '/'. Thus your command needs to be:

```
[kumarr@linux kumarr]$ ls -l /home/kumarr/nlp/augcfg.C
```

You have already used this method in the last section. The second way is to use a relative pathname, where you specify the pathname relative to where we are currently. Here you only need to recall that '..' stands for the parent directory of the current directory. So if you are at /home/khanz, you can say:

```
[kumarr@linux kumarr]$ ls -l ../kumarr/nlp/augcfg.C
```

The '..' takes you one level up, that is, to /home. From there you continue naming the file as before. Of course, you could have used the following rather convoluted way:

```
[kumarr@linux kumarr]$ ls -l ../../home/kumarr/nlp/augcfg.C
```

This is inefficient because you implicitly to root before naming the file. The first '..' takes you to /home and the second one level higher, to / or root itself. Then you begin your descent until you reach the file you desire. Here it would have been better to use an absolute pathname instead of this, for then you would not have had to use two steps to reach root.

Usually a filename is specified by the method that results in the shortest possible specification of the name, though of course that is not at all necessary. This depends on whether the filename is closer to you or to the root directory. If you are located in /home/khanz and want to specify a file in the directory /home/kumarr, it is easier to say ../kumarr rather than /home/kumarr.

There is a third way of looking at the size of augcfg.C. For this you will have to learn a new command, cd, which lets you change the current directory. This command can be given an argument which is your intended destination and it then changes your directory to what you asked, provided you have the appropriate permissions. And how do you specify your desired destination directory? By specifying, the pathname, of course. The pathname can be specifed, as you would have undoubtedly guessed now, either as a relative or absolute pathname. So you can say from /home/khanz

```
[kumarr@linux kumarr]$ cd /home/kumarr/nlp
```

or

```
[kumarr@linux kumarr]$ cd ../kumarr/nlp
```

and then look at the size by:

```
[kumarr@linux kumarr]$ ls -l augcfg.C
```

This really amounts to specifying the filename relative to /home/kumarr/nlp, the current directory. In general when you specify a bare filename you are specifying the filename relative to the current working directory. So the command above is really a shorter way of saying:

```
[kumarr@linux kumarr]$ ls ./augcfg.C
```

One form of the cd command can be very convenient if you have wandered far off your home directory and want to return there, especially if your home directory happens to far away from the root directory. This is:

```
[kumarr@linux kumarr]$ cd
```

without any arguments. It always brings you back to your home directory irrespective of where you are, even if you were already there to start with.

☞ Check Your Progress 10

1) Go to the root directory and then try to go to its parent with 'cd ..'. What happens? What do you conclude?

....

....

....

2) What happens if you try to cd to a non-existent directory?

....

....

....

3) Can there be a file under a directory with the same name? Why?

....

....

....

2.3.4 Some Linux Directories and Files

It will be interesting and useful to now get acquainted with the Linux system directory structure. We will look at the layout and contents of the Linux system directories and understand how the various system files are grouped under directories. We will also learn about the functions of some of the system files. The typical Linux directory structure is as described in the earlier section.

We again emphasize that only some of the system directories are shown here. Your machine could have a somewhat different organization. How will you find out the directory tree for your Linux system? You can do this by exploring the files on your machine.

As you have already seen, the / bin directory contains some of the Linux system commands and utilities. These include some of the commands that you have learnt so far, such as ls and pwd. You can look at the long listing of this directory and note the information provided. Look at the sizes to get an idea of the size of executables on your machine. These will depend, among other things, on the architecture of your computer.

The /dev directory contains device special files concerned with hardware devices like printers, mice, audio devices, storage devices such as floppy drives and CD-ROM drives and so on. You will learn more about these files and the /dev directory later in this block.

The /etc directory, as the name suggests, has several miscellaneous files and directories. It contains many files and commands that are reserved for the use of the system administrator. Many of the system defaults are set up using these files. Ordinary users cannot execute these commands or use these files. For example, look at /etc/issue or /etc/motd. The first contains the text that is displayed at your login prompt before you have logged in, while the second file (for message of the day) contains the text that you see just after you login. These files can be blank, and sometimes are. The file /etc/group has the names and group numbers of all the groups in the installation. The /etc/passwd file contains the login name of each user, his user identification number, his home directory, his default shell and sometimes some commentary. In Linux the encrypted password of each user is stored in a separate file called /etc/shadow that cannot be read by ordinary users. So you cannot see even another user's encrypted password.

The /lib directory contains system libraries that are used with compilers and shared libraries that are needed at run time for executing commands and running executables.

The /sbin directory contains some standalone commands and utilities used during installation. These are of interest to system administrators and those who need to install and maintain the system.

The /tmp directory contains temporary work files that might be created by utilities and commands when they run. It provides work space to such commands. This directory is cleared out periodically on many installations. In any case you should assume that any file in the /tmp directory can be erased without warning. So you should not store any useful files here and put them under your home directory only.

The directory /usr/bin contains useful and important commands and utilities for users. There is no sharp distinction between the commands in /bin and here, though. Our old friend, the cal command, is to be found here. /usr/include contains header files that are useful in writing C or C++ programs.

An interesting directory is /usr/games that traditionally contained text games in older Unix installations. Today it could contain some more sophisticated games, such as chromium or maelstrom.

The directory /usr/local/bin is often used as a repository of commands used locally and frequently developed by local talent. Such commands are often those that are found useful and convenient in that installation.

The /mnt directory is used to mount different devices such as cdroms to make them part of the directory tree.

While looking at these directories, you might have noticed that files under /usr/include often end in ".h" and that there are many files ending in ".so" in /lib. Although Linux places no restrictions on the characters possible in filenames, there are some conventions followed in some cases. Such files are often referred to as '.h' files, '.so' files and so on, or sometimes simply as h or so files. The Linux commands or utilities might enforce restrictions on the names of these files although Linux itself does not do so. Thus C program files end in ".c", C++ program files end in ".C", assembler source files end in ".s", and so on.

There is a useful command, file, to determine the type of a file. This takes any number of files as arguments and tries to determine the type of each. Although it is not foolproof and is open to deceit, it usually does a good job.

☞ Check Your Progress 11

1) Look up the Linux documentation for the various utilities above and find out which of them enforce file naming conventions.

....

....

....

2) Run the file command on various kinds of files from the various directories you have seen and see if their types are reported correctly.

....

....

....

2.4 SUMMARY

In this unit we have started at the beginning and looked at many basic Linux commands. However, there are many useful commands that we have not been able to examine. You will need to refer to the documentation and learn these. By now you would know enough about Linux to conduct a session with ease. In the units to follow we will examine some more commands and look at some utilities in slight detail.

2.5 SOLUTIONS/ ANSWERS

Check Your Progress 1

1) Yes, any number of persons can use the same user account on a linux system. A linux system does not try to identify a person physically and there is no constraint on more than one person using the same account. So long as they all know the password of the account they can always use it.

2) No. There can be only one account under one name.

3) Yes, there is no restriction in linux on this. Any number of accounts can have the same password.

Check Your Progress 2

1) You will not be able to login, unless your login name is in fact all in upper case, as linux is case sensitive. So the exact case of the login name matters and you will need to type it in exactly as it is.

2) You will not be able to get the prompt unless you are able to guess his password. Thus, your own password should be such that it is not easy to guess.

3) There is no difference in the system's response, which is "Login incorrect". This is so that an intruder (who might not know for sure whether a particular account exists) does not get any information about the existence of an account unless he is first able to login.

4) You will not be able to use the mouse in text mode. It is possible that you see a block cursor and that some rows of the screen get highlighted as you move the mouse about. But you will not be able to perform any useful operations.

Check Your Progress 3

1) You can use the backspace key to delete the character before the cursor position, or use Ctrl-U to erase all the characters that you have entered. You might not be able to use Ctrl-H instead of the backspace key. The same holds while trying to correct typing mistakes while entering the password, but you will not be able to see anything – neither the cursor nor the characters typed.

2) In bash this depends on the value of the environment variable HISTSIZE. It has a default value of 500.

3) You can do this by entering two consecutive '\' characters, such as in abc\\def, which will issue the command abc\def. The first \ escapes the second, which thereupon loses its special meaning.

Check Your Progress 4

1) Say

```
$ cal mm yyyy
```

where mm represents the two digits of the month and yyyy the four digits of the year.

2) Look at the calendar for September.

3) Use the `man` command. It is an important system level operation that affects all users and so no ordinary user can be allowed to change the date.

4) The command

```
$ who –b
```

tells you when the system was booted. To see how many users are logged in, say

```
$ who –q
```

5) There will be a different entry for each terminal window. But the username will of course be the same.

Check your Progress 5

1) As an ordinary user linux will complain that the password is not changed.

2) An ordinary user cannot help you if you have forgotten your password because s/he cannot look at or change your password unless s/he knows it already.

3) Try yourself

4) Try yourself

Check Your Progress 6

1) There are of course several ways. One is to give the same command repeatedly such as

```
$ ls;ls;ls;ls;ls
```

or use

```
$ ls –R /
```

that gives a complete listing of every (non-hidden) file on the machine assuming have the requisite permissions.

2) Using ^S manually to control screen output is not at all a convenient or even feasible way. The machine's response is too fast for you to control. So you will not be able to read a long file in sequence easily by this method.

Check Your Progress 7

1) Linux commands tend to have a large number of options. Though there are exceptions such as the `cal` command, most commands have so many options that you cannot possibly remember all of them. You can cope with this by remembering some of the most frequently used and useful options first. Later as you need to perform some other tasks, you can look up and remember more options.

Check Your Progress 8

1) Linux would complain that your home directory did not exist and would log you in to the root directory by default. You will likely have permission problems if your home directory does not exist and might not be able to do much by way of saving files.

2) There would be more than one way. You can enclose the entire filename in single or double quotes, or you could escape the spaces with a \ immediately preceding the space character.

Check Your Progress 9

1) Try Yourself.

2) Use ls -1.

3) While you could certainly store information in the filename itself, that would not offer the same convenience as storing the information in the file. This is because you could not edit the filename conveniently, or find patterns in it or use any file manipulation commands on it.

4) Try Yourself

5) Try Yourself

Check Your Progress 10

1) Try Yourself.

2) Linux complains that there is no such file or directory and you remain wherever you were.

3) Yes, there is no restriction on this or on another directory of the same name under a directory. It is possible because the absolute pathnames of the two will not be the same.

Check Your Progress 11

1) Try Yourself

2) Try Yourself

2.6 FURTHER READINGS

There are a host of resources available for further reading on the subject of Red Hat Linux version 9.0.

1) http://www.redhat.com/docs/manuals/linux

2) http://www.linux.org gives among other information, a list of good books on Red

   Hat Linux.

3) Consider joining a good linux mailing list, e.g.

UNIT 3 LINUX UTILITIES AND EDITOR

Structure **Page Nos.**

3.0 INTRODUCTION

In this unit, you will start to delve deeper into Linux and learn some more useful commands. You will also see how to combine commands together to perform useful tasks for which there might not be any single command available. You will learn to write simple programs in the bash shell that will let you write your own Linux commands. You will also see how to use the graphical user interface of Linux to perform many tasks without issuing any commands on the command line. Finally, you will look at the editor, vi, available in Linux for you to edit text files.

Because of the large amount of material to be covered, we will have to be brief. However, we shall try to illustrate important concepts with realistic examples. You will then need to practice whatever you learn on a Linux computer so that you understand the variations and nuances of the commands.

3.1 OBJECTIVES

After studying this unit, you should be able to:

- use some simple and important Linux commands;
- understand the concept of standard input, standard output and standard error;
- be able to use filters and pipes to connect commands together;
- write simple shell scripts to produce your own commands;
- use the graphical user interface to perform tasks without needing the command line, and
- use the programmer's editor iv,

3.2 SOME USEFUL COMMANDS

In the last unit, you have seen how to use some basic Linux commands like ls, cd and pwd. We will now look at some more useful commands that are commonly required. Linux has a large number of commands some of which are useful for system administrators, some for software developers and so on. Here we will consider only general-purpose commands that any category of user needs. Also remember that we will look at only a few most commonly used options of the commands. For a full description you should refer to the documentation for the command.

File Manipulation Commands

cat

You have so far learnt how to see directory listings that tell you the contents of a directory in Linux. Usually there will be several files and perhaps some other directories listed. But how do you see the contents of a file? This is done with the cat command.

```
[kumarr@linux kumarr] cat first_file
```

prints out the contents of `first_file` on the screen.

Remember that Linux is not concerned with what kind of file `first_file` is. If it is an executable file or a file produced by using some editor that does formatting, then the output will most likely not be intelligible to a human reader. In any case, the file will probably be printed out so fast that you will not be able to see anything but the last screenful. So it is actually useful only if you want to see what is there in a small text file. There are other forms of the command that you will study a bit later in this unit.

For now, try the following command

```
[kumarr@linux kumarr] cat first_file second_file
```

You will find that the contents of both the files are printed on the screen one after the other without any kind of gap in between. Actually `cat` stands for concatenate, that is, join the files together. You can give any number of arguments to the cat command and it will print them out on the screen in the sequence you have specified. You will see later how to use this facility to advantage.

tail

Sometimes you might only be interested in something at the end of a text file. You could use `cat` and see what lies at the end, but if the file is a big one, you could be waiting for quite a while before the gobbledygook on the screen stops and you can make sense of what you see. For such a situation, you can use the `tail` command. It shows what is in the last part of a file.

```
[kumarr@linux kumarr] tail first_file
```

This prints out the last 10 lines of first_file. What if you need to see something that is in the 14th line from the end? You can say

```
[kumarr@linux kumarr] tail -14 first_file
```

You can use any other number instead of 14 that you need. So to see just the last three lines, say

```
[kumarr@linux kumarr] tail -3 first_file
```

Like `cat`, you can give multiple file names as arguments; the last part of each is printed out with a header showing the name of the file that follows. Some other useful options are `+n` to start printing from the nth line of the file instead of the beginning, unlike cat. You can also use `-cn` to print the last few bytes instead of lines, with n being the number of bytes you want to see.

cmp

This command lets you compare two files and determine whether their contents are the same or different. If they are the same, the command says nothing. Otherwise it prints out the first byte for both files where there is a difference between the two. There is also the -l option to print out all differences.

```
[kumarr@linux kumarr] cmp first_file second_file
```

Remember that if one file is a part of the other, they are still considered to be different files.

diff

This command compares two files line by line and reports the differences between them. Unlike the mechanical comparison of the cmp command, `diff` makes a much more intelligent comparison. It indicates the differences between the files in three ways a, d and c. These stand for lines which have been added, deleted and changed between the two files. The symbol ”<”refers to the first file and “>” to the second file.

```
kumarr@linux kumarr] diff first_file second_file
```

There is also the -b option to diff that ignores white space, and the -B option that ignores blank lines.

wc

This command counts the number of characters, words and lines in a text file. You can give the command any number of arguments, whereupon it performs the count for each file and gives the total on the last line. It takes the -c option to report only characters, the -w option to report only words and the -l option to report only lines.

```
[kumarr@linux kumarr] wc first_file
```

You can also combine options together, such as `-cl` to report characters as well as lines.

sort

This is a useful command that lets you print out the contents of a file in sorted fashion. You can choose the delimiter that separates fields and the default is the space character. It produces output on the screen by default but you can place the output into a file using the `-f` option.

```
[kumarr@linux kumarr] sort unsorted_file
```

There are several options to sort on fields, to use numeric order, to choose the collating sequence and so on.

There are many other useful commands that manipulate text files. The `tr` command can be used to translate or change characters in a file. The `split` command breaks up large text files into a number of files with a fixed number of lines each. The cut command can be used to produce vertical sections of a text file. To get formatted output on the screen, rather than the plain dump of the file that `cat` gives, you can use the `pr` command.

☞ **Check Your Progress 1**

1) Try using the `cat` command on a directory. What do you see?

....

....

2) What happens if you try to `cat` a non-existent file?

....

....

3) The `cmp` command normally reports files as different if they differ in any way. Suppose you have two files that differ at the beginning but their last portions are the same. How can you ascertain using cmp that two files are the same after a certain offset?

....

....

4) Are you able to use `wc` on a binary file? Are the results meaningful?

....

....

3.3 PERMISSION MODES AND STANDARD FILES

We will now see what file permissions are and how to change the permission modes of files and directories. So far we have had occasion to look at various commands, many of which have to do with files of various types. The permissions of files can make a great deal of difference to the way the commands behave. What are these?

You have already seen in Unit 2 that the long form of the ls command, the one with the -l option, tells you about the permissions of the file.

```
     [kumarr@linux kumarr] ls -l
total 32
-rw-rw-r—    1 kumarr   users    4 Oct 15 01:18 abc
-rw-rw-r—    1 kumarr   users    4 Oct 16 17:31 abcd
-rw-rw-r—    1 kumarr   users    5 Oct 15 01:17 abc\d
-rw-rw-r—    1 kumarr   users    5 Oct 15 01:17 abc d e f
drwxr-xr-x    2 kumarr   users    4096 Aug 21 20:33 _files
drwxrwxr-x    3 kumarr   users    4096 Oct  9 18:01 ignou
-rw-rw-r—    1 kumarr   users    27 Oct 11 22:51 xx
-rw-rw-r—    1 kumarr   users    75 Nov  2 22:13 xx.c
```

The first column of the first field has a “d” in it if the file is a directory. So `ignou` and `_files` are both directories. For ordinary files, the first column of the first field is a hyphen. The next 9 columns specify the file permissions. The user community in Linux is divided into three categories – owner, group and others. The owner is the person who first creates the file. Several users at an installation can be made part of a user group. Such a facility is useful in keeping working on the same project or department categoriesed together. All group members form the second category of users. Finally, the rest of the community is lumped under others, which are users who are neither the owner nor part of the same group. In the example shown above, the owner of the files and directories is kumarr and he belongs to the group users. Other users might also belong to the same group, though it is quite possible for kumarr to be the only one who is part of the group.

Having studied this characterisation of Linux users, you can now begin to understand the permission modes. Every file has three possible modes of access – read, write and execute, represented in the directory listing by r, w and x respectively. A file to which you have read access can be read by you, which means you can look at its contents by using any method like the `cat` command. If you have write access to a file, you can alter its contents. Execute permission is relevant in the case of executable files like those that we have seen in`/bin`, or for shell scripts, that we will study later in this unit. You can run or execute a file only if you have execute permission on it. Execute permission does not mean anything in the case of text files, nor actually for any file that is not executable, except for directories where this permission has another meaning.

Now you know enough to understand the permission information. The nine columns are divided into three parts – owner, group and others – of three columns each. The permissions are specified in the order read, write and execute. If a permission is available, the corresponding letter is shown, while the absence of a permission is indicated by a hyphen. So `rwx` means the category concerned can read the file, can write to it or alter its contents and can execute the program which the file contains. Similarly the permission `r-x` means that the file can be read or executed but not altered or written to, because write permission is absent. `r—` means the file can only be read, and `–x` means it can only be executed. –– means that there are no permissions available and the file cannot be accessed at all. However, you can still see an ordinary file like this listed in a directory listing taken in the usual way. So you see that the permission columns in the listing shown above mean that for directories the owner has read, write and execute permission, whereas other members of his group have read and execute permission but no write permission. Similarly others, that is, users who are not part of the owner's group, also have read and execute but no write permission. Thus if you want to have all the permissions on a file, while denying group members write permission and allowing others only execute permission, the permission modes should be `rwxr-xr-x.`

With this knowledge, you should again look at the Linux system files and study the owners and permission bits for each. You will find that all users have executed permission on the files containing system commands. This is obviously necessary otherwise you could not use those commands. If possible, ask your system administrator to remove your execute permission on the ls command for a short while and then try to see your directory listing. If you are at a large installation where this kind of thing might not be possible, you can either wait until you know more about Linux to be able to see the effects of the absence of permissions, or ask somebody to make a copy of a system command in your directory, remove execute permissions on it and try to run it from your directory. Here you must take care not to run the system version of that command.

So far we have talked only about ordinary files and what file permissions mean in their case. But directories are also files, as you have seen earlier. Do the permissions all have the same meaning where directories are concerned? If so, what is it like to execute a directory? Let us delve a bit into this and find out some answers.

First of all you must understand that directories are special kinds of files, which contain information about other files, the permission bits have somewhat different meanings than what a hasty guess would suggest. Before we even look at what read permission for a directory means, try running the cat command on a directory and see what output you get. Linux will report an error and tell you that you were trying to cat a directory.

```
[kumarr@linux kumarr] cat ignou

cat: ignou: Is a directory
```

Actually a directory contains information on the files it contains, such as their names. Knowing this, it should be easy to deduce what read permission for a directory could mean. If you can read a directory you can see what files are in it, which means you can do an ls on the directory. In the absence of read permission you will not be able to look at the directory listing for that directory.

What about write permission? If you have write permission on a file you can change its contents. In the same way, having write permission on a directory allows you to change its contents. What does that mean? Creating, renaming or removing files would mean altering the contents of that directory. So it follows that having write permission on a directory enables you to create, rename or delete files in that directory.

Beginners often find it a bit difficult to grasp this point though it is not really hard to understand. You should remember that having write permission on an ordinary file allows you to change the contents of the file but does not allow you to delete or rename the file. That is possible only if you have write permission on the directory containing the file. Later we will see how this could lead to a security lapse in some situations.

Lastly, coming to execute permission you would agree that there is no way you could execute a directory corresponding to the normal sense that the operation refers to for an ordinary file. For a directory this permission bit determines whether you can cd to the directory or can copy files from that directory (this only if you have read permission on the directory as well). This permission is often called search permission. To be able to cd to any directory you must have search permission on every component of the absolute pathname of the directory. If search permission is absent on any component, all files and directories on that component and below it become inaccessible.

Apart from these permissions there are some other permission modes that you will come across. We will take these up in the unit on system administration. For now it will be sufficient to know that some other permission modes exist so that you do not get taken aback if you find characters other than r, w or x in the permission modes of a directory listing.

Changing Permission Modes

We will now see how to change the permission modes of files and directories. The action of a command can differ greatly depending on permissions. For example, the `cat` command will usually type a file on the terminal but will refuse to do so if the user does not have permission to read the file.

How do we change the permission modes of a file? The command to do so is chmod. It can be used only by the user who created the file or by the superuser. The owner is the user who created the file first, by any means such as by using a text editor or by copying an existing file.

Note that having all permissions on a file does not amount to owning it. If you can read somebody else’s file, it does not mean that you can prevent others from doing the same, but you can establish such protection for a file of your own. Also do not get confused between the original file and a copy you might have created, having only read access to the source file. You can change your copy in any fashion you wish, but you will not be able to alter the original.

