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**

) 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,