There are two forms in which you can use the chmod command. Let us look at the absolute method first, as it is slightly easier to understand and use. In this the permission mode desired for the files is given to the command in an octal notation that we will explain shortly. The mode of the file then gets changed to what was asked irrespective of the permissions before the command was run. The form of the command is thus

```
[kumarr@linux kumarr] chmode mode file
```

where `filename` is a list of one or more files whose permission modes are to be set to `mode`. If the permission bits are `mode` to start with there is no effect on the file or files after running the chmod command.

You know that file permissions are specified by 9 columns, for example `rwxr-xr-x` or `rw-r—r—`. In the absolute method the presence of a permission is indicated by a 1 and its absence by a 0. The resulting 9 bit binary number is then converted to octal. This octal number is what has to be specified as the mode for the chmod command.

You know that a binary number can be easily converted to octal by making groups of 3 bits starting from the right. Now convert each group into octal as if it were a single number. The resulting string of octal digits is the number in octal. Thus

```
rwxr-xr-x
```

can be written in binary as `111101101` after replacing each permission by a 1 and each hyphen by a 0. This binary number can be written as `111 101 101` after grouping the bits in threes. The octal form of the number is thus 751. So to convert a file to this mode say

```
[kumarr@linux kumarr] chmod 755 progfile
```

This will give `progfile` the permissions `rwxr-xr-x`. Likewise `rw-r—r—` in octal is 644. So you can provide these permissions to your file `motd` by saying

```
[kumarr@linux kumarr] chmod 644 motd
```

Instead of one file you can set the permissions on several files at the same time (all to the same value) by listing the files after the mode. So

```
[kumarr@linux kumarr] chmod 600 motd passwd
```

will make their permissions `rw———-`. Thus you can read these files or change them whereas nobody else (except root) can even read them. So

```
[kumarr@linux kumarr] chmod 0 passwd
```

will mean nobody has any permissions on the file passwd and even you will not be able to read a file of your own with such a set of permissions. However, you can change the permissions any time since you own your file. Also, the super user can change the permissions of any file.

Let us now look at the symbolic method of telling `chmod` the mode. Here the permission types are, as always, `r, w` and `x`. In addition, there is a set of characters which specify the target of the actions. The target can be `u` (users), `g` (group), `o` (others) or a (all of these). The actions are + to add a permission, = to set it absolutely and – to remove a permission. So you can say (there must be no spaces in the mode argument)

```
[kumarr@linux kumarr] chmod a+x progfile
```

to allow everybody to run progfile, irrespective of the earlier execute permissions on it. However, this is different from saying

```
[kumarr@linux kumarr] chmod 111 progfile
```

because this would remove read or write permission for everybody, whereas in the earlier case, those permissions would have been left untouched. If the owner had read, write and execute permission, he would retain it. If the group earlier had read and execute permission it would continue to have that privilege. If others had no permission, they would acquire execute permission. One can remove read and write permissions for others by saying

```
[kumarr@linux kumarr] chmod o-rw progfile
```

One can specify absolute permissions by saying

```
[kumarr@linux kumarr] chmod u=rwx,g=rx,o=x progfile
```

Here the different target categories are given different permissions on `progfile` by separating them with commas. No spaces should be present in the argument, otherwise only the first part will be taken as the desired mode. The portion after the space will be treated as a filename, which has to be assigned those permissions.

You might find the numerical way easier to use. However, if you do not want to alter some permission bits, there is no straightforward alternative to using the symbolic mode. For example, if you want to deny others any permission on a file but do not want to alter your own or your group's permissions, you can say

```
[kumarr@linux kumarr] chmod o-rwx progfile
```

But to achieve the same result using the absolute method you would have to first determine the existing permissions. Suppose the value is 644. You will now have to say

```
[kumarr@linux kumarr] chmod 640 progfile
```

If the initial permissions were 666, you need to say

```
[kumarr@linux kumarr] chmod 660 progfile
```

instead. If using the symbolic method, you would not need to worry. The command you need to give remains the same in both cases since the o action leaves the u and g permissions intact.

☞ Check Your Progress 2

1) Try executing a directory on which you do not have search permission, and another on which you do have it. What happens?

....

....

1) Can you look at the listing of a directory if you do not have search permission on it? Why?

....

....

3) Can you remove files form a directory if you lack permission on them?

....

....

3.4 PIPES, FILTERS AND REDIRECTION

By now you have seen quite a few Linux commands, and you must have observed that many commands produce or can produce output on the terminal screen. Likewise many commands can take input from the keyboard. Actually, these commands have been written to accept input from a standard input file and to produce output in a standard output file. Usually these files are set to the keyboard and the terminal screen respectively. Let us look at this in somewhat more detail by studying some examples.

Standard Output

If you make a list of commands you have learnt so far you will find that many of them produce some output. For instance let us say

```
[kumarr@linux kumarr] cal
```

which prints the calendar for the current month and year on the screen. In practice there are very few commands designed to produce output on the screen specifically. The programs are written to produce output on what is called the standard output, and Linux sets the standard output to be the screen by default. That is how the output happens to appear on the terminal.

The shell, which interprets all your commands and passes them onto the Linux kernel for execution, has a facility to alter the standard output. In other words, you can define a file, rather than the screen, to be your standard output. (Actually the terminal screen is also a file as far as Linux is concerned.) To do this you need to say

```
[kumarr@linux kumarr] cal > calfile
```

There can be zero or more spaces before and after the sign. This sign indicates that the standard output of the command preceding it should go to the file specified to its right rather than to the terminal screen. This is called redirecting the standard output. In the current case the calendar for the current month will be placed in the file calfile. You can verify this by

```
[kumarr@linux kumarr] cat calfile
```

although you could have redirected this output as well.

```
[kumarr@linux kumarr] cat calfile > catfile
```

Is that not a way of copying calfile to catfile? Note that the file to which output gets redirected gets overwritten if it already exists. You can verify this easily by

```
[kumarr@linux kumarr] ls -l > calfile
```

and now examining the contents of `calfile`.

There is another operator, which appends output to the file specified rather than overwriting it. This is achieved by

```
[kumarr@linux kumarr] cal 06 1994 > > calfile
```

Now calfile will contain the calendars for the current month as well as for June 1994. Compare this with

```
[kumarr@linux kumarr] cal 06 1994 > calfile
```

which leaves only the calendar for June 1994 in `calfile`.

Thus the >> sign is safer to use because it never destroys any data, but this operation will keep adding to the file, and it can sometimes be difficult to make out what part of the output was produced by your last command and which portion is the outcome of previous redirections or was simply the original content of the file.

Standard Input

Just as many commands produce output on the screen, some commands take input from the keyboard although most take input from files. Look at an aspect of the cat command you have not studied so far.

```
[kumarr@linux kumarr] cat
```

The result of this command is deafening silence. The uninitiated might wait several minutes before aborting the command, thinking there is something wrong because the system does not appear to be doing anything at all. The truth is that `cat` can take its input both from the standard input as well as from a file. However, the output is always produced on the standard output. If any filenames are specified they are used as the input but if none is mentioned the input is taken from the standard input. There are also some commands that take input only from the standard input.

In the present case no filename has been specified and `cat` is waiting for input from the standard input, the keyboard here. So if you type something cat writes it out to the standard output and the effect is that of echoing your input.

```
A foolish consistency is the hobgoblin of little minds – Emerson

A foolish consistency is the hobgoblin of little minds – Emerson
```

If you want to put an end to your misery you can terminate your input file by saying ^d, thereby causing cat to finish and present you with your prompt.

To redirect standard input, say

```
[kumarr@linux kumarr] cat < catfilesrc
```

whereupon cat will print the contents on the screen. This is just the same as

```
[kumarr@linux kumarr] cat catfilesrc
```

because cat can take its input from a file as well. So to copy this file to `catfiletarget`, you can say

```
[kumarr@linux kumarr] cat < catfilesrc > catfiletarget
```

or

```
[kumarr@linux kumarr] cat catfilesrc > catfiletarget
```

Thus you can redirect both standard input and standard output in the same command. Some commands do not take input from the standard input. In such cases redirection of the input is not possible, as with the `ls` or `who` commands.

Remember that redirection is a facility provided by the shell, not by the command. The command being run does not know or care what its standard input and output are connected to, and it continues to use them. So the command has to be designed to take input from the standard input if redirection of input is to be possible. Thus you cannot say

```
[kumarr@linux kumarr] cp < cpsrcfile
```

because cp does not take its input from the standard input. Similarly, output redirection is not possible unless the command is designed to write to its standard output.

Standard Error

So far we have seen the effect of redirecting the output of some commands that completed successfully. Let us look at this a bit more closely. For example, if there is no command like gah, say

```
[kumarr@linux kumarr] gah > gahfile
```

If you do so you will find that you get a protest message from Linux on the terminal but that gahfile is empty. Similarly

```
[kumarr@linux kumarr] ls -l gah > lsfile
```

produces a message on the terminal but nothing in lsfile. Why does the redirection fail? After all the command did produce output.

The reason is that there is a third standard file in Linux, called the standard error. Linux utilities and programs are usually designed to provide error messages in case there is something wrong and the program is not able to proceed as expected. Such messages are often referred to as diagnostic output because they can help the user diagnose the reason for failure. This kind of output is usually written to the standard error file. Usually the standard error is also connected to the terminal by default, but like the standard input or output, the standard error can also be redirected. To do this in the bash shell, say

```
[kumarr@linux kumarr] gah 2> gahfile
```

This will place the standard error in gahfile. How does one place both the standard output and standard error in the same file? For this, say

```
[kumarr@linux kumarr] ls -l gahfile yy > lsfile 2>&1
```

To redirect the standard error and standard output to different files, say

```
[kumarr@linux kumarr] ls -l gahfile yy > lsfile
2>lserrfile
```

Filters

A filter is a command which can take its input from the standard input and can produce output on the standard output. Having the capability to read from or write to files is not a disqualification. So ls is not a filter because it does not read from the standard input but cat is one because it can do so (although it can read from a file as well) and also writes to the standard output.

You can think of a filter as a "device" placed between the standard input and the standard output which filters the standard input before placing it on the standard output. In the case of cat there is no filtering action at all, but a command like `grep` does perform some weeding action on its output.

The standard output of a command can serve as the standard input of another. Several commands can be chained together like this. Such an arrangement is called a pipeline. Pipelines are one of the big strengths of Linux, because they often enable us to group several existing commands quickly to perform a task for which there is no command directly available.

A major design goal of Linux was to have an operating system which allowed easy sharing of data and programs, and allowed people to build on the work of others instead of having to do things from scratch. The facility of pipelining helps meet this goal because you can piece together commands written by different people to achieve your objective rather than wasting your time on doing things which have already been done. Let us take a simple example.

Suppose you want to find out how many of the files in a directory are directories rather than ordinary files. It would have been wonderful if there had been an option to ls which did this job, but since that is not the case we will have to try something else. One-way is to look at the directory listing with ls -p and count lines, which end in /. Such a visual method is tedious and prone to error, especially if there are many files in the directory. So let us try to make Linux do this for us. How about the following?

```
[kumarr@linux kumarr] ls -1 -p > tmp

[kumarr@linux kumarr] grep -c '/$' tmp
```

We first get the listing in a temporary file tmp and then count the number of occurrences of / at the end of a line in tmp using the `grep` command. The result will be available on the standard output. While this method will work it has a few disadvantages. One is that it is slow because an intermediate file has to be created. Secondly we cannot start the `grep` command before the `ls` finishes. Also if we run many commands like this we will be left with temporary files, which we will have to meticulously delete lest they clutter up our directory listing and otherwise waste disk space. So we can use a pipeline like this

```
[kumarr@linux kumarr] ls -1 -p | grep -c '/$'
```

The '|' symbol is the pipe character. It means that the standard outpu of ls -p is passed to grep. The act of connecting the standard output of a command to the standard input of another is also referred to as piping the output of the first command to the second. Here no temporary files need to be created or cleaned up by the user as Linux itself takes care of the details. Also the speed improves because the subsequent commands can start as soon as some data is available to them.

A command like ls which does not take its input from the standard input can only be the first command in a pipeline. Similarly a command which does not write to the standard output can only be the last command in a pipeline. Also, it is the user's responsibility to see that each command receives input in a form which it can meaningfully transform, otherwise the results will be gibberish. Thus do not pass data files other than text files to grep because grep works only with text files, with lines delimited by the newline character.

☞ **Check Your Progress 3**

1) Which of the commands you have learnt so far are filters?

....

2) How will you count the number of all files in a directory?

....

3) Look up the fee command. How do you think it can be useful?

....

....

3.5 SHELL SCRIPTS

The shell in Linux is a wonderful entity that serves us in various ways. It is started up automatically every time you login to the system. The shell sets up your environment when you start off on the machine. It is the shell that lets you run different commands without having to type the full pathname to them, even when they do not exist in the current directory. The shell expands wildcard characters, thus saving you laborious typing. It gives you the ability to run previously run commands without having to run the full command again. It is the shell that does input, output and error redirection.

You can use the shell as a programming language. It has all the usual language constructs like sequencing, looping, decisions, variables, functions and parameters. Here we will take a very brief look at bash, the shell commonly used in Linux. A shell itself is a program and there can be many different shells available. Linux also has a shell called tcsh that has a C like syntax and is an enhanced version of the Unix C shell. Bash is the Bourne again shell and is compatible with the Unix Bourne shell. To discuss the capabilities and features of bash to some extent would require an entire block in itself. What we will try to do here is introduce some basic features and refer you to the documentation or to other books on the subject for more detail. To become comfortable with shell programming, you will need to practice a lot, just as you would need to do for any other programming language.

After writing some shell programs you will realise that some Linux commands like sed that earlier seemed to you to be of limited utility are actually very useful. Shell programs are often called shell scripts.

A Linux machine can be thought of as being composed of several layers. At the lowest layer is the hardware which does all the physical tasks and without which there would be no computer and no Linux. Above that is the Linux kernel, which is the core of the operating system and does memory management, device handling and all the other mundane tasks needed to make the hardware easily usable by us. The Linux commands and utilities come next. At the top is the shell which can be considered to be the outermost layer and which enables us to run the utilities and other Linux commands. You can also construct higher application layers of your own which run above the shell. However, despite the layering that we have talked of, the shell is itself a program like any other.

Wild Cards

Wild cards characters are characters which can stand for characters other than themselves, somewhat like a joker in a pack of cards (though, unlike wild card characters, a joker has no intrinsic meaning by itself). A judicious use of wild card characters can make many commands easy to issue by saving a lot of typing and preliminary research. Suppose you have been writing a series of programs for enciphering text. You have been calling them cph01.C, cph02.C, cph03.C and so on. You suddenly realize that you have been doing this in the directory ~khanz/crypt, while actually these programs are for a particular project and you would like them in the directory ~khanz/crypt/knapsack. All you have to do to rectify the situation is to move your programs to the correct directory after creating it. So you can start off

```
[kumarr@linux kumarr] cd ~khanz/crypt

[kumarr@linux kumarr] mkdir knapsack

[kumarr@linux kumarr] mv cph01.C knapsack

[kumarr@linux kumarr] mv cph02.C knapsack

[kumarr@linux kumarr] mv cph03.C knapsack
```

Soon you get sick of typing almost the same thing again and again, even if you just use the up arrow key and keep changing the required characters in the filename. Moreover, when do you stop? If you have used a naming convention whereby the filenames have numbers from 01 onwards, you could stop as soon as the mv command reports that the file does not exist. But this method is hardly a rigorous one. What if there are gaps in the sequence, and cph08.C does not exist but you have other programs going up to chp39.C? So you would have to keep looking at the directory listing first, and also from time to time in the process, just to be sure. With many such files, the method is tedious and prone to error.

There is much less effort if you use wildcards. After making the subdirectory, just say

```
[kumarr@linux kumarr] mv cph??.C knapsack
```

The ? is a wildcard character that can stand for any single character including itself. So `cph??.C` expands to `cph` followed by any two characters and then by .C. Also remember that ? stands for exactly one character. Therefore a filename like `cph1.C` will not be matched by the command given above. To match that filename, you would need to say `cph?.C`. So the command given will leave any files from `cph1.C` to `cph9.C` in the original directory.

What do we do if our naming convention for files starts the filenames with cph but allows any characters after that, with of course `.C` at the end? Does it mean we have to give several mv commands with one, two, three, and many more ? characters? That would be quite tedious again. Also, after how many ? characters could we stop, knowing that there is no specific limit on the number of characters in a filename? The answer to that is another wildcard character, the *. It can expand to any number of any other characters. So all you have to do is to issue the command

```
[kumarr@linux kumarr] mv cph*.C knapsack
```

However, the * does not expand filenames starting with a leading . character. Also, in bash, a filename like `yy*.c` is not expanded unless there is at least one filename around to which it expands. So if you do not have a filename starting with `yy` and you issue a command like

```
[kumarr@linux kumarr] vi yy*.c
```

what the vi editor will create is a file called `yy*.c`. If a file like `yy1.c` already exists, then you can create `yy*.c` by escaping the * with a backslash, so that it is taken literally.

```
[kumarr@linux kumarr] vi yy\*.c
```

You can create a file called `yy?.c` in the same way, though it might be a good idea to avoid such characters in filenames to prevent confusion.

Simple Shell Programs

Let us now look at a simple shell program. Suppose you are kumarr and are working on your cryptography project in a directory `~/prj/crypt/pkc/src`, where your source files are located. But sometimes you are looking at documentation in another location, or are also working on some other project. To look at the files in this directory you have to issue a longish command, which can be cumbersome if you have to do it often. What you can do instead is to create your own shell program, say in a file called `wd`. This you can do using any editor. The file `wd` should contain

```
ls ~/prj/crypt/pkc/src
```

Now you need to make `wd` executable by setting its permissions to 755. Otherwise you would need to invoke it by saying

```
[kumarr@linux kumarr] bash wd
```

But if you just said `wd`, it would not work because when bash is given some command to execute, it looks for the commands in different directories in a fixed sequence. Typically this sequence is /usr/local/bin, followed by /bin and /usr/bin, but this can be controlled in a way we will describe shortly. Since `wd` is not in any of those directories, Linux complains. So you can say `~/wd` to take care of this problem.

What we have now done is created a command of our own that anybody can use. Currenly it allows others to look at our directory, something we might not really care to do, but it nevertheless illustrates the point. And we have created the command by taking an already available command and using it in a different way. This is at the core of the Linux philosophy of building on the work of others.

Many installations have locally useful commands available in /usr/local/bin. You could also be contributing to this repository of useful commands by and by, and you should first explore to see whether the task you want to do is not already accomplished by using these commands. However, before you can release shell scripts for general use, you will need to make them robust, which is something we have not looked at in the current case. If you have some commands that you feel you need but are not fit for general release, you can place them in your own bin directory like `~/bin`.

Suppose the system default form of the `ls` command at your installation is not what you find useful; you would like it to be `ls -l`. One way is to create a new command like ls that contains the line `/bin/ls -l`. Note that if you put only `ls -l` in your private command, it might keep calling itself indefinitely and so will not work. Also your program will not be able to use arguments because we have not given that ability.

Variables

Variables can be defined and used in bash like in any other programming language. For example, to set the value of a variable vehicle to "bus", you can say

```
[kumarr@linux kumarr] export vehicle=bus
```

To see the value of a variable you can say

```
[kumarr@linux kumarr] echo $vehic

bus
```

If the variable has not been defined, nothing is printed. You can also print several variables together or print literals using the echo command.

```
[kumarr@linux kumarr] echo $vehicle and car

bus and car
```

To set the value of a variable to the output of a command, give the command in backquotes.

```
[kumarr@linux kumarr] export mydir=`pwd`

[kumarr@linux kumarr] echo $mydir

/home/kumarr
```

To let your command take arguments, you can refer to them as $1, $2, $3 and so on up to $9. $0 stands for the command itself and $10 would be interpreted as $1 followed by 0. If you want to use all arguments then say $*. Similarly, $# stands for the number of arguments. So if you write a command ech that echoes only the first and fifth arguments, it would have

```
[kumarr@linux kumarr] cat ech

/bin/echo $1 $5
```

Now you would find the other arguments would be ignored. The shell has several inbuilt variables of its own that you can look at by using the env command, to show the environment. You would be able to recognize several of them such as HISTSIZE for the number of commands that are kept in the history buffer, LOGNAME for your login name, SHELL for the shell you are using, HOME for your home directory and so on.

Programming Constructs

The shell scripts we have seen so far have been nothing but a sequence of commands that we could have anyway issued at the prompt itself. Shell programming would not have been of much use in that case. What makes it powerful are the programming constructs available such as loops and decisions. Let us first look at a program mkupper that converts the contents of its arguments to upper case.

```
[kumarr@linux kumarr] cat mkupper
for i in $1 $2 $3
do
    tr '[a-z]' '[A-Z]' < $i > $i.up
done
```

This converts the contents of up to three files to upper case and places them in a file of the same name with a `.up` suffix.

Besides the for loop, you can use the while or until loops with their usual meanings. For example, the script above could be written

```
[kumarr@linux kumarr] cat mkupper
while test $# -gt 0
do
tr '[a-z]' '[A-Z]' < $1 > $1.up
shift
done
```

Here suppose there are 25 arguments to the mkupper command. The condition in the while loop tests whether there is an argument available. If so, the contents of the file are converted to upper case. After each argument has been dealt with, it is discarded and the next argument becomes the first argument. This is done by the `shift` command in the `mkupper` file. Finally when there are no arguments left, the test fails and the loop terminates. You can achieve a similar effect using the `until` loop, given below

```
if test $# -eq 0
    then echo 'No files to translate'
    exit
else
    until test $# -eq 0
    do
         tr '[a-z]' '[A-Z]' < $1 > $1.up
         shift
    done
fi
```

The test operation can be used to check various conditions, such as if a variable equals a number. You can also use the other relational operators with the keywords `le, gt, lt, ge` and `ne`. Other tests that can be performed are `-w or -r` for checking if the filename is writeable or readable, `-d` to check if it is a directory and `-`

f to see if it is an ordinary file. In the script above we print an error message and exit if there are no arguments to the file. We could also have chosen to exit silently in such a case, as we did with the `while` loop example. Or we could wait for input from the standard input. The example also shows how to make a decision using an `if` statement. The statement is terminated with the `fi` keyword. The `else` part is optional. We can also use the `exit` keyword to break out of the shell script. Several else parts can be present in an if statement, they are then introduced with `elif`. Let us take an example script called countargs that counts the number of arguments and prints the result.

```
if test $# -eq 0

    then echo “No arguments”

elif test $# -eq 1

    then echo “Only one argument”

elif test $# -eq 2

    then echo “Two arguments”

else

    echo “Many arguments”

fi
```

Within a loop you can use the `break` and `continue` statements to exit the loop or to go back to the beginning of the loop respectively. Comments are introduced by the # character. Anything after this on a line is treated as a comment. In a shell script, like in any program, it is important and recommended practice to provide comments that explain the working of the program.

You can also use the case statement in place of multiple `if...elif` statements when it is the same condition that has to be checked each time. For example, let us write a shell script that checks the first argument to decide the operation to be performed and then performs the operation on the second argument. The case is terminated with an `esac` statement.

```
if test $# -ne 2

then echo “Usage: $0 operation files”

exit

fi

case $1 in

upper) tr ‘[a-z]’ ‘[A-Z]’ < $2 > $2.up;;

lower) tr ‘[A-Z]’ ‘[a-z]’ < $2 > $2.lw;;

*) echo “Invalid operation specified”;;

esac
```

You must bear in mind that the options in the case statement have to be specified in the right order. If we give the *) option first, then it would match every case and the script would do nothing.

You can also pass input from the user to a shell script. For this use the `read` command. Take the following simple script that prints out the directory listing as desired by the user. It has a friendlier interface than the Linux command ls.

```
echo "1 for long listing"
echo "2 for stream list"
echo "3 for single column list"
read x
case $x in
1) ls -l $*;;
2) ls -m $*;;
3) ls -1 $*;;
*) echo "Invalid choice"
esac
```

In the read statement you can assign values to several variables. The first value goes to the first variable, the second to the second variable and so on. If there are more values provided by the user than there are variables, the extra values go to the last variable. If there are few values, the remaining variables do not get any values. Values are delimited by spaces and a newline character causes the assignment to happen. So if you say

```
[kumarr@linux kumarr] read x y z
a b c d
[kumarr@linux kumarr] echo $x
a
[kumarr@linux kumarr] echo $y
b
[kumarr@linux kumarr] echo $z
c d
```

You can do simple arithmetic in the shell with the expr command. The operators and operands have to be delimited by spaces. When you use the * for multiplication, you have to escape it with a \, lest the * be expanded to all available filenames.

```
[kumarr@linux kumarr] expr 2 + 3
5
[kumarr@linux kumarr] expr 18 / 3
6
[kumarr@linux kumarr] expr 4 \* 5
20
[kumarr@linux kumarr] expr 7 – 5
2
```

To help debug shell scripts you can use the `–v` or `–x` options that give verbose output. The -x option precedes each command with a + sign. Thus

```
[kumarr@linux kumarr] bash goodls –v
```

will print each command as it is executed.

☞ Check Your Progress 4

1) Try the command?
   echo*
   Write files does it neglect to furnish you with? How can you get all filenames?

   ....

2) Write a shell script that prints out the contents of some fixed file in upper case.

   ....

3) Write a shell script that prints out a list of every unique word contained in the file in alphabetical order?

   ....

3.6 GRAPHICAL USER INTERFACE

Unlike the early versions of Unix, Linux has a good graphical user interface (GUI). This makes it different from the cryptic command line interface that proved daunting to most lay users who ventured to try Unix. Most of the common operations can be performed graphically and it is not necessary to use the command line. However, the power that the command line gives is still available to power users.

When you login you are presented with a text based screen, but thereafter you can have the system set up so that you reach the graphical user interface mode. This can be done by issuing the command

```
[kumarr@linux kumarr] startx
```

whereupon the X-window system is started up and you reach GUI mode. You have a default of 4 desktops that are available to you. Each of these can be arranged differently as you wish. To start up a terminal session, just right click the mouse on the desktop and select the “New Terminal” option. You will get a terminal window where you can use the mouse as well for editing, such as copy and paste. These options are available by right clicking the mouse or by choosing the Edit option on the menu bar in the terminal window.

The bottom tray contains some useful icons that will depend on what software packages have been installed. The leftmost icon is the red hat, the logo of the version of Linux that we are studying here. It has a small arrow to its right, clicking which brings up a list of options such as Accessories, Games, Graphics, Internet, Office and so on. Many of these options have further suboptions that you can select with your mouse.

The tray typically would contain icons for invoking the browser, email, the Open Office Writer, Presentation software and the spreadsheet, a printer manager and the icons for the desktops. To invoke any application, simply click its icon once in the bottom tray. This saves you having to know the name of the program for each application. When you move your mouse over any icon, a small explanation of the icon appears in a yellow box. It tells you the name of the program and what it does. This mouseover makes it easy for you to identify the correct icon for the program you want to run so that you do not have to rely on your memory to locate it.

If you right click on any of these icons, you get a menu where you have options to look at its properties or obtain help on the application. You can also remove the icon from the tray, but then if you want to run the application you will have to locate it in the directory structure. You can also move the icon around in the tray.

At the very left you have an icon in the shape of a red hat with an arrow pointing upwards. Clicking on this brings you to the main menu that has several options and suboptions. For instance, you can go to your home folder using one of the options. You can also copy and move files using mouse commands, or simply drag and drop files from one folder to another if you want to move them.

While in the folder window, you can right click the mouse and create a new folder. You can also rename folders or files or delete them.

3.7 EDITOR

Linux is rich in text manipulation and document preparation facilities. Here we take a brief look at `vi`, a line editor that is useful for creating and modifying text files. For example, if you want to be writing shell scripts or other programs, you will need an editor so that the program file can be created.

Text editors are different from the commands you have studied so far because you will be able to change existing files directly by using them. You need not be writing out the file to be changed to another file. Moreover, editors are interactive in that you need to be telling them what to do, command by command. This is unlike the commands you have seen so far, where you give the command once and it then does its job and terminates. Editors can be line editors or screen editors. Line editors work on a line at a time. Two line editors available in Linux are `ed` and `ex`. In contrast, screen editors present you with a screen of text and you can move around there, making changes as you want. So screen editors are more powerful and easier to use. A screen editor available in Linux is `vi`. Again because of lack of space we will only be able to look at a few basic features of `vi`. It is a programmer's editor that is not too easy to learn, though.

Unlike some other editors, `vi` does not automatically create a backup copy of the file being edited. Although editing does occur on a copy of the file, this copy is in the tmp directory that gets deleted when you save the file. The actual editing is done in a buffer in memory and the changes are written to the file only when you tell it to do so. You can therefore easily abandon a session that has gone badly wrong. But the same feature can be a problem in case of a system crash, though `vi` will try to recover as much as possible of the file that was then being edited.

It is difficult to describe an interactive command on paper and so `vi` is best understood by trying out the commands on a terminal. To start up `vi` and edit a file called linuxdoc, you say

```
[kumarr@linux kumarr] vi linuxdoc
```

At this the screen gets cleared, the file gets read into the edit buffer, the first portion of the window to the buffer appears on the screen and the cursor is at the first character of the line that you were editing when you last saved the file. This presupposes that `vi` knows how to deal with your terminal type, a task that would have been ensured by your system administrator when she set up your machine. You can resize your window at any time you wish without affecting your file. The bottom line of the screen shows the name of the file and its size in characters and lines. You can also see the current cursor position at the right of this status line. At times when you give commands to `vi`, the status bar disappears and instead you see the command that you are issuing. We will shortly look at some of these commands.

The text that you enter in `vi` is organised in lines. The editor is not a word processor and works only in non-document mode. So you need to explicitly tell `vi` when a line has ended. Otherwise it will continue to add text to the line until it reaches its limit for the length of a line. How do such long lines look on the screen? They are wrapped onto the next physical line on the screen if the width of the screen has been reached. If you increase the width of the window you can see the line getting redrawn. Similarly if you reduce the width of the window you will be able to see the line being rearranged on the screen. While by mere visual inspection you cannot make out whether multiple lines are actually the same line or are separate, you can easily find out, say by resizing the screen or going to the end of the line by the $ command.

You can start up `vi` with more than one filename, and you can even give wildcards.

```
[kumarr@linux kumarr] vi linuxdoc xx xx.c
```
or
```
[kumarr@linux kumarr] vi *.c
```

In such a case the files are presented to you one by one. After you are done with the first file, you are presented with the next one unless you choose to exit the whole operation, in which case the rest of the files are not presented to you. In this respect vi differes from other word processors where you can only bring up one file at a time. In fact you can start up `vi` without any filename at all. If you do that you will be presented with a blank screen and can then start entering commands. You can give the file a name when you save it.

Another option with which you can start up `vi` is the `–x` option that allows you to encrypt the file. You have to supply a key (you need to retype it to confirm) and when you save the file it is encrypted with that key. To now read the file you must supply the same key, without which it will not be intelligible. While not a foolproof method, it will certainly safeguard you against the casual busybody. One more option to vi is to open it in readonly mode with `–R`. You can navigate and examine the file as you wish but cannot make accidental changes as you have to use the force options to commands that alter the file. This mode can also be called up by using the command `view` instead of `vi`.

The -r mode of vi tries its best to recover from system crashes. If you want to start editing from a line other than the first, you can use the `+n` option where `n` is the line number you want to be at. To go to the last line of the file, just omit the number and simply say +.

From the number of different ways in which you can start it up, you would have had some idea of the kinds of commands available and the bewildering array of options that must be supported by `vi`. Now that you know how to enter `vi`, let us also learn how to come out of it. You can save the file with :w and force a save, say when you are in readonly mode, `by :w!.` Similarly :`q` will quit the file without saving changes and :`q`! Will quit without trying to save any changes. To save and quit you can also say :`x` or simply `ZZ`.

Navigating Around the File

When you open up a file in `vi` you will find the cursor at the line you asked for during the invocation. Unlike a word processor, `vi` starts up in command mode. It does not start inserting the text you enter. So any key you press is taken to be a command. `Vi` has many commands, and just about any key is likely to be taken as one. Once a command is given, subsequent keys pressed will pertain to the command until you exit the command. If you press a key that is not a valid command in command mode, nothing will happen and vi will emit a beep.

For example, to move around the file, you can use the arrow keys. If your terminal type is not properly set and you have some problem with them, you can use `j, k,` l and `h` for down, up, right and left respectively. The commands l or h will work only within the line. At the last or first character respectively of the line, they have no effect. Similarly pressing k at the first line or `j` at the last line of the file has no effect. To scroll forward half a screen, use `^D` and to scroll half a screen back, use

^U. Similarly, ^F and ^B will take you forward and backward one whole screenful respectively. If you are at the bottom of the screen and you press j or the down arrow key, the display scrolls up by one line and your cursor continues to remain at the last line. To scroll forward one line without changing the line at which the cursor is, use ^Y. Likewise, ^E will scroll backward one line without moving the cursor. The commands 0 and $ take you to the beginning and end of the current line respectively.

Operations on the window can blank out the status line, which you can always get back by ^G. To move forward one word, say w and use b to move backward one word. But unlike the character at a time commands, this works througout the file, which means that pressing b at the first word of a line will take you to the last word of the previous line. The command will not have any effect if you are at the first word of the file. Similarly w will have no effect at the end of the file. Note that b will bring you to the beginning of the current word if you are not already there and to the beginning of the previous word otherwise. Similarly e brings you to the end of the next word if you are at the end of the current one, and to the end of the current word itself otherwise. These lower case commands define a word in a way that is similar to the definition of an identifier in programming languages like C. This is very convenient while programming, as vi is really a programmer's editor. If you want to consider a word to be a sequence of characters delimited by spaces, you can use the upper case equivalents W, B and E.

These commands all take counts. This means 20W will take you 20 words forward and 90b will take you 90 words backward, with the appropriate definition of word. The % command takes you to the matching opening or closing (, { or [character. To move forward a sentence, use (and) to move a sentence back.

Similarly [and] move forward and backward one paragraph. These too can be preceded by a number that specifies the count.

Adding, Deleting and Changing Text

By now you have a good idea of the kind of commands vi takes. With your feet wet, it will not be hard to understand the commands for actually changing text. First let us see how to insert text. You can do this by using the i command. This puts you into insert mode. In this mode, whatever you type becomes part of the file. The text is added starting from before the current position of the cursor. To come out of insert mode back into command mode, you can say ESC or ^[. You can also use I to insert text at the beginning of the current line. Similarly, the a command adds text after the current position of the cursor, and A adds text at the end of the current line. To open up a new line under the current line, type o and use O if you want to open up a line before the current line. To add a control character to the file, type ^V followed by the character. The s command deletes the current character and puts into insert mode, while S deletes all text on the current line and puts you into insert mode.

Deleting text is similarly straightforward. The x command deletes the character at the current cursor position but does not move the cursor. At the end of the line, the command brings the previous character under the cursor. So pressing x repeatedly anywhere on a line will finally leave you with a blank line. The X command deletes characters to the left of the current cursor position and on reaching the beginning of the line no more characters are deleted.

To delete more than one character use the d command. This has to be followed by another letter to indicate what is to be deleted. So dw deletes a word, db deletes a word in the backword direction and dW and dB delete a word in the forward and backward directions according to the respective definitions. Word deletions happen across line boundaries. When deleting words on the next or previous line, the two lines are joined together. To delete the rest of a sentence from the current position, use d(, and d) will do the same up to the beginning of the sentence. The same can be done for paragraphs as well with the d{ or d} commands.

All the above deletion commands can be preceded by a number to indicate how many times they should be performed. To delete a line completely, say `dd`, or use `D` to delete the rest of the line from the current cursor position. Similarly d^ will delete a line from the beginning till the current cursor position. These commands will not remove the newline character. To delete a block of lines, give the range preceded by a colon, as in `:4,10d` to delete from the 4th to the 10th line (both inclusive). Here you can use a . to indicate the current line and $ to indicate the end of the file. So :.,`$d` will delete the rest of the file starting from the current line. You can also use the + sign to indicate the number of lines to delete, as in `:15,+8d` to delete 8 lines starting from the 15th line.

Internally `vi` maintains the line number of each line in the edit buffer. You can display line numbers by saying :`set` number or :`se nu`. All commands that begin with a colon have to be followed by the ENTER key before they will take effect.

Changing text is done in much the same way. The r command will replace the current character with whatever you type next, while R places you in replace mode. Then each character you type will replace the next character until you press ESC. To replace a fixed number of characters by the same character, precede r with the count, such as 23r. The ~ command changes the case of each character that is a letter. It too can be preceded by a count.

For changing blocks of text, use the c command followed by what you want to change, such as cw for a word, `cw` to use the other definition of a word, c$ to change text upto the end of the line and `c^` to change text upto the beginning of the line. To change the complete line on which the cursor is located, use cc. You can precede cw, cW, cb, cB or cc by a count to indicate how many words or lines are to be changed. All c commands place you in insert mode that you need to exit in the usual way with ESC or ^[.

The . command can be used to repeat the last action. To join the next line to the current one by deleting the newline character between them use the J command.

Searching for, Copying and Moving Text

Let us now see how to search for text. The f command searches for the next character you give on the same line. This can be preceded by a count, so saying 3fx will look for the 3rd x from the current cursor position. Similary F looks for the character in the backward direction. A ; continues the search for the same character in the same direction, while a , continues to search for the same character in the reverse direction. Both these commands work only on the current line, but the ; or , commands can be used on a different line after repositioning the cursor.

To search for any text, precede it by a / character. It place the cursor at the beginning of the first occurrence of the text pattern. To go to the next occurrence, say n and say N to find the same pattern in the reverse direction. If the pattern, say hello, is not found, the message

```
        Pattern not found: hello
```

is displayed on the status line at the bottom of the window. You can start the search in the backward direction from the current curson position by preceding the text with ?. Both the / and ? will wrap around the file end or beginning. So after reaching the end, the search will continue from the beginning of the file. You can use the d command to delete text from the current cursor position to an occurrence of a search string, say `hello`, by saying d/`hello`.

You can bookmark a position in the file with m followed by any letter. So mx will bookmark the current cursor position. Now you can reach that position from anywhere in the file by saying ‘x. To reach the beginning of the line containing the bookmark x, you can say ‘x. Bookmarks are valid only for that edit session.

You can undo the effect of most commands by the u command, but repeating it merely brings back the previous situation. So the command works like a toggle. However, you can use U to undo all changes on a line if you have not moved away from it. Deleted text goes into an unnamed buffer, from where it can be retrieved by using the p command and placed after the current cursor position. The P command places it before the current cursor position. You can use these commands repeatedly to place the contents of the buffer at different points in the file. Thus to take care of a transposition error involving two characters, say `xp` at the first character.

The y command yanks a block of text into the unnamed buffer without deleting it. So you can use yw or yy (also Y) to yank a word or a line. These can be preceded by a count. You can then place the text wherever you want by first going to that point and then using p or P.

Finally, you can use any of 26 named buffers by preceding the command with “ followed by the buffer name, which can be any lower case letter of the alphabet. Thus “m4yy will yank 4 lines and place them in the named buffer m. To put back the text at some point in the file, just go to that point and say “mp. You can append text to a named buffer by using the same buffer name but in upper case.

This completes our very quick overview of the vi editor. Linux also has available an office suite that includes a full fledged word processor suitable for editing text documents, while vi is useful for editing programs.

Linux has a full screen windows based editor, `gedit`, that creates text files. But it does not offer the wealth of commands and features that vi does, in spite of the fact that it is windows based. For a programmer, it might yet be better to use `vi`.

3.8 SUMMARY

In this chapter we have covered a lot of ground and the treatment of the topics has had to be brief. We have looked at some text manipulation commands that are frequently useful. We studied the meaning of permission modes for files that you see in the directory long listing, and also saw how to change them for files that we own. Next we saw how we can construct our own commands easily by joining together filters with pipes. This also brought us to the concept of the standard input, standard output and standard error. We then looked at how to write simple shell scripts in the bash shell. The graphical facilities available in Linux were then touched upon followed by a brief discussion of the programmer’s editor vi.

3.9 SOLUTIONS/ ANSWERS

Check Your Progress 1

1) Linux does not type out the file. On trying to type out a directory called “linux” that does not exist, we get an error message

```
cat: linux: Is a directory
```

2) On trying to cat a file that does not exist, we get an error message

```
cat: highest: No such file or directory
```

3) If there are two files file1 and file2 that are the same after byte offsets offset1 and offset 2, you can verify this using cmp using

```
$ cmp file1 file2 offset1 offset2
```

4) You can use wc on a binary file but the results are not meaningful for such files.

Check Your Progress 2

1) You will not be able to do so and will get an error message in both cases.

2) No. This is because of the definition of search permission. But you might be able to find out the files in it if you give it as the argument to the ls command.

3) Yes, you only need write permission on the directory. It does not matter

   whether you can write to the files or not or even whether you own them or not.

Check Your Progress 3

1) Try yourself

2) You can say
   ls -1 | wc -l

3) It is useful when you not only want to pipe the output of a command to another command, but also want to see or save that output.

Check Your Progress 4

1) It gives all the filenames except those that are hidden, that is, those that start with a period (.). To get these you can say
   echo .*

2) Try yourself

3) Try yourself

3.10 FURTHER READINGS

There are a host of resources available for further reading on the subject of Red Hat Linux version 9.0.

1) http://www.redhat.com/docs/manuals/linux

2) http://www.linux.org gives among other information, a list of good books on Red Hat Linux.

3) Consider joining a good Linux mailing list.

UNIT 4 USER TO USER COMMUNICATION

Structure **Page Nos.**

4.0 INTRODUCTION

In this unit, you will learn how to communicate with other users in Linux. Such communication can be offline or online. Online communication can be done using the write command. While instant messaging applications are commonplace these days, they are not specific to Linux and are available for most popular operating systems. We will take a quick tour of an instant messaging application from one of the popular providers to understand some of the facilities that are available. Electronic mail is one of the offline applications available in Linux and we will look at the Ximian Evolution e-mail client that also allows you to schedule your engagements. The Apache webserver is a very popular one, and you will learn the elements of configuring it to enable your computer to serve out web pages. You will also see how to configure some other network services such as the domain name service and a network file system server. Again, because of the lack of space, we will be able to look at only the basic concepts and features and will not be able to delve into the details of these matters. There are also some other services that we will not be able to touch upon in this short chapter.

4.1 OBJECTIVES

After going through this unit you should be able to do the following tasks. The configurations that you will be able to perform will be at a basic level and you will not become an expert.

- communicate with other users on the same machine with the write command;
- use an instant messaging application to chat with another user on the network;
- use the Ximian Evolution e-mail client to send and receive mail;
- perform simple configuration of the Apache webserver on your machine;
- configure your machine as a domain name server, and
- configure your machine to be a network file system server.

4.2 ON-LINE COMMUNICATION

In Linux there are two ways of communicating with other users online. The first is available if the other user is logged in to the same machine. Whenever this is so the user will be a terminal on the machine to which you can send messages using the `write` command, even if that user has connected to your machine from another machine over the network. While the facilities are rudimentary, `write` is quick and simple to use. If you require more sophisticated features, you can use an instant messaging application from any of the popular providers, or use the `gaim` instant messenger that comes with Linux.

The write command

This command allows you to send a message to another user logged onto the same machine at the same time. It is thus a means of online communication because the recipient obtains the message immediately. There are other means of communication that we will look at that are offline in the sense that you send the message and the other user might not pay any attention to them if he so desires. Of course, even with the write command, the other user might not pay attention to the message that you send. The same is true for an instant messaging application as well. So the distinction between what we are calling online and offline methods is just that in the former, the other user or users are expected to be available at that time – they may or may not choose to respond. If they do decide to engage in the communication, they are available at the other end and you can then conduct the conversation.

You should be careful while using `write` as it can be very annoying to receive a message which clutters up your screen while you are concentrating hard trying to do some piece of work. Sometimes you have some precious output on the screen which you have obtained after some effort and a write message appears and causes that output to scroll off. If you are working in a terminal window you can scroll back to see your output, but it can still be annoying to be disturbed. Or you are working in an editor and the message garbles your screen. There are many such situations in which you would rather not receive any message on your screen. Again, some people are more touchy than others on this issue. So you should be careful and use write only when you have something urgent to say, or when you are sure that the other party will not mind. In fact, one of the most irritating times to receive a `write` message is when you are already replying to somebody else's `write` message. You might even want to arrange with some friends for such a situation to happen and see how it feels.

One thing you could do is to find out using the `ps` command what the intended recipient is doing. If your party is in the shell, it might not be so inconvenient for him. On the other hand, if he is in `vi` or some application package, then it might be best not to bother the person. Then again, the urgency of your situation needs to be taken into account. Another possibility is to send the user a brief `write` message asking whether it is all right to send more. If the recipient does not want to be bothered at that time, she can tell you so. All said and done, you should be aware of the fact that `write` can be a rude command to use.

After this long sermon, we can come to the command itself, which should be a breeze for you after all of the two units of Linux experience that you have had. Let us find out some of the users on the system currently.

```
[kumarr@linux kumarr]$ who
ramk    tty1 Dec 10 20:47
kumarr  pts/0    Dec 10 22:59 (:0.0)
khanz   tty2 Dec 10 21:29
pramod  tty3 Dec 10 22:11
```

Suppose you, `kumarr`, want to write to `khanz` and ask him about some program he had promised to give you. So you say:

```
[kumarr@linux kumarr]$ write khanz

Can I have that matrix inversion program please?

I need it badly to test out my work.

^D

[kumarr@linux kumarr]$
```

We get back the prompt after sending the message. Notice a few things about `write`. One is that after issuing the command there is no indication at all that you should proceed with your message, because there is no prompt on the screen nor any automatic acknowledgement which comes from the other end. The command expects input from the standard input and faithfully transmits it to the destination.

Therefore after issuing the command you can type in your message, which can be as long as you like. When you are finished, you should type the end of file character to signal the end of input. This character is usually ^D. On doing this you will get back the prompt. There is no indication whether the other party has received the message or not. Also the message should not be longer than a screenful or the other person might have difficulty reading it, even though you can scroll back in a graphical terminal window in Linux.

Now what about the other end of the channel? Let us zip across to khanz's terminal and see what he has got, omitting the secret work he was busy with when you butted in.

```
Message from kumarr@linux on pts/0 at 23:08 ...

I need it badly to test out my work.

EOF
```

So he did got your message, but what about a reply? Well, you did not see anything on your screen because he has not had a chance to answer yet. Let us see his reply now;

```
[kumarr@linux kumarr]$ write kumarr

I was downloading it but the line got cut midway.

Will give it when I have it.

Have patience or get it yourself.

^D
```

and that is exactly what you receive on your terminal except for the letters EOF in place of the ^D.

So we now have seen a straightforward case of exchanging urgent messages with `write`. Often such messages would have gone by mail. The `write` command is more commonly used for carrying on an online conversation with others who are also logged in at the time, rather than just sending a message. Let us see how to do this.

Using the information from the `who` command earlier, let us say we try to converse with pramod.

```
[kumarr@linux kumarr]$ write pramod
```

Instead of immediately writing out your message, you can wait to see if pramod is in a mood to respond. After a minute or so you have not had a reply and so you just come out with ^D. He is too busy to talk to you, or maybe he just does not want to be disturbed and has arranged matters that way. This is getting to be irritating now, so let us try somebody else.

```
[kumarr@linux kumarr]$ write ramk
```

```
Message from ramk@linux on tty1 at 11:44 ...
```

At last somebody is ready to talk to you! This means that ramk has something like this on his screen.

```
Message from kumarr@linux on pts/0 at 11:43 ...
```

Now you are both all set to carry on a conversation, because both have issued a write command for writing to each other and can consequently write to each other's terminal. So go ahead and pour out your woes to your only friend in the installation.

```
I wanted a program so badly but khan has not yet got it.  I

Which program? I am sure he must have tried

don't know what to do...
```

What is happening here? This can be disconcerting for beginners, but there is nothing mysterious about it. The communication is asynchronous and full duplex. Both sides can transmit and receive at the same time, and unless you wait until the other side has finished, there can always arise opportunities for confusion. What you need is a protocol to be adhered to so that the screen does not get cluttered up and cause bemusement. The thing to understand here is that there is no way of knowing when the other party has finished unless the protocol is set up and observed. This is because after every linefeed character the message is sent to the other side and there is nothing to restrict a message to one screen line. So the other party might respond before you have finished, especially if you have also entered a linefeed in your message.

There are many local conventions you could come up with for this, but let us see a protocol where we use ga at the end of each message and terminate the conversation with a bye. Now the previous conversation will be more intelligible to both parties.

```
I wanted a program badly but khan has not yet got it.

ga

Which program?

ga

I don't know what to do.

ga

I am sure he must have tried.

Bye

EOF

bye

^D
```

This was when everything was favourable. Sometimes you will have the following spot of bad luck.

```
[kumarr@linux kumarr]$ write ramk

write: ramk is not logged in

[kumarr@linux kumarr]$
```

This response is self explanatory. If you try to write to a user who is not logged in you will get this response from linux. Since write is an online command, it is not enough for the user to be a valid user on that installation. The user must actually be logged in to the system at the time you try to write to him. You can always make sure of that by doing a who first. If the person you want to write to is not logged in, you can save yourself the trouble of trying to write, knowing you will fail.

Then there are the times when you check out the users of the system and find something like this;

```
[kumarr@linux kumarr]$ who

ramk    tty1     Dec 10 20:47

kumarr  pts/0    Dec 10 22:59 (:0.0)

khanz   tty2     Dec 10 21:29

pramod  tty3     Dec 10 22:11

shyama  tty4     Dec 10 23:46

shyama  tty5     Dec 11 01:29
```

Now if you try to write to shyama you find

```
[kumarr@linux kumarr]$ write shyama

write: shyama is logged in more than once; writing to
tty4
```

If that user is logged in onto more than one place, you will find that Linux automatically writes to the lowest numbered terminal into which that person is logged in. That is usually fine, but if you want to specifically write to the person at some other terminal, you can do so by giving the terminal number of your recepient as well with the command itself. Thus

```
[kumarr@linux kumarr]$ write shyama tty5
```

will `write` to shyama at the terminal `tty5` rather than the default terminal of tty4. So you see that you `write` to a terminal, not to a user. It follows that you can `write` to yourself if you wish.

Are you at the mercy of others to be able to work peacefully? Not entirely, because there is a command in Linux to prevent other users from being able to write to your terminal. In Linux every device is a file and you could use the chmod command to turn off `write` permission for others to that file. Then users will not be able to `write` to your terminal and consequently will not be able to disturb you while you are working. But to use `write` you would need to know which device file corresponds to your terminal. An easier way is to use the mesg command.

```
[kumarr@linux kumarr]$ mesg n
```

Now anybody trying to write to you will get a message like this

```
write: kumarr has messages disabled on pts/0
```

If you just want to check the status of write permission on your terminal, just issue the mesg command without any arguments. To restore write permission, you can use the y argument to mesg. Any denial of write permission lasts only for the duration of your login session. You can have a different write status on some other terminal to which you are logged in. You need to be aware of the fact that the superuser can always `write` to your terminal, irrespective of mesg status.

You should use the mesg command only when you are doing something important and do not want to be interrupted. Just as it is rude to unnecessarily bother a person, it is also rude to shut your doors to others trying to reach you, perhaps with something important.

The `write` command gets its input from the standard input and so you can always write the contents of a file to a

```
[kumarr@linux kumarr]$ write ramk < mymessage
```

Here the recipient will see the contents of the file. So the message should be short, lest it be hard to read off the screen. This session will not be interactive, because it will end after the file has been sent off and the party at the other end will get an EOF. Another facility you have is to send a message to all users logged in at the time. This can be done with the wall command.

```
[kumarr@linux kumarr]$ wall

Don't any of you guys want lunch?

^D

[kumarr@linux kumarr]$

Broadcast message from kumarr(pts/0) (Sun Dec 12
21:38:50 2004):

Don't any of you guys want lunch?

EOF
```

Here even the sender gets the message. An ordinary user cannot be sure that all those logged in will get the message because some users might have disabled the receipt of messages. So this command is suitable for the superuser only. It is usually used to send important messages pertaining to the installation to the users.

☞ Check Your Progress 1

1) What do you think are the advantages and disadvantages of electronic communication compared to

   a) Postal Communication

   b) A telephone conversation

   ....

   ....

2) Can you think of a couple of other ways of sending output to another terminal other than the ones covered here?

   ....

   ....

3) What happens if a user logs out after you have started writing to him?

....

....

4) If you wanted to send some message requiring some preparation, how would you use to write?

....

....

Instant Messaging Applications

The `write` command that we looked at is rather rudimentary in that it does not offer very much by way of features or convenience. But it is useful if you want to communicate quick and fast with no set up required. For something that is much more powerful, you can use any of the instant messaging applications available today. With these applications you can communicate online with any user anywhere in the world if both of you are connected to the Internet. You can also hold conferences where more than two users can participate.

The act of using an instant messaging application is often referred to as chatting. This has become a very popular means of communication today. While not specific to Linux, chat applications are commonly used and will be described here briefly. Such applications are available from various providers but the one we will describe here is the Yahoo chat, called Yahoo messenger. This should not be construed as any endorsement of this particular application by the author or by IGNOU. This example has been used because you will not be able to appreciate the features of such applications unless we discuss them with reference to some specific application.

The Yahoo messenger application has many good features that make it useful and popular. It is available free of charge from the Yahoo website and is provided under different operating systems including Red Hat Linux. Once the file has been downloaded, it needs to be installed on your machine. This act requires root permission and will therefore not be discussed in this unit. We assume that your system administrator has done this for you. You can set up things in various ways but we will also assume here that you have to start up the application yourself whenever you want to chat with somebody or want to set up a conference. For this you just have to issue the command:

```
[kumarr@linux kumarr]$ /usr/bin/ymessenger
```

If you have set up your path properly in your login environment, you can omit the `/usr/bin` and simply say `ymessenger` after entering into graphics mode and creating a terminal window. This brings up two windows that read Yahoo Messenger and Login. To be able to use the application you must already have a Yahoo id. If you do not have one, you could click on the button in the Login window that says “Get a Yahoo! ID”. In the Login window you can now login by providing your id and password. The password is not echoed on the screen as you type and instead, you see an asterisk (*) for every character that you type. This is to provide some security because then somebody looking over your shoulder will not be able to make out your password easily.

There are also two checkboxes available in the Login window. The first one will tell the application to remember your id and password the next time you login. This can be very convenient, but there are two points to consider here. First, you might well forget your password if you do not use it for long. That can be inconvenient, though you can provide some information that you gave while signing up for the id and obtain a new password. That process takes time, however. Secondly, it is certainly most unwise to use that option unless you are working on a personal machine that you do not share with others, or at least with those you cannot trust. On a public computer, you must never set this option. Otherwise somebody could masquerade as you and perhaps perform objectionable acts in your name.

The second feature is to be able to login under invisible mode. When you enter the application, you will not be visible to others as having logged in. This will prevent others from trying to converse with you as they will not know you are online. We will see later that you can also put yourself into invisible mode after logging in normally.

The other Messenger window is blank in the beginning. While the Login window is visible, the Messenger window will not respond to your input (except for window manager operations that allow you to resize or close the window). It also shows in a status bar at the bottom that you are Not Connected.

Once you provide a valid id and password the login window closes, the application connects to the Internet and shows this in the status bar at the bottom of the Messenger window. It will also state that you are available, unless you have logged in invisible mode. You will initially not have any friends on the application so the middle portion of the window will be blank.

To be able to make use of the application, you need to have people you can chat with. These are called friends. So let us now add some friends to our list. To do this, click on the icon labelled Add, to open up a window that says “Add Friend”.

The Add Friend window shows you four steps that you need to take to add a friend. You must know that person’s id to be able to do so. Next, you have to decide what group that friend is to belong to. The application allows you to choose different kinds of groups into which you can classify your friends or contacts. You have the groups “Work” and “Personal” available to you by default. You can create a new group if you want to do so.

The ability to create groups is very useful as you can classify your contacts appropriately. For example, you could set up a group like “Family” where you put only your family members and relatives. Business or workplace contacts could be in the group called “Work” and so on.

In the next step you can enter the identity under which you want to add your friend. This will default to your login id.

In the last step you can enter a short message that your friend will see when she next logs in to the application, thereby telling her that she is now on your friend list. She could then choose to likewise add you to her friend list under the appropriate group. After this you can click on the “Add Friend” button to add the person to your list of friends. Now the person is displayed in the Messenger window under the appropriate group as your friend.

You can change your status by right clicking on the status bar at the bottom of the Messenger window. You get a list of status messages that you can choose from. These include “Busy”, “Not At Home” and so on. When people who have you on your friends list login, they will see that status message against your name. Friends who are online at the time are displayed in bold, the others are displayed in ordinary type. Under invisible mode, your friend list is displayed in italics.

Now suppose your friend is online. You can now begin a chat session with that person by clicking on the Message icon in the Messenger window. This brings up another window that shows up the name of the friend in the title bar at the top. Below that are two rows of menu options. This is followed below by a text area where you can see the messages that have been exchanged in that session. Below that is another area where you can type in your message.

For this message area there are several options available to you. You can set your messages to bold, italics or underline format by clicking on the B, I or U buttons. That changes the style of the text that you type subsequently in the message area. To stop bold, italics or underline, just click on the button again – it acts as a toggle. The recipient at the other end will see the message with the proper style. You can select more than one modification to the text style at the same time, such as bold and underline simultaneously.

Besides, there are options to change the font and the size of the characters you type. There is a drop down list available for both of these that includes a bewildering array of choices.

That is not all! You can also change the colour of the text that you type to any of some preset colours. As if that were not enough, you can create any custom colour you want to by specifying the RGB or HSB values of the colour. And then there are the different emoticons you can send by clicking on the smileys icon. There are several of them to display all kinds of feelings or emotions.

Once you have constructed your message, you can press the enter key or click the send button to have your recipient see what you typed. This is visible at your end also. Messages sent and received can be distinguished both by the id of the person as well as the colour.

When your conversation is over, or at any point that you wish, you can save the transcript of the entire discourse in a file on your disk. You could also cut and paste the conversation into a file in your word processor. If a person is not responding, you can “buzz” your friend to draw attention using the “Friend —> Buzz Friend” option in the menu bar. This will send an audible sound to the other end (will work only if the other party has their speakers on) as well as shaking the chat window of your correspondent.

There are several other options available and we will not be able to talk of all of them here. But one very useful and important facility is that of conferencing. Here you can invite a third party, and a fourth and more, to join in the conversation. Here all of you get to see what all of the others are saying. Thus you can conduct a meeting on-line. They are of course free to join your conference or to decline.

You can send a file to your recipient by using the “File —> Send” option or simply use the “Send File” option. You are then directed to another window where you can indicate the details of the file to be sent. With this you can also send a message to your friend. One more feature is the ability to ignore a user. You will then no longer receive any messages from them.

As your friends come online, you will receive a notification telling you about the event. Similarly when you come online, any users who have you on their friends list will be advised about the fact.

There are several preferences you can set to customize the look and feel of the application to your taste. These can be accessed by using the `"File —> Preferences"` menu option. You can, among other things, change the privacy options, decide the colour scheme used or set the alerts that you will receive.

So you see that an Instant Messaging application is much more sophisticated than the rudimentary `write` command that we saw earlier. However, for a quick and short conversation, you could still use `write`. These days you can also conduct a voice chat with your friends, effectively using your computer as a telephone. This would require a microphone and speakers at both ends and a reasonably good Internet connection. You can also conduct a voice conference.

☞ **Check Your Progress 2**

1) What is meant by being in invisible mode in Yahoo Messenger?

....

2) How do you being chatting with a friend?

....

3) What would you want to buzz a friend?

....

4) What are the different status massages that you can show others?

....

5) How can you save a transcript of the conversation you have had with a friend?

....

4.3 OFF-LINE COMMUNICATION

Let us now explore the facilities available in Linux for offline communication. This term is used here to refer to a method of communication where the other party is not necessarily on-line at the time. The recipient sees your message only the next time that she logs in. After looking at the array of features available in an instant messaging application, you might wonder what good it would be to use any other method! But consider some of its limitations and disadvantages.

The other party has to be available at the same time as you are. This can be inconvenient across different time zones.

Unless you keep yourself focussed, the conversation can tend to ramble and go on an on. This means more time spent on the chat proper.

If you are using text mode, then typing things out quickly can be difficult and tiresome unless you are a trained typist.

Infrastructural issues are important. If either side has a power or network outage, the conversation cannot happen.

Lengthy files cannot be sent, or even if they are sent, there is no chance for the recipient to peruse them.

We can get around many of these limitations by using an offline mode of communication such as electronic mail. While some of the good features of instant messaging might not be available, there are some advantages to it.

Communication is asynchronous. You send your message and the other party can reply at leisure. If they are not available, the message resides in their mailbox until they are ready to look at it.

File attachments can be sent and the other party can reply after examining the file.

Typing speed is not much of an issue as there is no one impatiently waiting for your reply at the other end.

Infrastructural bottlenecks are of less significance. The message can be sent or received as and when resources become available.

At the same time, if both parties are online, e-mail can be almost like an online means of communication as people exchange e-mail messages rapidly one after the other. We will therefore look at the e-mail facilities available in Linux. Here we will not concern ourselves with the mechanisms of setting up an e-mail system, which is the job of the system administrator or the network specialist at your installation. We assume that has been done and we will only look at an e-mail client to explore the features and facilities that it offers.

Linux has a rudimentary e-mail client in the mail command. This is a text oriented facility that is difficult to use, especially in today's scenario where we are all accustomed to graphical interfaces. It works at the command line level and allows us to send, receive and otherwise manage e-mail messages. But here we will look at a more sophisticated e-mail client – Evolution from Ximian. To be able to use it, it needs to be set up on your machine. Here we assume that the basic set up has been done for you and that you are ready to use it as your primary e-mail client.

To start up the client, click on the little arrow near the Red Hat on your bottom tray. This brings up a menu of choices from which you need to choose "Internet —> Evolution Email". This takes you to the main window panel on your screen. This window has several parts that are briefly described below. You will find that there are many ways of reaching a particular option. For example, to see your inbox, you can click on the Inbox link on the right hand panel in the summary page, or choose Inbox from the left hand panel, called the Shortcut bar, itself. We will here look briefly at some of the main features of the client.

The client has features such as helping you keep track of your appointments and tasks that you need to do. Here we will not dwell on those aspects as we would like to focus only on the communication aspects of the client that are made available through the e-mail related facilities.

Menu Bar

At the top is a menu bar with the options File, View, Actions, Tools and Help. Each of these in turn has several options. The bar is context sensitive. For example, when you are viewing the summary, you will not see the Edit button. Likewise, the options that appear under the View or Actions button depend on where you are currently.

The View option lets you choose the appearance of the window. You can choose to see the Shortcut bar on the left or to hide it. You can also choose to see or hide a Folder bar that shows all the mail folders that you have. This appears to the right of the Shortcut bar.

The Actions option lets you send or receive your messages through your e-mail server that has been configured for your client. If you choose this option then all messages that have arrived on the server will be transmitted to your client and will appear in your Inbox. Also any messages in your Outbox will be sent out.

The Tools option lets you change the Settings of the client. This will allow a host of preferences to be set. These include the settings related to mail messages, message composition, servers, contact lists and others. You can also configure the synchronization of your computer with a Palm Pilot.

The Help option gives you a full description of how to configure and use the Evolution e-mail client. You should refer to it for all the details that we will not be able to discuss here.

The File option lets you create a new mail message, set up an appointment or add a task. You can also work with folders, import external files, print or work offline. Many of these choices can also be used through other navigational paths.

Shortcut bar

This bar appears on the leftmost part of the window, unless you have chosen to hide it. It has several choices.

The Summary shows you some weather information in the centre pane, while the right hand pane shows the number of messages in your outbox and inbox. You also see any appointments and tasks that you have left. All of these are links that take you to the respective feature. So clicking on the "Inbox" will take you to your inbox.

Clicking on Inbox takes you straightaway to your Inbox, where you can see a list of the messages that you have received with the sender, subject and date. You see the total number of messages and the number of new messages as well.

There are also shortcuts to your appointments calendar and todo list.

You can maintain you Contact list by clicking on Contacts.

If you right click on the Shortcut bar, you get a menu whereby you can change the appearance of the bar. You can now —

- Hide the Shortcut bar, whereupon you can get more space for the actual option that you are at currently. So if you are in your Inbox and you hide the bar, the space that becomes available is now used for your Inbox itself. You can get back the bar by clicking on the View option in the menu bar at the top of the window and selecting the Shortcut bar.
- Choose to see large or small icons in the Shortcut bar according to taste. Small icons take up little space and give a compact appearance.
- You can create another Shortcut group where you can put in your own shortcuts to folders that you choose, such as a folder that you have created.
- You can change the names of groups or remove groups that you have created.

Inbox

The Inbox shows you at the top statistics on the number of new and total messages that you have. By clicking on a column header such as "From", "Subject" or "Date", you can sort the list in ascending or descending order, or even remove the column. You can also add columns of your own by selecting one of the available choices and you can place the column wherever you like.

When you click on a message summary you can see the actual message in the bottom half of the window. You can now perform the usual actions of replying to the message, deleting it or forwarding it to another set of recipients.

You can search through your messages based on several criteria, such as

What the subject does or does not contain

What the body does or does not contain

What the sender's name contains

What the recipient's name contains

Your own custom criteria that you can build up using the options provided

Composing Mail

One way of beginning to compose mail is to select Actions —> Compose New Message when in the Inbox. This brings up the "Compose a Message" window that lets you enter.

The "To" or recipient information

Information on those to whom you want to copy the message, using the "Cc" field

You can choose to send a "blind" copy by using the "Bcc" field. Other recipients will not be able to see that the mail has gone to the Bcc addresses, but they will be able to see the other addresses to which the mail has been Cc'd. To be able to use the Bcc field, you might have to choose the View option on the top menu bar and check the Bcc option in it.

For the above three fields, you can bring up your contact list by clicking on the To, Cc or Bcc buttons. Now you can select by name the contacts to whom you want to send the mail. When the list appears, you can place contacts in any of the three fields that you need. This way you do not have to remember any e-mail addresses, nor do you have to type them in.

The subject line. Although this is not compulsory, it is good to enter a subject so that the recipient can make out what the message is about when it sits in her inbox. The subject should be descriptive enough so that this can be done.

The Format option on the menu bar in the Compose windows has many features.

You can now start entering the message itself. Here again you have many features available. To begin with, the message can be in plain text or in HTML. In either case, you can make the text bold, underline it, put it in italics or use strikethrough mode.

You can choose the font size for your message.

You can align the text that is entered – left, centered or right alignment are possible.

You can change the colour of the text from among several preset choices, or you can even create your own custom colour.

You can create bulleted or numbered lists, with choices for different kinds of numbering such as Roman or alphabetical.

You can indent your text as desired.

While you are entering your message, you have several facilities available through the "Edit" option on the menu bar.

You can undo an action that you have performed by mistake, such as changing the indent or the font colour.

If you find you have undone something by mistake, you can redo that same action.

You can cut or copy and paste a block of text from another file to the message window or vice versa. The cut option will delete the text from the original location, while the copy option will leave the original as it is.

You can find and replace text in your message.

You can spell check the text that you have entered in your message.

Besides, you can add an attachment to your message. This attachment can be any file that you like to put in. Various emoticons, or smileys, can be inserted into the message body as well. Apart from this, you can insert images that you have, hypertext links, ruler lines, text and html files directly into the body of the message. Having done all this and composed your message, you can now use the “Send” icon on the toolbar, just below the menu bar, to send your message off to the recipients.

If you wish, you can save your message as a file on the disk using the “File” option in the menu bar. You can also save your message in your Drafts folder for editing and sending later. This is useful when you are not able to complete an elaborate message in one session and want to continue to work on it later.

Folders

To be more organised, you can store your mail in various folders. The default set of mail folders consists of Inbox, Outbox, Drafts, Sent and Trash, besides which there is a folder for your contacts. There are also folders for your calendar and todo list, but we do not look at them here.

The Inbox stores all messages that you have received. New messages that you have not read are to be seen in bold. The Outbox stores messages that you have composed and that you have asked to be sent. They remain here until they have been sent off successfully. If, for example, the network link is down, the message might remain here for some time till the link is restored. The Drafts folder contains messages that you have composed and chosen to save instead of sending off. As we saw before, this could be because you need to work on the message some more in a later session. The Sent folder contains messages that have been successfully sent off. They are moved here from the Outbox when this happens.

You can delete a message from any of these folders if you do not want them lying around. Such messages are not removed permanently at that time but are instead moved to the Trash folder. This gives you a chance to undo the delete if you realize you have made a mistake, or if you change your mind. To remove a message for good, you need to delete it from the Trash folder.

You can also create your own folders by using the File —> New —> Folder option. You will be asked to specify under which folder you want this to be created. You might find it useful to create new folders to organise your messages better. For example, instead of letting all your incoming mail clutter up your Inbox, you might like to store it in folders organized by sender name or organisation.

Contacts

You can create a list of contacts with various details like their name, organisation, title, contact details with e-mail, postal address, phone numbers and so on. This can have other notes on them together with their birthday, spouse’s name, anniversary, web page and other information too.

You can remove contacts and edit their information as well. This contact list can then be used while sending mail, as already described. You do not have to remember e-mail addresses and can just use the name. You can search for contacts using various fields as well.

Besides what we have described above, you can set a host of preferences for the look and feel of your client. With this wealth of features, it is a real pleasure composing and sending e-mail on your Linux machine. This is a big advance over the rudimentary facilities offered by the mail command that is inbuilt in Linux.

☞ **Check Your Progress 3**

1) What are some of the advantages of e-mail over instant messaging?

....

....

2) Why would you not want to use the inbuilt mail command for managing your e-mail?

....

....

3) What is the utility of having different folders set up in your mail client?

....

....

4) Why would you want to set up an address book?

....

....

4.4 APACHE SERVER SETTINGS

The Apache webserver is a very popular webserver used by many websites around the world. Here we will take a brief look at its facilities and see how you can perform basic configuration of the webserver. While this will not make you an expert, it will give you an introduction and start you off.

In Linux, Apache version 2.0 is available as your webserver. This package can be installed while installing Linux. If it is not already available, then you can install the package, that is called httpd, using the command

```
[root@linux root]# rpm -i httpd
```

The configuration of Apache is specified in the configuration file that lies in `/etc/httpd/conf/httpd.conf`. You could make entries into this file by hand if you wish. When you install the package, a default version of this file is put in automatically. However, hand configuring this file is only for experts, and you should not try it until you are thoroughly familiar with it.

You can start the http server by issuing the command

```
[root@linux root]# httpd -k start
```

If you now start up your browser and point to http://localhost, you should see the Apache test page. This indicates that the Apache webserver has been configured properly and that it is working correctly. If there is something wrong with the configuration you will get an error message.

To be able to work with the webserver, you will need superuser or root access to your machine. Ordinary users cannot configure the webserver or stop or start it. This is because the impact of a reconfiguration is installation wide and is not confined to the user who makes the change. Also it requires access to facilities that are accessible only to the superuser.

What if you are not an expert user and are not very familiar with the webserver? How do you begin to configure it? One way is to use the graphical HTTP Configuration Tool that comes with Linux. This gives you a user friendly interface and saves you from having to know any complex commands. To be able to use this tool, apart from superuser or root access to the machine, you need to have the X-Window system running. This needs setting up by your administrator, and after logging in, you can invoke it using the command

```
[root@linux root]# startx
```

You need to understand that if you use the HTTP Configuration Tool, you cannot configure the webserver by hand. This is because the tool generates the file after you exit the tool. Any changes you have made are therefore not preserved. Also there could be some modules that you have added to your webserver. The tool only supports modules that are made available with Linux. If you use any other modules from other sources, they cannot be configured using this tool.

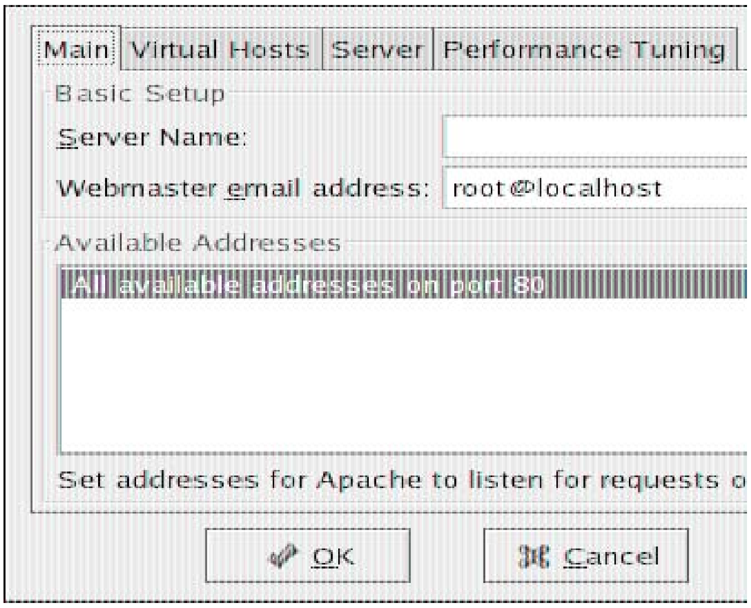
To start the tool click on the red hat icon with a little upward pointing arrow. This is the Main Menu Button for Red Hat Linux. So you need to say Main Menu Button —> System Settings —> Server Settings —> HTTP Server. If you wish, you can also start up the tool from the command line at the root prompt.

```
[root@linux root]# redhat-config-httpd
```

This brings up the main window of the tool, shown in *Figure 1*. You are currently at the Main tab where you enter the basic settings for the webserver.

Main

In the Server Name, you should enter the complete or fully qualified domain name of the machine on which you are doing the configuration. This could be something like www.mycompany.com, or www.mycompany.co.in. If you do not give a valid name, when the webserver starts up, it tries to obtain the name from the IP address of the machine. The name of the machine that you give does not have to be the same as this one.



Basic Setup

Server Name:

Webmaster email address: root@localhost

Available Addresses

All available addresses on port 80

Set addresses for Apache to listen for requests o

Figure 1: Main Window of HTTP Configuration Tool

The next field that you enter is the e-mail address of the webmaster. It defaults to root@localhost as can be seen in the figure. You can configure the webserver's error pages to contain an e-mail address. This can then be used by users to send e-mail to report errors. You can make the address something like

webmaster@mycompany.com.

Next you can set up the ports on which the webserver should listen for requests. The default is port 80 for non-secure communication. You can add an address by clicking on the Add button at the right. This brings up another window where you can either choose to Listen to all addresses or specify an IP address. You can also specify the port number at which you want to listen to requests from that address. You will have to make a separate entry for every combination of IP address and port number. It is better not to use domain names because of the possibility of DNS lookup failures.

You can edit or delete an entry by selecting it and then choosing the appropriate option from the buttons on the right. Note here that while the edit option lets you cancel your edit and seeks confirmation from the OK button before your entries take effect, the delete button immediately deletes the entry without further ado or warning.

Virtual Hosts

This completes your basic settings. Now you should go to the next tab, that is "Virtual Hosts". On clicking here you will see the virtual hosts listed by name and address. Initially both will say "Default Virtual Host". In the lower portion of the window is a button labelled "Edit Default Settings". You need to click on this to bring up another window titled "Virtual Host Properties". On the left of this window you will be on the option for "Site Configuration". On the right you see entries for the Directory Search Page and the Error Page. You should not need to change these settings in the beginning. However, the tool does give you options to Add, Edit or Delete these entries.

The Directory Search Page is the page where the webserver looks for content to serve out when a user points a browser at the server's URL without specifying any directory or with a directory name ending in a /. These are searched for in the order that they appear here.

The Error Page is used to gracefully handle errors. The message in the footer is what the user sees in case of an error. If you click on the Edit button, you can change this behaviour. The user can be redirected to any URL that you choose in such a situation, and this can be an internal or external one. For an internal URL choose the File option. Let us say you want to show the user a message that you have stored in a file called badrequest.html whenever he makes an invalid request. In the Error Page, select Bad Request and click on Edit. In the behaviour, from the drop down list, choose File and enter the name of the file. This file must be under the Document Root directory.

The last menu option here is for the Default Error Page Footer. Here you can choose the default, no footer or the default together with the e-mail address of the person maintaining the website, that was specified during the basic settings.

Having done with Site Configuration, you can move on to the Logging menu. Here you can configure the Transfer log and the Error log. The transfer log logs all connection attempts to the webserver, with their IP address, date and time and the file that the client is trying to access. The default for this is the file `/var/log/httpd/access_log`. Similarly the error log holds all errors that the webserver encounters and the default for this is the file called `error_log` in the same directory.

You can change these default log files by entering either an absolute pathname or a pathname relative to the server root. You could also choose a custom log format by entering a string that defines the format, but we will not go into this here.

The log level determines the amount of logging that goes on and you can set to one of the eight possible levels, from Emergency that logs the least amount, to debug level that puts in a large amount of information. The default is to log only errors or more severe problems.

The last option here is the Reverse DNS Lookup, which you should leave at the "No Reverse Lookup". You can also do a Reverse Lookup or a Double Reverse Lookup, but this is not recommended because of the load it would place on your server and the Internet in general.

The next menu option under Virtual Hosts is Environment Variables. Here you can control the environment variables that are passed onto Common Gateway Interface (CGI) or Server Side Include (SSI) pages. This has three options – to set, to pass and to prevent an environment from being passed to the CGI script.

In the "Set for CGI Scripts" section, you can choose the Add button to add an environment variable to be sent to CGI scripts. This brings up a small window called "Environment Variable" where you can enter the name of the variable and its value. Then choose OK to add it to the list. Similarly in the "Pass to CGI Scripts" section, you can choose the Add button to bring up a window where you can enter the name of the environment variable. This passes to any CGI scripts the value of that variable when the webserver was started. Click on OK to add this to the list. Finally, in the "Unset for CGI Scripts" section, you can enter the name of variables that you do not want to pass to CGI scripts. For all the three cases, you can also edit and delete entries alreay made.

The last menu option here is Directories. Here you can set various options for different directories. At the top are options that are the default for all directories not mentioned in the lower portion of the window. You can edit these default options by using the Edit button the right. In the bottom part of the window you can set options for specific directories. For this click on the Add button at the right.

You now see a "Directory Options" window with various sections. In the Order section, you can either allow all machines to access the directory, process a deny list first or process an allow list first. In the Deny List and Allow List sections, you can either select all hosts or specify hosts based on matching with a partially specified domain name, an IP address or a subnet. You can also put in the directory to which these options apply. Moreover, you can choose to allow directives in a `.htpaccess` file to take precedence. There are also some options that you can set. These same options can also be set for the default directories. These include allowing CGI scripts to be executed, allowing server side includes and allowing the following of symbolic links. You can also edit and delete directories that have options already set.

You would recollect that we have done all the above using the "Edit Default Settings" button, which means that what we did applies to all virtual hosts. Different virtual hosts can be hosted on the same physical machine, and they can have different IP addresses, ports or names. We will now see how to manage virtual hosts in Apache.

For every virtual host that you add, you can configure directories, environment variables, logging and site options the same way as described earlier. To add a host, click on the Add button which will bring you to the "General Options" button. You will now see a window where you can specify the name of the host, its document root, and the e-mail address for the webmaster. In the Host Information section, you can choose the type of host, which can be default, IP based or Name based. You should have only one default virtual host. For IP based virtual hosts you have to specify the IP address. This can be in the form `IP:port` if you want to specify the port as well, and you can specify multiple addresses by separating them with a space. You can likewise configure name based hosts by putting in the information that is asked for. Name based virtual hosts cannot work with SSL and can work only with a non-secure webserver. You can set up SSL options by choosing the SSL menu option for your virtual host. This requires that your webserver have been set up to allow SSL support, and will require you to allow access through port 443 in the basic settings.

Server

The next tab is “Server” and it allows you to configure some basic settings for your webserver. You should not normally have to alter these, at least the lock file and the PID file. The Core dump directory is where the server dumps core if it crashes.

The Apache webserver must not be run as root as that would expose many security holes in your system. The user for the server is specified in the User section, and by default the user is Apache. Here you can also enter the group to which that user belongs. This too is Apache by default. The user id is what determines access to the system resources. Any file that cannot be accessed by this user cannot be accessible to the server and will return an error. Also any CGI scripts run are run with the user id of this user. So you must take care to see that any scripts that are not to be run by the outside world cannot be accessed by the webserver user. Also any files that you do not want to allow others to read should not be accessible to the webserver user.

You must ensure that the core dump directory permissions allow the Apache user to write to it. If not, then it will not be possible to write core to the disk in case of a crash

Performance Tuning

This is the last tab in the HTTP Configuration Tool. The window that it opens up has a few options that we look at here. In this case too the default options are likely to be appropriate choices for most situations.

The “Max Number of Connections” is the upper limit on the number of simultaneous client requests that can be handled by the webserver. This needs to be below 256 and is 150 by default. For a higher value, above 256, you will have to compile the Apache webserver again with the proper options – not recommended for beginners. If there are more client requests they will be refused unless connections become available.

The “Connection Timeout” setting sets the number of seconds for which the webserver will wait for a response to a communication request. The default value, as you can see, is 300 seconds. You can also set the maximum number of requests that are allowed per connection or allow any number of requests. If you allow persistent connections, the server will keep a connection open even after a request has been serviced. The time for which it will do so can be specified in “Timeout for next Connection”. This should not be set too high as it could slow down the webserver as it waits for another request from clients.

With this we have looked at how to perform the basic configuration for the Apache webserver. There are of course many features that we have not touched upon, but after reading the material here you should be able to at least start off.

4.5 NETWORK SERVER SETTINGS

In this section we will look at how to set up your machine to serve as a Domain Name server and as a Network File server. A domain name server helps convert domain names to IP addresses so that human beings do not have to remember incomprehensible strings of numbers and can instead work with names that they find much more intelligible. You will need to have sufficient understanding of the domain name service and the program, bind, that provides the service.

4.5.1 Domain Name Server

It is certainly possible to configure bind by hand, but that requires a good understanding of the format of the configuration files. If you are not an expert, the Bind Configuration Tool provides a graphical, user friendly way of performing basic configuration of the DNS service. However, any complex configuration will not be possible through this tool.

Like other server configurations, setting up the DNS service also requires you to have root or superuser access on your machine. Moreover, to run the tool, you need to be working in the X-Window system so that graphical facilities are available. You can start the tool through the command line at the root prompt, though.

[root@linux root]# `redhat-config-bind`

To start the tool from your Gnome desktop, click on the red hat icon with a little upward pointing arrow. This is the Main Menu Button for Red Hat Linux. So you need to say Main Menu Button —> System Settings —> Server Settings —> Domain Name Service. This will bring up the main window of the tool, as shown in *Figure 2*.

The configuration file for bind is `/etc/named.conf`. If you are going to use the configuration tool, do not edit the file by hand as all changes you make will be lost when the tool writes out to the file when it is closed. Unlike the HTTP Configuration Tool, though, you can also put in your own configuration that you make by hand, by placing it in the file `/etc/named.custom`.

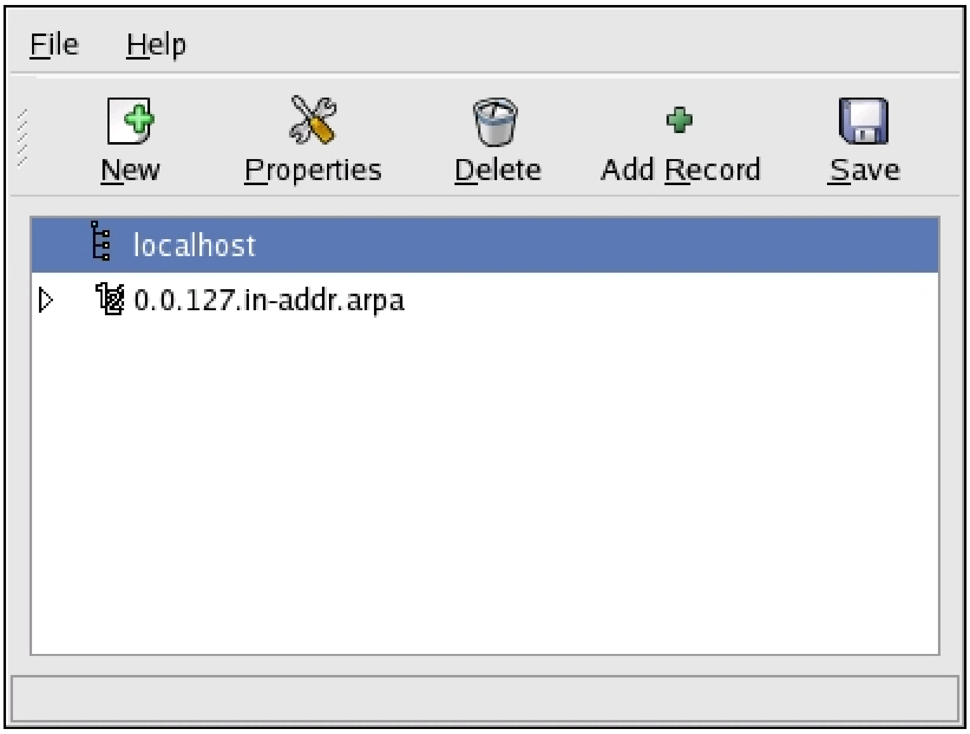


Figure 2: Bind Configuration Tool

You can work with the zone files and can add, delete or edit zones, which can be a forward master, a reverse master or slaves. To save your changes you need to use the menu option File —> Save or just use the save button. If you want to quit without making changes, use File —> Quit. The changes you made are reloaded by named so that they take effect right away.

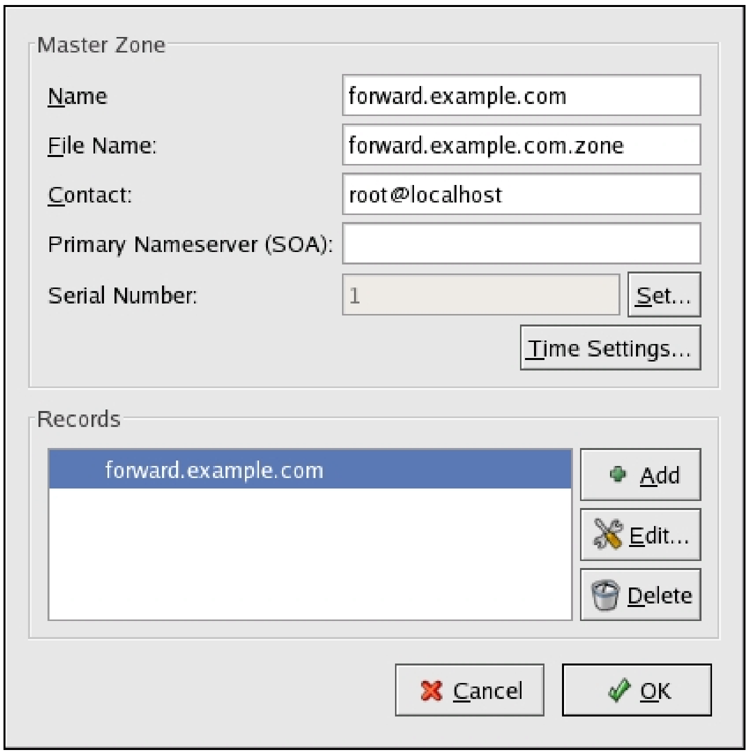
You can add zones by clicking on the New button in the main window. A window titled “Select a zone type” appears where you can select from Forward, Reverse or Slave.

Forward Master

A window for “Name to IP Translations”, shown in Figure 3, appears with the name of the zone that you just entered. The zone file name is the same as that of the zone, but with a `.zone` suffix. This filename is relative to the `/var/named` directory. In the contact field you can enter the e-mail address of the primary contact for the domain. The default, `root@localhost`, is probably not the best choice.

Next you need to enter the name of server that is authoritative for this domain. This is used to create the Start of Authority record for the domain. You must specify a primary nameserver and enter a record for it. The serial number has to be incremented each time there is any change to the configuration file, so that slave servers will update their own files. Usually it is set to the date of the file followed by a number, such as 20041226001. To change the serial number from the default of 1, click on the set button and enter it.

The button for “Time Settings...” lets you change the different life cycle times that you have to specify in a DNS file. These are the time to refresh, retry, expire and the minimum time to live (TTL). The values are to be entered in seconds, and here the defaults that appear, 28800, 7200, 604800 and 86400, are reasonable enough that you can leave them alone. However, depending on the characteristics of your site, you might have to alter these values.


Master Zone

Name: forward.example.com

File Name: forward.example.com.zone

Contact: root@localhost

Primary Nameserver (SOA):

Serial Number: 1

Records

forward.example.com

Figure 3: Forward Master Configuration

Finally you can add records for hosts, aliases and nameservers by clicking on the Add button in the Records section. You must add an entry for a nameserver. When you are done you can save the configuration.

Reverse Master

When you choose to add a reverse master zone, you have to enter the first three octets of a valid IP address. The tool performs the necessary checks to see that the address entered is of a valid format. After you click OK, you will see a new window titled “IP to Name Translations” that shows several fields for you to enter. The first is the IP address field that is the same as the IP address that you just entered. The name of the zone is constructed from this by putting the octets in reverse order and appending “.in-addr.arpa”. The name of the zone file is the same as this with the suffix being ".zone". The contact field contains the e-mail address for the primary contact for the zone. As in the case of a forward master, you have to enter the name of the server that is authoritative for the domain together with the serial number. This is automatically incremented by the tool and can also be set manually by using the Set button.

You have to again add the zone life cycle information and provide at least one name server for the zone. The portion here that is different from the forward zone is the reverse lookup table of that gives the hostname for each IP address in the zone. When you click on the Add button you are prompted to add the IP address and the complete hostname (ending with a period) for hosts in the zone.

At the end of this you can save the configuration or quit. If you save, the named service will take cognizance of the changes and they will take effect. The window for Reverse Master configuration is shown in *Figure 4*.

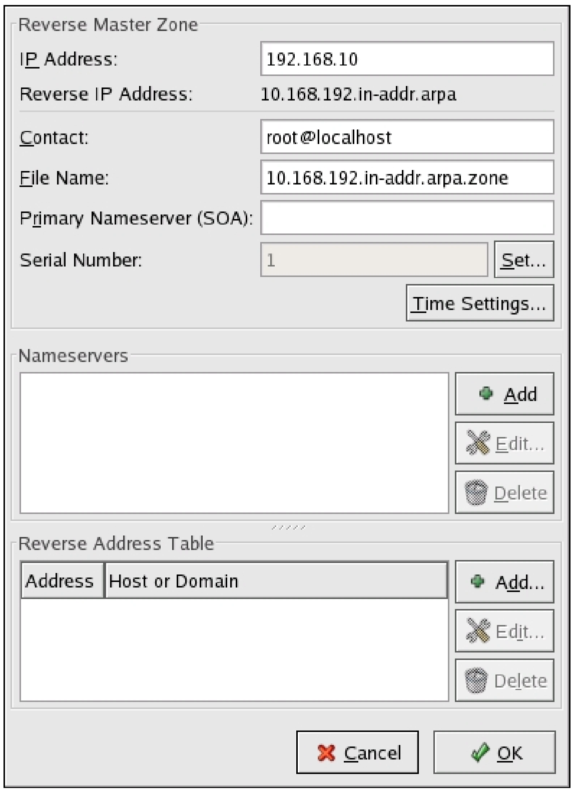

Reverse Master Zone
IP Address: 192.168.10
Reverse IP Address: 10.168.192.in-addr.arpa
Contact: root@localhost
File Name: 10.168.192.in-addr.arpa.zone
Primary Nameserver (SOA):
Serial Number: 1

Figure 4: Reverse Master Configuration Window

Slave Zones

You can also add a secondary master or a slave zone to your configuration. These serve as backups to the primary master for the zone. Here you have to add information on the nameservers from which the slave is to pick up its data, together with the name of the DNS database file. This name is specified relative to the /var/named directory. The nameservers are specified as IP addresses.

4.5.2 Network File Server

You can configure a Linux machine to work with a Network File System (NFS), where files on other machines on the network can be made available as if they were local files. A Linux machine can work as an NFS client, whereby it accesses files on the network. You can also configure your Linux machine as an NFS server, whereby you can let other machines access files on yours. In this section we will look at how this can be done.

As we have seen for the webserver and DNS server cases, although you can construct an NFS configuration file by hand, Linux comes with a tool to ease the task. This is the NFS Server Configuration Tool. It requires superuser or root access to use the tool. Being graphical, you must have the X-Window system running to be able to use the tool. But you can still start up the tool from the command line by issuing the following command at the root prompt

[root@linux root]# `redhat-config-nfs`

The NFS Server Configuration Tool both reads from and writes to the configuration file /etc/exports, and so you can modify the configuration file by hand after using the tool. If you use the tool again later, it will understand and recognize your changes, provided you did the configuration correctly with the proper syntax. The main window of the tool is shown in *Figure 5* below:

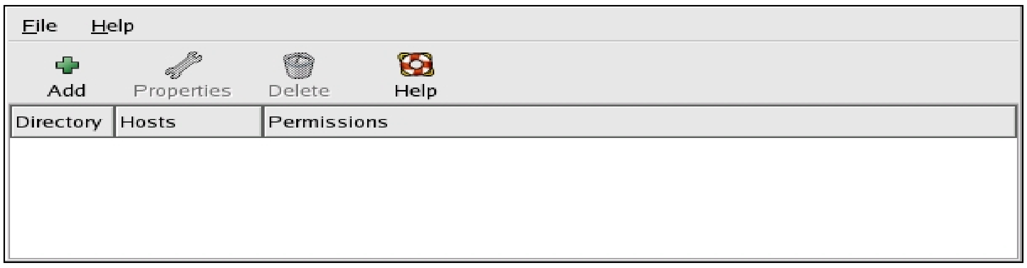


Figure 5: NFS Server Configuration Tool Main Window

To share a directory, called adding an NFS share, you need to click on the Add button above. This brings up a window with the title "Add NFS Share" that has three tabs. The "Basic" tab allows you to specify a directory and a radio button lets you decide whether you want to allow read-write or read only access to others on it. You also have to specify the machines or hosts that are to be allowed access to that directory. This can be done by:

- Giving a fully qualified domain name. This should be something your machine can resolve to an IP address.
- Giving an IP address.
- Giving a host name, again your machine should be able to resolve this to an IP address.
- Giving a group of machines by specifying them as a domain name or host name with wildcards. You can use a * for matching any number of characters except a period, and a ? to match any single character.
- Giving an IP network by specifying the network and a / followed by the number of bits in the netmask, or by specifying the netmask itself.

Doing the above makes the directory accessible to the host or hosts with permissions as desired.

The "General Options" tab has five options as described below:

- If you want to allow ordinary users to be able to start the NFS service and allow shares, you have to allow the service to be started on ports higher than 1024. This does make the service less secure because the share does not require the concurrence of the administrator.
- You can decide to allow insecure file locking.
- You can decide to disable subtree checking. This is useful if you have exported an entire file system, because your server will no longer check to see whether a file requested by a client is in the directory that has been shared.
- You can choose to force synchronization of writes immediately.
- You can choose to disable synchronization of write options, where the server first writes out to disk the changes caused by a request before replying to it.

The "User Access" tab has the following options that you can set.

- You can allow the superuser of a client machine root privileges on your machine. This is a big security risk and should be used only if necessary. Otherwise, by default, even the root user of the client is treated as an anonymous user on your machine.
- You can map all users on the client to the anonymous user on your machine. If you choose this option, you can set the user id and group id of the anonymous user.

You can now click on the OK button to save the configuration you have made. Of course you can add as many directories as you wish to share. You can also edit directory properties by selecting it and choosing the "Properties" button in the main window. This button is initially greyed out when there are no directories shared. Similarly you can delete a directory by selecting it and choosing the "Delete" button. Whether you add, edit or delete a directory, the configuration takes effect immediately after you save it. This is done by generating the new /etc/exports file and restarting the NFS server daemon.

4.6 SUMMARY

This brings us to the end of this unit. Here we have looked at two broad themes – communicating with other users and setting up your machine as a server for different kinds of services.

In user communication, we have looked at online and offline communication. We have seen how to use the write command to quickly communicate with other users on your machine. It is a simple command that provides only a rudimentary communication facility. We then looked at an instant messaging application from Yahoo! That provides many sophisticated features and allows us to talk to anyone else connected to the Internet. The only demand this places on us is that it needs a good connection to the Internet, while the write command has no such constraints.

For offline communication we looked at the Evolution e-mail client from Ximian. This allows us to communicate asynchronously with users who might not be online at the same time that we are. It has a wealth of features that make the task of sending and receiving mail easy and we therefore did not consider the inbuilt mail command in Linux.

We then looked at setting up three different kinds of services. First was the Apache webserver that is available in Linux and is one of the most widely used webservers. We saw how to use the configuration tool provided with Linux to configure this webserver.

Next we saw how to configure our machine as a DNS server using the configuration tool available in Linux. Finally, we saw how to set up our machine so as to make it an NFS server.

All the server settings were discussed with reference to the user friendly, graphical configuration tools available in linux.

4.7 SOLUTIONS/ ANSWERS

Check Your Progress 1

1) a) Postal communication is slow and can be unreliable. But you can send physical material by post, something that is quite impossible electronically.

b) A telephone conversation is synchronous and we can listen to the other party's response immediately. We also know if the other party has got our message. When communicating using a computer, we can use asynchronous methods so that both parties do not have to be present simultaneously.

2) Try

```
cat /dev/tty04
```

where /dev/tty04 is the device file corresponding to the desired terminal.

One can also try

```
cp msgfile > /dev/tty04
```

3) You get an error message like

```
write: Input/output error
```

4) Prepare the message and store it in a file such as msgfile. For this you can use vi or any other editor. Then say

```
write khanz < msgfile
```

because write reads the standard input. You will have to see that khanz is logged in. If he is logged in from more than one terminal you will need to specify the terminal as well.

Check Your Progress 2

1) You can login but others who have you in your friend list will not be able to see that you are online. This is called logging in invisible mode.
2) Try Yourself
3) When she has not responded to your message for quite some time and you want to draw her attention.
4) Try Yourself
5) One way is to select the transcript and copy it to a file in your favourite word processor.

Check Your Progress 3

1) Try Yourself
2) The mail command has a rudimentary, hard to use interface. It does not have the features one would expect to have in a good e-mail client.
3) It allows one to organise one's mail better. Related messages can be stored in different folders rather than all messages being in one folder with hundreds of messages. This makes it easier to look for a message.
4) This saves us from the trouble of having to remember a large number of e-mail addresses. One can save the contact details of a person and then just use a short name or his nickname to send mail to him.

4.8 FURTHER READINGS

There are a host of resources available for further reading on the subject of Red Hat Linux version 9.0.

1. http://www.redhat.com/docs/manuals/linux
2. http://www.linux.org gives among other information, a list of good books on Red Hat Linux.
3. Consider joining a good linux mailing list.

UNIT 5 UNIX SYSTEM ADMINISTRATION

Structure **Page Nos.**

5.0 INTRODUCTION

In the previous units you have been introduced to Linux and have got an idea of its features and the facilities available in it. In this unit we will look at how to administer a Linux system and take care of it so that you can make use of its power. This would mean being able to install Linux on a machine and configuring and setting it up so that you could start working on it. It also requires maintaining the machine subsequently, such as by adding user accounts and keeping your data safe by backing it up.

The activities described in this unit are mostly performed as the superuser. So you have to be very careful and understand the commands you issue thoroughly, because Linux does not conduct many checks on what the superuser asks it to do. As an ordinary user you could not modify a file if you did not have permission, but the superuser can change any file irrespective of its permission settings. This might seem like a convenience, and it is indeed so. But it also means that you have the potential to cause severe damage to the installation through one mistakenly issued command.

5.1 OBJECTIVES

After completing this unit, you should be able to have an understanding of basic system administration tasks and be able to perform basic installation, configuration and maintenance of a Linux system. Some of the abilities you should have acquired are:

- understand what is meant by system administration;
- understand the responsibilities of the system administrator;

- install Linux on a personal computer;
- understand how to boot a Linux system and what goes on while starting it up;
- how to add and maintain user accounts;
- understand character and block special files, and
- know how you can back up data from the system and restore it when required.

5.2 SYSTEM ADMINISTRATION

You have started working on a Linux system and have learnt some of the basic facilities provided by the operating system. When you began your exploration of Linux, you were given a user account on a machine. For this to have happened, what are the actions that somebody would have performed for you? Let us try to work backwards and figure this out.

First of all, somebody would have had to load Linux on the machine that you are working on. This would mean having to know how the operating system is to be loaded on to your machine. Secondly, having got the machine working under Linux, somebody would have to set up user accounts and give them permission to log in to the machine and work on it. There are several other such activities that one can think of.

Configuring the system to connect to the local area network (LAN)

Adding new hardware devices to the system, such as hard disks, a printer or a CD-ROM drive. Taking backups of the important system and configuration files.

Booting up and shutting down the system as needed. Restoring the system to normal working in case of a crash, by using the backups.

Fixing software problems that might arise, such as a corrupted system file. Addressing issues such as performance tuning if the system is slow. Making sure that the system is secure from the assault of unscrupulous persons. Installing operating system upgrades and patches as required. Installing and configuring any new software that has been acquired by the organisation. Providing required network services such as e-mail, Internet connectivity and other services as appropriate. Helping ordinary users in trouble, such as those who might have forgotten their password.

Being able to do the above requires much more knowledge of the working of Linux than you have obtained so far. It also requires experience with various commands, tools and utilities. The tasks listed above are not the only ones that need to be performed for ordinary users to make use of the system and are only indicative.

Moreover, most of these responsibilities can be performed only by persons having superuser or root access to the system. Ordinary users, without root permission, cannot run many of the required commands. Because of this, these tasks have to be performed by experienced individuals, and with great care. A moment of carelessness could result in a wrong command being issued or an incorrect option being used, with the potential of catastrophic damage to the installation and to the work of several users who use that machine.

The activities we have described above are the domain of a role called the system administrator. The person who performs these tasks is called the system administrator. Depending on the size of the organisation, the responsibilities of a system administrator can be discharged by one person or by a group dedicated to this work. These people need to have adequate experience and training or knowledge about the working of

Linux, and need to keep themselves updated with the latest happenings to keep their knowledge current.

In a larger organisation, the work of a system administrator can be divided into various specialities, with a different person or group looking after each such speciality. These can be:

- A Network Administrator to look after the LAN, Internet connectivity and network services such as e-mail.
- A Security group to ensure that the system is secure from hackers and other disreputable individuals.
- A Hardware support group to keep hardware in fine fettle and to ensure it works well with your Linux system.

Sometimes you can have a Systems Group with different individuals or sub-groups looking at these functions. The exact structure and division of responsibility will depend heavily on the kind of organisation, the number of users there and the nature of work performed. In some companies, all installations might be performed by a different group or might be part of the computer supplier's work itself. A software development group in a financial services company might have much more need for security and so that aspect might receive more emphasis in such an establishment.

Any installation will tend to find the need for performing certain tasks again and again, or there might be a need to do certain things for which there are no specific commands or utilities in Linux. The system administrator therefore has occasion to create and use shell scripts that will help her do all this easily. She should therefore be skilled at shell programming. Such locally developed programs are often kept in a directory such as /user/local/bin for general use.

So we see that the system administrator is responsible for installing and maintaining the Linux installation. The specific nature of duties will depend on the organisation, but there are a few tasks that are common. The system administrator will therefore need to have adequate skill and experience in those tasks. In addition, he will often be faced with problems that have not occurred before. He should therefore be good at problem solving and innovation. Some of the qualities and training that he needs to have are:

- Knowledge of the structure of Linux and its usage
- Tools and utilities with their various options
- Shell programming
- Diagnosing the causes of problems.

These are therefore the qualities you have to imbibe as you take on the task of system administration. We will now look at how to perform some of the tasks of the system administrator.

5.3 INSTALLING LINUX

One of the basic tasks that you need to perform first of all, before you can do anything else, is to install Linux on your system. There are many ways in which you can do this and we will look at the main methods here. As always, because of the limited amount of space that we have here, we will concentrate on the main points. For more details, you should refer to the documentation that Red Hat Linux makes available to you. The installation process can be broken down into the following steps:

- Choosing an installation method
- Choosing an installation class
- Pre-installation checks
- Actual installation
- Configuration and set up

In some cases you might find it expedient to perform an upgrade instead of a full blown installation. This is something you could decide while choosing the installation method. There are also different things to be taken care of when installing Linux on different kinds of computers. For large machines that are more powerful than the typical microcomputer, the vendor of the machine would provide the Linux installation and support mechanism. Here we are looking at installing Linux on an IBM compatible personal computer. If your PC is of another type, such as an Apple Macintosh or a laptop or an older IBM compatible, there could be additional intricacies involved that we will not be able to cover here. You would need to make sure that Red Hat Linux will work on that system.

So now let us assume that we have a computer on which Linux will work and begin the steps for performing the installation.

5.3.1 Choosing an Installation Method

You have to first decide how you are going to be performing the Linux installation, as there are many methods available. One straightforward way is to do a local installation using a Linux distribution purchased from Red Hat. In that case you get a CD-ROM that you can used to boot your machine with. The rest of the installation can then be continued using the Linux CD-ROMs provided. For simplicity, we are assuming here that your computer has an IDE CD-ROM drive and that there is no other operating system installed on it. This means that your computer is blank, with nothing else on it and your Linux installation is going to be the first time anything has been put on it. If you have some other hardware, then there will be an additional driver diskette that you will need to have and that you will have to insert at the appropriate time. You can boot the Linux installation program from a floppy also.

Some other ways of installing Linux are:

- Using an FTP server that has the Linux images on it
- Using an HTTP server that has the Linux images on it
- Using an NFS server that has the Linux images on it.

In all the above cases, you will need a network driver floppy diskette or a PCMCIA driver diskette.

5.3.2 Choosing an Installation Class

There are different classes or types of Linux installations that you can choose from. They are appropriate in different situations and you can decide the one to use based on the intended end use of the system. Remember that these types really determine what kind of software components are installed and there is no fundamental difference between them. They are really a short, predetermined set of components or packages. The classes are briefly described here.

Personal Desktop

This is the simplest installation that brings in the least number of software packages. It is useful if the machine is going to be used by an end user, such as at home, or for office productivity. It will also install the GUI desktop environment with the X-Window system. This is useful for new users who are just getting familiar with Linux and will not want to perform advanced tasks.

Such an installation needs 1.8 GB of free space. During the installation you can still choose to install additional packages that are not part of the default installation. So what you really put on the machine is still up to you. But it does save you from the tedium of choosing every single package to put in.

This installation will use up to twice your RAM as disk space for your swap partition. The root partition will hold all the system files and user files. The size of the boot partition will be 100 MB.

Workstation

A workstation installation is useful for professional software developers. It puts in a graphical user interface together with basic software development tools and more system administration utilities. It will use up to 2.2 GB of disk space. The other characteristics are the same as the Personal Desktop installation. Both of these can take up any other software packages that you want to install as you go through the process. Remember that if you choose to put in more packages, the disk space required will be correspondingly greater.

Server

If you are going to use your machine as a server then you would perhaps not need much configuration to be done on it. In such a case you can do a server installation without even putting in a graphical user interface. You would then need just 850 MB to 1.5 GB of space depending on how much functionality you plan to put in. Should you decide to put in the GUI, the space required would increase correspondingly. The other characteristics of the default server installation are the same as for the other classes.

Custom

A custom installation is the most flexible option as it does not have any predetermined packages that will be put in. You have to decide which packages you want as you proceed. This therefore requires an understanding of the packages that are available and their dependencies. It is meant for advanced users who know exactly what they want and need the freedom to decide. The disk space required varies from 475 MB to 5.0 GB, depending on what you choose to put in.

All the above classes of installations have the same partitions if you choose automatic partitioning. Also, the disk space requirements mentioned above assume you are installing only the English language version. Should you choose other languages as well, the space requirements will go up in each of the above cases.

Upgrade

It is also possible to install Linux on a system that already has some earlier version of the same operating system. The old Linux version should be 6.2 or later, because that is when the rpm method of package installation was introduced. If that is the case, you do not have to go through all the steps needed in a fresh installation. The advantage is that all the data that you have will be preserved in an upgrade. Your partitions will not be altered and only the appropriate software packages will be upgraded. So you do not have to go through the exercise of backing up your data and restoring it after the installation has been completed.

5.3.3 Pre-installation Checks

Before commencing your installation, it would be politic to garner information about your machine. You first need to see how much hard disk space you have. That could have an impact on the kind of installation you can perform. However, in these days of large hard disks of 80 GB and above, it is unlikely to be a constraint. Yet it is best to make sure.

When we refer to the disk space, it means unpartitioned disk space that is not currently being used by another operating system because every partition on the disk behaves like a separate disk drive. It is quite possible to install Linux on a machine that is currently running some other version of Linux or of an operating system like Microsoft Windows of some flavour, such as Windows 2000 or Windows 98.

You next need to find about more about the type and make of the hardware on your machine, as this is information you will have to provide as you go ahead with the installation process. For this, you will need to refer to the documentation provided by the manufacturer of your machine. If you have an assembled machine, refer to the documentation that came with the computer components that you put together. In many cases, your existing operating system can give you information about your hardware. For example, in Windows 2000 you can look at your Device Manager tab under the System icon in the Settings to find out about the hardware and the specific type and make. While this is no substitute for actually finding out about your hardware physically, it is usually sufficient for you to be able to perform the installation.

Here is a list of some of the hardware that you should get more information about, to be able to carry out your Linux installation without interruption.

- The hard disk – is it an IDE or a SCSI? What is its capacity and is there any volume label? What are the partitions you will want to create and what will be their sizes? For example, do you want to have `/tmp` or `/home` as separate partitions or will everything be in `/root`?
- Similarly you need to know about your CD-ROM drive. Is it an IDE or a SCSI type?
- Which is your hard disk controller? If using a SCSI adapter, who is the manufacturer and what is the model number?
- What kind of mouse do you have? Here you must know the mouse type, such as serial, bus or USB and the protocol, such as generic. Also note the number of buttons, which are commonly 2 or 3. Does it have a vertical or horizontal scroll wheel?
- What kind of monitor do you have? Here note the manufacturer and the model number for reference.
- What are the characteristics of your display adapter? This is especially important if you are going to install the Graphical User Interface. Remember the manufacturer and the model number as well as the amount of video RAM used.
- Find out about your sound card. For this you again need to know the manufacturer and the model number. In addition, note down the chipset used.
- Do not forget the amount of RAM in your computer.

Besides the above, you will need to have some more information for your computer to be able to access the network. First you have to choose a name for your system. This is a personal choice and has to be unique within a domain. Organisations have their own schemes for this. Some use the names of rivers, mountains, sages or places. Others leave it to the system owner while still others use mundane numbering schemes such as `marketing001`. You will have to follow the organisation's policy here. Make sure you know the organisation's Internet domain name. This is typically derived from the organisation's name but could be different. For example, for IGNOU, the domain name is `ignou.ac.in`. In larger companies, the Internet domain name is further subdivided and if so, you have to know the domain name for your subdivision in the organization. This could be something like `marketing.mycompany.com.`

Next, make note of your network interface card. Again, here you must know the manufacturer and the model number.

Find out the IP address allotted to your machine. This could be a static IP address decided by the organisation. If it is allocated dynamically by DHCP, you should be aware of the DHCP server address that does this allocation. In many cases, while the address is allocated by a DHCP server, it is permanent in that it does not change from one boot to the next. In addition, you need to know the network mask for your machine.

In case your machine will boot off a machine on the network, you need to supply that machine's IP address.

You also need to know the IP address of the default gateway for your network that connects it to the outside world.

Lastly, you need to note down the IP address of the DNS server that your machine will use to resolve hostnames to IP addresses and vice versa. There could be more than one nameserver, in which case you need to know the IP addresses of those machines as well.

Once you have all of the above information, you can proceed to the actual installation of Linux on your machine. You should see that all of this is noted down on a piece of paper for ready reference as you go ahead with the process.

5.3.4 Installation

We will now see how to install Linux on your machine using the Linux bootable CD-ROM. As we have already observed, there are several ways of installing Linux. Here we are assuming a simple scenario where you are installing Linux from a Red Hat distribution from a CD-ROM on a computer with IDE drives. There is no other operating system running on the computer, so we do not have to take into consideration the issues involved in creating a dual boot system.

As a beginner to installation, you would prefer to use the graphical user interface with a mouse for your work. Of course, the keyboard can also be used for working with the GUI. Later, when you become more experienced, you can look at using a command line interface and performing more intricate installations. These can include using text mode installation, booting from a different source, controlling memory usage, using kernel options and so on.

To begin loading Linux, insert your bootable CD-ROM into the drive and turn on the machine. Here you will need to make sure that your system is configured to boot from the CD-ROM. For this you might have to change your BIOS settings to alter the boot order. If this is taken care of, your machine will display a prompt

```
        boot:
```

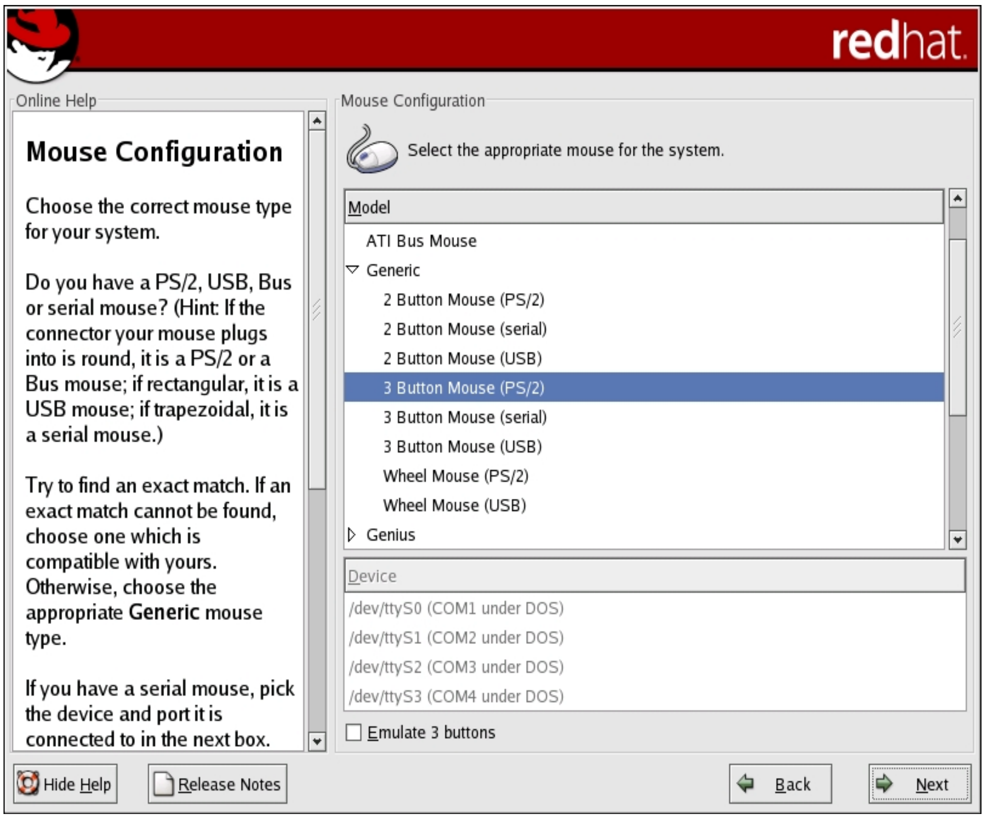
At this point you can either do nothing and the boot process will start automatically after some time. Alternatively, to begin right away, you can press the ENTER key. If you do this, you would be ignoring the various boot options that you are presented with by the program. That is just what you would probably want to do as you learn about the process of loading Linux.

Once the booting process starts, you should find that your hardware is automatically detected. If this does not happen, you will have to do the installation in expert mode. But here we assume that things go well and you are able to proceed. You will now get the boot loader screen, where you should select the option to install from CD-ROM. The system will first try to determine the type of your CD-ROM drive. If yours is an IDE type drive, the program should detect it. In case you have a SCSI drive, you will be asked to choose a SCSI driver. If your drive is an IDE but the program is not able to detect it, you will need to refer to the Linux installation documentation or seek expert help on the matter.

If all goes well, you will come to a screen that says Language Selection. Here you have to specify the language that you would like to use during the installation process. There is a wide choice here, that includes Chinese, Japanese, French, Italian and many more. Let us assume that you want to work with English, in which case you should select it and then click on the Next button to continue.

The next screen that you get prompts you to select your keyboard layout type that you want to use. Here you would probably want to use the U.S. English layout. This is the QWERTY keyboard that we are familiar with and that has the $ sign over the number 4 in the top row. So select this layout using the mouse. You can also change your keyboard layout after the installation is over by using the keyboard configuration tool.

This brings us to the next screen, which is where we specify the kind of mouse that we have. Here there are many choices, and unlike the previous two cases, it is not at once apparent what type you are likely to have. The research on your hardware that you



Mouse Configuration

Choose the correct mouse type for your system.

Do you have a PS/2, USB, Bus or serial mouse? (Hint: If the connector your mouse plugs into is round, it is a PS/2 or a Bus mouse; if rectangular, it is a USB mouse; if trapezoidal, it is a serial mouse.)

Try to find an exact match. If an exact match cannot be found, choose one which is compatible with yours. Otherwise, choose the appropriate **Generic** mouse type.

If you have a serial mouse, pick the device and port it is connected to in the next box.

Select the appropriate mouse for the system.

Model
ATI Bus Mouse
▽ Generic
2 Button Mouse (PS/2)
2 Button Mouse (serial)
2 Button Mouse (USB)
3 Button Mouse (PS/2)
3 Button Mouse (serial)
3 Button Mouse (USB)
Wheel Mouse (PS/2)
Wheel Mouse (USB)
▷ Genius

Device
/dev/ttyS0 (COM1 under DOS)
/dev/ttyS1 (COM2 under DOS)
/dev/ttyS2 (COM3 under DOS)
/dev/ttyS3 (COM4 under DOS)

☐ Emulate 3 buttons

Figure 1: The Mouse Selection Screen

would have done in the previous section is what will now come in useful. You need to choose from among the many mouse types supported. Since you already know the kind of interface your mouse uses (PS/2, AT, serial, USB) and the number of buttons that it has, you will be easily to make the appropriate selection. *Figure 1* shows this screen to make things clearer for you.

What does one do when one cannot find an entry that seems to match our mouse? You can try the Generic type. If your mouse has a scroll wheel, you can make the selection as applicable, by choosing USB or PS/2. Another selection that you have to make is whether to emulate three buttons when your mouse has only two. This can make it easier to use a graphical user interface. The emulation is done by considering the middle button to be substituted by both the left and right buttons. So if you press both of them simultaneously, it is equivalent to pressing the middle button.

After completing the installation, you can make your mouse left handed. This is more convenient for left handers as it corresponds to their natural hand. Here the functions of the left and right mouse buttons are interchanged. So the right button becomes the main button.

Finally, if your mouse is a serial mouse, you will have to specify the device or port (such as COM1 or COM2) that your mouse connects to. Now having answered all the questions related to your mouse, you can click on the Next button and proceed to the next screen.

The installation program can detect a previous version of Linux that is already present on your system. In such a case, you will now come to a special screen that will prompt you and ask whether you want to upgrade your existing installation. Should you choose to upgrade, all your existing data will remain safe. None of your partitions will be changed and only the required files will be altered and overwritten. This option will work for Linux versions 6.2 and later. You will have the option to customise the installation and to choose the packages you want upgraded.

However, it might be that you simply want to throw away your old installation and begin again from scratch. This time your data might or might not be preserved, depending on what kind of partitions you make in your fresh installation. If they are the same as the existing ones, your data will be safe. Should you choose a different partitioning scheme, then your data will be erased and will no longer be accessible after the process is over.

Once you have made your choice, you need to click on the “Next” button to move on to the next screen. This is where you choose your installation class or type. You already know about the differences between the Personal, Workstation, Server and Custom installations. If your machine will be used by developers in an office environment, the Workstation type might be best for you. After making your selection, go to the next part.

Now you come to an important screen where you have to partition your hard disks. This can be a daunting task for a beginner, but since you are a system administrator, it should be easy for you. Linux comes with a tool called Disk Druid to help you do this. Whenever, you perform an installation, you should be prepared for the possibility of losing all data on the disk. Therefore, it is important to back up all your important data before commencing the operation. In this discussion, we have not so far dwelt on the point because we assume you are installing on a new machine that does not have anything on it.

When using Disk Druid, you will have full control over the partitions. You will be able to choose the partition sizes and the types of the file systems you want to create on them. You will be able to decide where they will be mounted when Linux runs. However, if you do not want to go through this, you can ask the program to perform automatic partitioning. This does not give you much control but is a good option if you do not want to change anything or are doing your first Linux installation.

Even automatic partitioning will allow you to choose what kind of existing partitions are to be deleted from your disk. These choices are presented to you on the next screen. You can decide that you want to delete all old Linux partitions, or all existing partitions that might have other operating systems or file system types installed on them. Instead, you can also choose to preserve all existing partitions and use whatever free space is available on the hard disk. Since you might have more than one hard disk, the screen allows you to choose the one on which you want to create these partitions. All the partitioning choices will apply only to the disk that you have selected.

You can then review the partition selections that will apply and confirm or decide to modify them. Here you will be working with the Disk Druid utility, as you would have had you chosen to partition manually. At this point you will be faced with some detail, as you have to decide where you want Linux to be installed. You would have already made these choices while preparing for the installation.

The Disk Druid shows you the hard disk and its model number at the top of the screen. You also see the number of cylinders, heads and sectors that collectively describe the drive's geometry. You should make sure that this is in tune with what you already know about your hard disk. The tool then shows the device file corresponding to the device, the point in the directory hierarchy at which it will be mounted, the type of the file system it will contain, whether the partition will be formatted, its size and the start and end cylinders for the partition. This is shown for every partition.

Let us take a brief look at the different kinds of file systems that are available to you. A swap partition is used to increase the amount of virtual memory that your Linux system can use, as it supplements the RAM. Whenever required, pages are swapped from RAM to the swap partition on the disk and are loaded back when possible. Usually the size of the swap partition is made twice the RAM that you have. Linux has a file system called VFAT that is compatible with Microsoft's FAT file system and supports its long filenames.

The tool also allows you to create RAID partitions for redundancy that makes for more safety of your data, because if one partition develops an error, you can still access your data from the other. However, in this first shot at Linux installation, we will not venture into configuring RAID partitions. But this can be done by first creating two or more partitions that have a RAID file system type.

You can also create logical volumes using the tool. Here you can use one or more partitions on the same or different hard disks to work like one single logical partition or volume. This can allow you to have volumes that are larger than any single hard disk that you have. Again, in this introduction, we will not go into more detail about creating and maintaining logical volumes. You can create physical logical volumes by selecting that as your file system type.

The ext2 file system is the basic Unix file system and ext3 is the journalling version of ext2. Journalling allows you to quickly recover from system crashes. The default file system type in Linux is ext3, and is also the recommended choice.

Let us now look at the operations we can perform with the tool. You can add, remove or modify partitions. You also have a reset button that lets you restore the initial state of the disk, the situation that prevailed at the time you started the session. If you reset, the changes that you might have asked for in the session will be lost.

You can remove a partition from the disk. If you do so, you will be asked to confirm, as this has the potential to cause loss of data if it was an existing partition that was in use.

When you choose to add a partition, you have to provide information about it. The first thing is the mount point. To help you, there is a pull down menu that you can use to select the appropriate value here. For example, the boot partition should be mounted on `/boot` and the root partition on /. Then you need to enter the type of the file system that you want the partition to contain. The next selection is where you specify the hard disks that can contain the partition. If you do not select a disk, then your partition will not be created on that disk. You would usually let the tool decide where to place the partition.

You now need to enter the size of the partition. You can either enter a fixed size in MB, or choose to allow the partition grow from a base size to an upper limit as required. You can even choose to let the partition grow to fill up the entire available space on the disk. Next you can decide to let this be one of the primary, or first four, partitions on the hard disk. If not, it would become a logical partition. The last option while adding a partition is specifying whether you want to check for bad blocks while formatting. While it is a good thing to do, checking for bad blocks can take a fair amount of time, especially with the large capacity hard disks of today.

Similarly, when you select a partition for editing, you can change its attributes such as the file system type or size. If the partition information has been already written to the disk, you can only change the mount point of the partition. Should you want to change other attributes, the only way is to remove the partition and create it again with the new attributes.

Once you have done partitioning your hard disks, you need to move ahead to the next installation screen that lets you install your boot loader. A boot loader is needed so that it can load Linux. Whenever the computer is started up, its BIOS loads the program that is installed on the Master Boot Record (MBR) of the hard disk. This program then loads the operating system. You can also install the boot loader on the first sector of your boot partition, in which case the boot loader on the MBR will need to be asked to start up your boot loader. So if your disk has only Linux as the operating system, you should place your boot loader on the MBR.

Linux has two boot loaders called Grub and Lilo. Grub is powerful and can load Linux as well as other operating systems. Lilo is also a good boot loader and can boot Linux from several different sources. You will also have to choose which operating system to boot from by default, in case you have another already installed on your hard disk. Then when you boot your machine, you will have to choose the other manually if it is not the default.

You need to remember that if you do not install a boot loader, you will be able to boot Linux only through a boot diskette. So you should install one unless you have some special reason for not doing so. This could be because you already have your favourite boot loader, such as a third party commercial offering like Partition Magic, installed on the hard disk. After the boot loader installation screen, the next screen you reach will depend on whether you have a network interface card on your computer. If so, you will reach the network screen. In an organisational set up, this is the most likely situation. So let us now look at the configuration to be done here.

If you were installing off a network, then you would have already used a network driver diskette to be able to work. But if you started installing using, say, a CD-ROM, then you can now configure your network cards. You have to select from the devices that are detected during installation and decide whether they should be activated automatically at boot time. For each device, you need to edit its properties, namely, its IP address and netmask. If these will be supplied by a DHCP server, you can specify this here. These values will depend on what is used in your organisation.

You can now enter the hostname and let it be detected automatically by a DHCP server. These decisions you would have already made as part of your pre-installation preparation. Now you also need to put in the IP addresses of the gateway for your network that connect it to the outside world and the IP addresses of up to 3 DNS servers for it. If you are using DHCP, these will be provided automatically and you need to enter this information only if you are providing the IP address of the machine manually.

It is quite all right to provide this information manually for one network interface card and to set these values for another card through DHCP. Once you are done with the network configuration you can select the Next button to move on to the next screen where you will set up your firewall.

In these days of networks, your computer is vulnerable to all manner of attacks by persons so inclined. A firewall offers protection against intruders over the network. Well configured, it reduces the chances of an attack greatly. You can choose from three levels of predetermined security policies. The first is the option of not using a firewall at all. While it might seem strange after what we have just said, this can be a good choice if you are sure your machine is going to be on a trusted network, such as a corporate network protected by other firewalls. Also, you might be planning to perform a more elaborate firewall configuration later.

The next level of security that you can choose is medium. This disables access from the outside to reserved ports, that is, up to port 1023. So services like HTTP, FTP and so on will not work. Also barred are NFS servers and clients and X-Window display access to remote machines. The X font server is also disallowed.

The highest level of security bars all but DNS and DHCP. If the pre-configured firewall rules seem too restrictive or are not what you want, you can always perform additional customization. One thing that is possible is to declare some devices as trusted, whereupon no firewall checking will be done for them. You could do this on a multi-homed machine, where some of the networks are trusted. So if you are on the Internet through one interface and are connected to your company network through another, you can decide to declare the company network as trusted. You can also choose to allow access to additional ports beyond those allowed by your security level. Some of these are listed as service options that you can select through a checkbox, for instance mail and FTP. Besides, you can allow any arbitrary ports by using the notation

```
      port: protocol
```

such as `1111:tcp.`

Having secured your system, you can select the Next button to move on to the succeeding screen where you can set your language options. Linux allows you to work in multiple natural languages. As you might imagine, you have to choose at least one language to work with. If you choose only one language, it is the default as well. If you choose more than one language, you need to choose one language that will be your default.

The language that you chose to use for installation at the beginning of the installation process might not be the same as the language you choose to install for use thereafter. However, that is the language selected by default. If they are different, then after the installation is over, you will be able to use only that language. For example, if you use English (USA) as the language for installation, and select French (France) as the language in this screen, then your installation will continue in English. Once the installation is over, you will be able to use only French.

You should choose all the languages that you might want to use during normal working. But it might be better to avoid choosing languages that you are not likely to use, because of the extra disk space that will be taken up by putting them in.

You are now at the tail end of the installation process. The next item to be configured is the time zone. Here you can choose to set your system clock to Universal Time Coordinated (UTC) that is based on the 0^{O} longitude that passes through Greenwich. The time that you will then see will be based on UTC and the time zone that you enter.

To select the time zone, you have two methods. You can enter the offset from UTC for your location. So for India choose the option for +5:30, as Indian Standard Time is 5 hours 30 minutes ahead of UTC. For countries that use daylight saving time, you can set that too.

The second method is to set the time zone interactively, based on your location. You see a map of the world with many important cities marked out with yellow dots. The place you select will be shown with a red X.

Once the time zone has been set, it is time to set up your root password. This is an important password, as the root or superuser has complete control over the system. Restrictions and permission settings that apply to ordinary user accounts are no bar for the root user. So this password must be set with care. Make sure that it is not easy to guess. You have to enter the password twice and both the entries must match for the installation to move forward. If you are going to be administering this system, you must preserve the password carefully. But even then, do not use this account for ordinary work, because a small mistake as root could damage your installation. Use it only for administrative work on the machine.

The next screen is for setting up authentication information. You need not really do anything here unless you are going to set up network passwords. But you could choose the "Enable shadow passwords" option. This keeps your password more secure because the actual password is stored in the file `/etc/shadow` instead of `/etc/passwd`. The shadow file is readable only by the superuser, unlike the `passwd` file that is readable by all.

After this you come to another important screen where you configure the packages to be installed on your machine. This is where choosing an installation class is useful. Depending on what has been chosen, a list of package groups appears. You have the option of accepting the recommended packages for installation or of deciding on them yourself. Each group of packages can have optional packages that can be installed depending on your preference, besides base packages that are installed with that group by default.

When you choose to customise the packages to be installed, you have to select the checkbox next to the package group. Then you can click on the details link to obtain a list of all packages that constitute the group. Here you can select the individual packages that you want to have. After selecting the package groups, you can see all the packages that will be installed on your machine. This can be done in an alphabetical listing or as a tree structure under each group. Certain required packages needed to run Linux cannot be selected or deselected as they do not appear in the package listings. They are always installed. Information about a package can be had by clicking on it.

What you need to be aware of is that some packages might be dependent on others in order to work properly. You need not worry about what these dependencies are as the install program finds that out automatically for you. After you have completed your package selection, you get a list of such unresolved dependencies. You are then given the option of resolving these dependencies by including the dependent packages, ignoring the dependencies or simply discarding the packages that have the dependencies. Remember that if you choose to ignore the dependencies, the packages with the dependencies might not work as expected. Of course, if there are no unresolved dependencies to start with, you will not see this screen.

You can see the amount of disk space that will be required by your selection, so if there are constraints you can adjust your package selection. However, in these days of large disks, it is not likely to be a problem.

Well, that has been a lot of choosing and decision making! Now it is time to sit back and let the machine do some work for a change. When you go to the next screen, you are at a point where you have to make one final decision about whether to go ahead with the installation as you have chosen. This is the last chance you have to abort safely, because if you go ahead here, the partition information will be written to disk and the installation will begin. Even so, to abort the installation, you need to reboot the computer.

Once you click on the Next button here, the install program will start installing Linux and all the selected packages according to the options that you have provided. There is nothing for you to do but to wait and watch the installation proceed. To relieve the monotony and to reassure you that progress is indeed being made, you can see a bar and other screen messages that give the latest situation regarding the install. As you can imagine, the time taken here will depend on how fast your computer is and what packages you have chosen for installation.

Presently your installation will be completed. You will then be asked whether you want to create a boot diskette. You should create one, because it will enable you to boot should anything happen to your boot loader. You can create the diskette even after the installation is over and you have started working on it. For that matter, you can perform most of the steps in the installation such as configuring devices, language selection, package selection, configuration of services, firewalling and so on at any time even after the initial installation is over. The only requirement is that you know the root password as only the superuser can perform such maintenance.

If you want, you can now configure the X-Window system server that is needed to get your graphical user interface. You have to choose the video card that your machine has from the list provided. If you know your card characteristics well, you can select the unlisted card and configure it, but in general, if your card does not appear in the program's list, it is not supported by the X-Window system. After selecting the card, you have to specify the amount of memory it has. Try to give the correct value and consult the card documentation if needed. While specifying more memory will not harm anything, it can mean that you have trouble with your X server.

The next screen lets you configure your monitor. You will be presented with a list of monitors from which you should choose your display, or the one closest to it. If your monitor does not appear on the list, choose an appropriate Generic monitor. Here you have to be careful that you do not select a monitor that has capabilities beyond yours. Your monitor can be damaged permanently if the selected monitor has horizontal or vertical synchronization frequencies beyond those of your display. Your monitor could get overclocked and be destroyed in such a case.

Now decide your colour depth and display resolution. You can also decide whether after booting you want to land up in graphics or text mode. In the latter case you will be presented with a command prompt from which you can issue any Linux commands that you wish. It might be better to set up the machine to get into graphics mode directly. Finally you can see your efforts bearing fruit. When you select the Next button on this screen, you will see the long awaited message that tells you the installation is complete. You will now be asked to reboot the machine. You should remove your CD-ROM or diskette from which you had booted the machine for installation, otherwise the computer will boot again from that device and not from your hard disk.

5.4 BOOTING THE SYSTEM

Now you are ready to boot up the machine from your hard disk, using your freshly installed Linux. You can reset the computer or power it off and on again. After the machine's preliminary power on checks are over, you will come to a screen where you have the choice of booting the default operating system. Let us assume here that you have configured Linux to be your default system.

If you have another operating system such as Microsoft Windows 2000 loaded on your machine, you would have the choice of selecting it to boot from. If you choose to take no action, the default operating system will be booted up anyway after the timeout period has elapsed.

Once Linux starts to boot, you will see several checks being performed, at the end of which you are presented with the prompt to login. This assumes you have not chosen to start up the graphical user interface, in which case you see a graphical login screen rather than a text based terminal.

When you boot up Linux for the first time there is some configuration you might like to do, such as setting the system date. You will be presented with a Setup Agent that will lead you on through such tasks.

Booting is so called because the job involves something akin to lifting oneself up by one's bootstraps, that is, from a system which is off you need to have a system running a complex operating system like Linux. This task is done in stages, with a very simple program in the computer's ROM that runs and loads a boot loader program on say, the hard disk, which is then able to load the whole of the Linux kernel.

Once the Linux kernel is loaded, it brings the system state to the run level that is specified in the file `/etc/inittab`. In this state you have the system in multiuser mode with NFS services available, if configured. This is the default state for a Linux system. If you want to perform some maintenance on the machine you would do it in single user mode that is run level 1. In this state only the console is available for logging in and other users cannot come in.

The tasks to be done when entering a particular run level are determined through a file called `/etc/rc.d/rc`*`n`*`.d`, where *n* is the run level. This gives a list of scripts that are run, and consists chiefly of services that are started up.

Together with booting up, you need to know how to shut down the system safely. When in operation, the file system is buffered in RAM, meaning that it is stored in the RAM and is then flushed to the disk from time to time. Thus the file system on the hard disk is not always in synchronization with that in RAM. So abruptly powering off the computer can badly damage the file system, sometimes beyond repair. To power off a computer running Linux, use the shutdown command, which really is a user friendly front end to the init command.

You can look at the manual entry for shutdown to see all of its many options. There are only two that you are likely to use frequently. To initiate a shutdown immediately, say:

```
[root@linux root]# shutdown -h now
```

You can also give an absolute time in the form hh:mm, such as `23:45` to shutdown the computer at that time. More likely you want to specify the amount of time to wait by specifying `+m` for the number of minutes you want to give as the grace period. The keyword now is equivalent to `+0`.

Sometimes you want to not just shutdown but to reboot as well. For this you can use the `-r` option to the command.

```
[root@linux root]# shutdown -r now
```

This command will perform all the necessary work to shutdown your machine in an orderly manner.

☞ **Check Your Progress 1**

1) List some of the duties of a system administrator.

   ....

2) Turn off the computer abruptly when you and other are working on it. See how a file system consistency check is performed. Is it possible that some data gets lost.

   ....

3) Find how to force a file system consistency check.

   ....

5.5 MAINTAINING USER ACCOUNTS

You have seen in the second unit how users can login into the system and how passwords are used to prevent unauthorised access. You also saw how one or more users could be associated with a particular group. When you studied the long listing provided by the `–l` option to the `ls` command, you saw how the user name and group were also given for each file in the listing.

You also know that an ordinary user cannot help you if you have forgotten your password, and that only the superuser can set somebody's password without already knowing what it is. One of the duties of the system administrator is in fact maintaining user accounts. This activity encompasses creating new accounts, deleting accounts of users who are no longer permitted to access the site and helping people who might have forgotten their passwords. On many large installations the system administrator will run regular checks on user passwords to make sure that they cannot be guessed easily.

The Linux utility for setting passwords does perform some checks for the quality of the password, but do not rely on them completely. Some of the messages it can give are:

BAD PASSWORD: it is based on your username

BAD PASSWORD: it is too short

BAD PASSWORD: it does not contain enough DIFFERENT characters

BAD PASSWORD: it is too simplistic/systematic

These give you an idea of the kind of checks that are done, but the system administrator can also help here by setting a minimum and maximum time between permitted password changes. The check for the minimum time is based on the premise that an intruder will want to change the password as soon as he gains access. The user can also be given a warning some time before the password is due to expire and passwords not changed for a given time even after they expire can be taken as inactive. You should choose passwords that are not easy to guess and should change them periodically.

With all this background, let us see just how user accounts are maintained. This information is kept in a file called /etc/passwd. This file looks like this:

```
[root@linux root]# cat /etc/passwd

root:x:0:0:root:/root:/bin/bash

bin:x:1:1:bin:/bin:/sbin/nologin

daemon:x:2:2:daemon:/sbin:/sbin/nologin

...

kumarr:x:500:500::/home/kumarr:/bin/bash

...
```

You should have been able to decipher some of this file. It contains a one line entry for every account. The different fields in this line are separated by a colon (:). The first field is the login name of the user. This is the name by which the user will be known on the system. The second field simply contains an `x` when you are using a shadow password file. Because the file is readable by everybody, placing the password, even though encrypted, in the shadow file gives you added security. The /`etc/shadow` file is readable only by root. So an intruder will not be able to easily read even the encrypted password. Look at the shadow file entries for the accounts shown above:

```
[root@linux root]# cat /etc/shadow

root:$1$oFtzPwDL$sw6qnXSqCTZeo5doxfpis0:12554:0:99999:7:::

bin:*:12554:0:99999:7:::

daemon:*:12554:0:99999:7:::

...

kumarr:$1$8yVJnCdk$HtTL8AIRNrX9hLj6dq0Ir.:12704:0:99999:7:::

...
```

The first field is the user's login name followed by the encrypted password. It can be up to 34 characters long depending on the algorithm used. A * in the password field means a user cannot login using that name because it will not match any password. But the same effect can be achieved by locking the account using the -l option to the passwd command. This does not apply to the superuser because in his case the password check is not performed at all.

Coming back to the `passwd` file, the third field is the user identification number, or user id or uid. The linux operating system works in terms of the user id rather than the name for most purposes. Any user with an id of 0 is a superuser, and ids below 100 are usually reserved for the use of Linux. Ordinary users get ids from 100 onwards. Now we will discover an interesting fact. Suppose you login as an ordinary user, kumarr, who has a user id of 500. Create a file /home/kumarr/checking, logout and login as root, and look its long listing.

```
[root@linux root]# cd /home/kumarr

[root@linux root]# ls -l checking
```

```
-rw-rw-r—    1 kumarr   kumarr         665 Nov 29 21:13
checking
```

Now edit the `passwd` file carefully and change `kumarr` in the first field to smith. Look at the listing of the file `checking` again.

[root@linux root]# `ls -l checking`

```
-rw-rw-r—    1 smith   kumarr         665 Nov 29 21:13
checking
```

You will see that `ls` now reports smith as the owner. This shows that the `ls` command uses the passwd file to translate the user id to the user name. If you delete the account of `kumarr` by removing his entry in the `passwd` file, you will see

[root@linux root]# `ls -l checking`

```
-rw-rw-r—    1 500   500         665 Nov 29 21:13
checking
```

This is because now `ls` is not able to find out who the owner is and therefore reports the user id instead. Do put back `kumarr's` account under his own name now!

The fourth field is the group identification number or group `id or gid`. The file `/etc/group` contains the group names corresponding to the group ids, and also contains information on which users belong to which group. This file is also used by the `ls` command to translate group ids to group names, just as it does with user names. The fifth field is a comment field containing up to 30 characters. The sixth field gives the home directory of the account and thus determines where `he` will reach when `he` logs in. The last field gives the shell he will be presented with. If the command there is not executable, the user will not be able to log in. The default shell is `/bin/sh`, used when the field is empty.

Managing User Accounts

You now need to know as a system administrator how to add new user accounts, remove or inactivate them and change their characteristics. This is quite simply done in Linux. You need to go to the start button which is the red hat icon with an upward pointing arrow, usually to the left of your tray. Then choose the appropriate options as shown here.

```
Start —> System Settings —> Users and Groups
```

If you are not the superuser, you will be prompted to enter the superuser password. You then reach the Red Hat User Manager screen. Here you will see a list of user accounts. This does not include system related accounts. You have buttons to Add an account, at which you will have to enter the relevant information for the user such as his group and initial password. You also have an option to manage groups, where you can change the groups to which a user belongs.

To change the attributes of an account, you need to select it and click the Properties button. This brings up a User Properties screen where you have four tabs. Using these you can change all information about the user, including setting his password and changing the groups he belongs to.

You can also delete an account if you need to, or you can just lock it and disable access to the account by any ordinary user.

5.6 FILE SYSTEMS AND SPECIAL FILES

So far we have looked at two kinds of files, directories and ordinary files. Linux looks at everything as a file. So all hardware devices like terminals, tape drives, floppy drives, CD-ROM or DVD-ROM drives, printers and scanners are considered to be files in Linux. What this means is that there is a filename associated with each device and you can read from or write to these files just as you would to an ordinary file. There is a device driver in the kernel for every device supported and the name of the file for the device points to it within the system. Thus when you perform an operation on such a device special file the device driver takes over.

There are two kinds of special files, character and block special, which are associated with character oriented devices and block oriented devices respectively. A tape is a character oriented device while a hard disk can be both character or block oriented. Thus you will find that both kinds of special files exist for your hard disk on the system, but that there are only character special files for any tape drives. Actually there are also pipe special files but we will not consider them here.

The bridge between the physical device name (the filename) and the driver for the device is located in the `/dev` directory. Let us look at a long listing of this directory.

```
[root@linux root]# cd /dev

[root@linux root]# ls -l

total 228

crw———      1 root    root    10,  10 Jan 30  2003    adbmouse
crw-r—r—    1 root    root    10, 175 Jan 30  2003    agpgart
crw———      1 root    root    10,   4 Jan 30  2003    amigamouse
crw———      1 root    root    10,   7 Jan 30  2003    amigamouse1
crw———      1 milind  root    10, 134 Jan 30  2003    apm_bios
drwxr-xr-x  2 root    root    4096 May 16  2004       ataraid
crw———      1 root    root    10,   5 Jan 30  2003    atarimouse
crw———      1 root    root    10,   3 Jan 30  2003    atibm
crw———      1 root    root    10,   3 Jan 30  2003    atimouse
crw———      1 milind  root    14,   4 Jan 30  2003    audio
crw———      1 milind  root    14,  20 Jan 30  2003    audio1
```

and many more such lines. Your listing might look somewhat different in terms of the devices and other fields here. But let us look at some of the main features of this listing.

The first thing you must have noticed is that the first character of the permission modes is no longer a hyphen (-), but is `c` or `b`. This indicates whether the device file is character or block special. The other permission modes have their usual meanings, except that you cannot execute a device no matter how disgusting its behaviour might be! So execute permission has no significance here.

Instead of the file size, you find two numbers separated by a comma. These are called the major and minor device numbers. A major number stands for a class of device like a terminal or a mouse, while the minor number gives the number of that kind of device. Thus different terminals might have minor device numbers 0 (for tty0), 1 (for tty1) and so on.

An entry for a device can be added by the `mknod` command. Suppose your terminal major device number is 4 and you have 32 entries in the /dev directory from 0 to 31. You will not be able to activate another terminal even if the driver can support it unless the device entry is made

```
[root@linux root]# cd /dev

[root@linux root]# /bin/mknod tty 32 c 4 32
```

The arguments to the command are the device name by which it will be known, the type (`c` for character and `b` for block) followed by the major and minor numbers. A device entry can be removed as usual with the rm command.

The `/dev` directory in Linux is itself organised into several directories like `/dev/raw` for the hard disk as a raw device, `/dev/ida` for the hard disk as a block device and so on. This is for convenience because of the large number of devices that are supported. You must remember that merely making a device entry is not enough. The Linux kernel must have support for that device. If you purchase a new device like a DVD-ROM it will come with a driver that you can install on your system. This will put the device driver for your device in the kernel so that you are able to use the device. The device entry is thus a link to the device driver for the device.

We will now look at hard disks and see how they are utilised in a Linux system. A file system is the complete hierarchy of directories and files starting from its root. A small device like a floppy diskette cannot hold many files but a large (80 GB) hard disk can easily hold several file systems. It is common to partition a large physical hard disk into several logical disks, on each of which a file system can then be created. This is done by the mkfs command. You have already seen about partitioning disks while installing Linux on your machine.

When you create a file system on a partition, you should use up all the space in the partition because otherwise the extra space is simply wasted. The device is named as a character device before the file system is made. When it has a file system on it, it is used as a block device. Although you could still read a disk with a file system as a character device, it would be very difficult for you to interpret the data stored on it unless you know intimately the structure imposed by the file system. Creating a file system is naturally done only in system maintenance mode.

What happens when a file system is created on a disk? The disk could previously be accessed only as a character device, but now a structure is imposed on this. You can now take advantage of the fact that you can access blocks on the disk randomly without having to go through all the previous blocks. So you can now look at the disk quickly. The file system structure consists of the boot block, the super block, the inodes and the data blocks.

The zeroeth block of every file system is reserved for storing booting information. It does not have any significance as far as the file system itself is concerned. It is the first block, called the super block, which contains information about the file system. Some of these items are the size of the file system, its name, the number of blocks kept apart for inodes, the list of inodes and the list of free blocks.

The index node or the inode blocks vary in number depending on the size of the file system and can actually be specified by the user while creating the file system. This number is the same as given in the super block. There is one inode for every file or directory in the file system and it contains information about the file such as the number of links to the file, the permission modes and the block numbers occupied by the file. The first 10 such numbers are called direct blocks because they directly hold information about the data blocks. The eleventh number holds a block address and that block holds the addresses of the actual blocks. There are more levels of indirection available so that you can store very large files in your file system. You cannot have more files and directories in the file system than the number of inodes. The remaining blocks in the file system contain the actual data in the files.

When the computer is booted, these file systems are not immediately accessible to users. This is because each file system has to be attached to the main file system. To do this you have to use the mount command.

```
[root@linux root]# /bin/mount /dev/hda4 /mnt
```

The first argument to the command is the device file associated with that file system. The second argument is the name of a directory. The mount command associates the logical device containing the file system with the directory. Now `/mnt` becomes the root of the directory hierarchy on the device. So if you had a directory called `khanz` on `/dev/hda4`, you can now access it as `/mnt/khanz`. The directory /mnt should preferably be empty before you mount some file system onto it, because while the file system is mounted you cannot access anything that was there on `/mnt` previously. To break the link between the file system and the directory say

```
[root@linux root]# /bin/umount /dev/hda4
```

`or simply`

```
[root@linux root]# /bin/umount /mnt
```

Now you can no longer get at the files in that file system. The various file systems are usually mounted automatically when the system reaches the multiuser state through commands in the `/etc/rc` shell script. If you issue the `mount` command by itself, it will list all the file systems currently mounted and the directories to which they are attached.

If somebody is working on a file system, it would be disastrous to unmount it because the file system would then become inconsistent when the device suddenly vanished. That is why you cannot unmount a file system unless no user or program is accessing it. The `umount` command will tell you that the device is busy and will not take any action.

In the booting process, a special root file system is mounted on the root directory (/). The major system directories like `bin`, etc and `dev` are under this directory. Since this file system is always in use, you cannot unmount or mount it yourself. All the other file systems that you mount are below / in the hierarchy, as you know. Sometimes `/tmp` is also mounted as a separate file system.

You have seen the intricate structure of the file system. This structure is subject to disruption due to many reasons. For example a block on the free list might also appear attached to a file, or a block might appear on neither. An inode might appear duplicated or a file might seem to be under no directory. Such situations potentially mean the loss of data. Fortunately the file system contains a fair amount of redundant information, and this enables a program to check whether it is consistent. If it is not, it is usually possible to repair the damage. The Linux system comes with a program called `/sbin/fsck` that performs a file system consistency check.

This program is run before mounting the various file systems because if the system is inconsistent to begin with it will get worse as users work on it. The `fsck` program looks for file systems to check in a file called `/etc/fstab`. Also, the fsck command is actually a front end to file system specific consistency check programs.

The file system check is not done every time the system reboots and is performed only if the system was shutdown abruptly leaving the file system in an inconsistent state. However after a certain number of reboots the check is forced anyway. The `ext3` file system that is the default in Linux allows journalling, which means that the check does not depend on the size of the file system but only on the size of the journal.

Every file system should contain a directory called `lost+found`, and this should be the first operation on it when the file system is made. Whenever `fsck` finds a file that is not linked to any directory, it places it in this directory. The program works in several phases and each phase performs some different check. You can set options to it that put it in interactive mode, where it expects a user response before taking any corrective action on a file system it is trying to repair.

☞ **Check Your Progress 2**

1) Remove the device file for a terminal and see what happens. Also try renaming the file and see the consequences.

   ....

2) Add a blank line to the beginning of the password file. What happens? What if there is a blank line in the middle?

   ....

3) Try umounting a file system when users are working on it. What happens and why?

   ....

5.7 BACKUPS AND RESTORATION

At a large installation, users do not usually have their own backups and the system administrator is responsible for the integrity of the system. On smaller installations and certainly on your own personal computer, you will be responsible for your own data and program files. So it is useful for every user to be able to backup data and know how to restore it if needed. As a computer professional you do not need to be told how important backups are.

There can be many different backup strategies and tools that you can use. We can classify backups into full backups, incremental backups and differential backups. A full backup is a complete backup of an installation or a part of its file system. If there is a complete crash, you can restore user data from this. But the operating system configuration files are also something that one produces over a period of time and with considerable effort. So you can consider a backup scheme for them as well. How often you should take backups depends on how much you are prepared to lose.

A full backup is easy to restore from, but given the large disk capacities of today, the amount of time taken to make a full backup can be non-trivial. So you can think of a full backup at periodic intervals with daily or more frequent incremental backups, where only files that have changed since the last backup are copied. But it can be difficult to locate a needed file in an incremental backup, so the concept of differential backups was introduced. Here you backup all files that have changed since the last full backup. So the amount of backing up required is in between that of a full backup and an incremental backup.

It is important to consider what the backup media should be as well. Very often it is a tape of some kind, but a USB disk or a network backup to a data centre can also be considered. You have to look at the reliability of a backup and make sure that you do not use media beyond their useful life. A backup is pointless if you cannot restore from it when you need to. So make sure to test your backups periodically. You cannot afford a single failure in a backup. You can also consider taking multiple backups each time.

Depending on the criticality of your business, you might keep different copies of your backup at different locations. But whatever your company characteristics, you have to make sure you have at least one backup offsite that is geographically sufficiently distant from your site.

There are several different kinds of backup tools. Some are sophisticated commercial offerings while at the other end you have the basic Linux commands such as tar and cpio. Both of them work quite well and you can decide what you will use.

The tar command allows you to take a backup of all or selected files in a directory hierarchy onto tape, floppy disk or the hard disk itself. Tar knows about directories and links, and maintains headers, checksums and file permissions and owners. To take a backup of all files under `/home/khanz`

```
[khanz@linux khanz]$ tar cvbf 40/dev/rmt0/home/khanz
```

Let us look at the meanings of the various letters after the `tar` command. The c stands for create, and it causes tar to create the backup. Remember that anything previously present on the backup medium gets erased thereby (unless it is a file on the hard disk). The v is the verbose option of `tar`, and makes it chatter about what it is backing up. The `b` is the blocking factor and defaults to 20. You then specify the file or backup medium where the backup is to be taken followed by the directories to backup at the end. Everything under the directories you mention is backed up. You can also specify individual files if you want to.

To look at the contents of a `tar file`, you can use the `t` option.

```
[khanz@linux khanz]$ tar tvf /dev/rmt0
```

For extracting, you use the `x` option. One thing you should be careful about is the use of absolute pathnames during the backing up. When you restore from such a backup, it will restore to that same pathname. So it might be better to use relative pathnames while creating a backup archive as you have the liberty of placing it wherever you want while restoring.

You can also use the z option to compress the archive, thereby saving a fair amount of space. The extent of the saving will depend on the kind of files that are being backed up.

Another useful command for creating and restoring from archives is `cpio.` It reads the list of files from the standard output and copies them to wherever you specify. For generating the list of filenames, you can use a program like find.

```
[khanz@linux khanz]$ find/home/khanz/|cpio-o>/dev/rmt0
```

You can use the many options of the find command to choose files that satisfy specific conditions so that only those files are backed up.

☞ **Check Your Progress 3**

1) You have backed up a directory containing several files onto 12 floppies using tar. When you try to restore these after a crash, the 4th floppy is found to be corrupted. How much data do you lose?

   ....

2) Why is it useful to take differential backups?

   ....

5.8 SUMMARY

This has again been a long unit wherein we have tried to cover a lot of ground in a very small amount of space. You saw what the responsibilities of a system administrator were and how the exact composition of such a group depends on the organization characteristics. The major topic of this chapter was Linux installation and you saw how to perform a simple Linux installation on a machine. You also were introduced to file systems and how the system was booted, with some idea of run levels. As a system administrator, you would need to maintain user accounts and you saw how to add, delete or change user account information including their passwords. Finally we briefly looked at the important topic of taking backups and restoring from them using the `tar` command.

This block would now have given you a fair idea of the Linux operating system and its capabilities. It should have prepared you to read up the large amount of documentation available on the subject and to explore the various resources devoted to Linux on the Internet.

5.9 SOLUTIONS/ANSWERS

Check Your Progress 1

1) Try yourself

2) Try yourself

3) Try yourself

Check Your Progress 2

1) Try yourself

2) Try yourself

3) Try yourself

Check Your Progress 3

1) You will lose data from the 4th floppy onwards.

2) These allow us to save space that would be wasted by taking a complete backup every time. At the same time you do not have to puzzle over which incremental backup has a copy of the files you want to restore.

5.10 FURTHER READING

There are a host of resources available for further reading on the subject of Red Hat Linux version 9.0.

1) http://www.redhat.com/docs/manuals/linux
2) http://www.linux.org gives among other information, a list of good books on Red Hat Linux.
3) Consider joining a good linux mailing list.

SOCIAS-IGNOU/P.O.10.5T/MAY, 2004