Computer Science

Higher Secondary - First Year
Volume - II   Practice

TAMILNADU
TEXTBOOK CORPORATION

Computer Science

PLUS I

Volume 2 : Practice

Government of Tamilnadu

FOREWORD

Computers are best known to their users through the applications. The important concepts that make computers work are presented in volume 1. This volume introduces the basic skills required to develop applications using computers. Today, computers are being used in almost every aspect related to the living of individuals. Space, Medicine, Education, Transportation are some of the well known application areas of computers.

There are many other applications such as accounting, banking, insurance that are providing valuable services to the users. To help the users overcome the difficulties in working on the hardware directly, operating systems were devised. This software makes it easy for the users to take advantage of the available hardware. It also helps in the faster development of the applications.

There are many operating systems. This volume trains the students on Windows XP and Linux. The students must understand that every operating system has its strengths and weaknesses. Knowing the two popular operating systems will make the students better users of the computers.

Operating system helps in the easy development of applications. The applications themselves are programs written in a specific language called 'Programming Language'. There are many programming languages with their own advantages and disadvantages. The students learn the C programming language in this volume.

A programming language can never be learnt by reading a book. It is learnt by solving a number of problems using the computer. It is only the intense practice that helps the users in developing good applications.

With the internet becoming a valuable source of information, it is important to know the ways of creating information for the World Wide Web. HTML is discussed in this volume as a simple and effective means of creating information on the WWW. The students must come out of the downloading habit and upload good content for the benefit of others.

The authors, reviewers and editors of this volume have taken great care in ensuring the accuracy of the contents. The presentation is lucid with many illustrations.

I wish the budding computer scientists a fruitful experience with the powerful tool called computer for the rest of their careers.

(E BALAGURUSAMY)
Vice Chancellor, Anna University, Chennai
Chairman Syllabus Committee

CONTENTS

CHAPTER 1

AN INTRODUCTION TO WINDOWS XP

1.1   What is Windows XP

Windows XP Professional is a user-friendly operating system designed for popular use. The most important advantage of using Windows is its GUI (pronounced as GOOYEE). It is said that the right side brain is good in processing the pictures and is the seat of creative thinking and intuitive ideas whereas the left side brain is good at logical thinking. It is belived , before the introduction of GUI, users of OS, mainly used their left side brain, keeping their right side brain idle. It is felt, Windows effectively uses the left and right side of the brain. Many other operating systems (including MS-DOS) use Command Line Interface (Interface lets any one connected with the machine. Actualy interface is a (virtual) connection between two entities. For example , T.V remote is an interface which connects a user and a T.V). In this kind of interface, you have to remember cryptic commands and type them without mistakes. To make things worse some operating systems are case-sensitive also (LS , Ls, lS or ls are not same). A simple spelling mistake or missed space will result in an error. Windows displays all the information on the screen and all you have to do is to point and select using the mouse, with its GUI. A picture is worth a thousand of words, as they say.

Windows XP Professional combines all the positive aspects of its Microsoft predecessors. This satisfies all the users who want to prevent frequent crashing of software and want to use easy techniques

1.2   Evolution of the Windows Operating System

Windows XP is the latest version in the series of Windows products in the Operating System.The Apple introduced the concept of Windows but Microsoft popularised the Windows concept. The first version which become reasonably popular was Windows 3.0. For the

1

first time, Windows came with file management utilities and other system tools. Soon, several applications that were meant to be used with Windows appeared in the market. Within a few years, Windows started being used in offices, homes and business establishments. Windows 3.0 was followed by Windows 3.1, which offered better features. Windows 3.1 used a window called Program Manager to launch applications. Almost at the same time, Microsoft introduced Windows 3.11 for workgroups. Now, Windows could be used on a LAN-based networking environment. None of these products was an actual operating system of its own. They were just programs that worked with MS-DOS.

The next major development came with the introduction of Windows 95. Unlike earlier versions of Windows, Windows 95 was a complete operating system. Now, Windows was no longer restricted by the conventions of MS-DOS.

It was easier to start applications in Windows 95. The Program Manager of Windows 3.1 was hidden from the user. This was replaced by new ways of starting applications and opening documents. It also gave the user better facilities to manage application windows, new context-sensitive short-cut menus, improved networking features and so on.

After Windows 95, came Windows 98 with a bang. Windows 98 offered many new utilities at that time, improved the performance and support of the latest hardware technologies of that time. It also provided several features and utilities that allowed easy access to the internet.

In the meantime Microsoft produced Windows NT(New Technology) independent of 9x(95 or 98) versions. Windows NT family produced Windows NT versions 3.5,3.51,4 each of which came in a workstation version and a server version. Some users thought Windows 9x crashed often, that is they felt Windows 9x lacked stability. Some others thought Windows NT lacked compatibility. That is they cannot run some of their favourite programs in NT which could run successfully in Windows 9x. The windows 9x line gave a new offspring Me (Millen-

nium edition) which provided some of the much needed stability. NT line of development resulted in Windows 2000 professional. Windows 2000 professional increased the compatibility of its parent. The never tiring Microsoft development team, at last brought the stability of NT and the compatibility of 9x, under one roof, which resulted in Windows XP Professional through Windows XP home. Windows XP professional is designed to satisfy the insatiable demand of the business community but its immediate predecessor, Windows XP home targetted home users. At the time of writing this book, Microsoft is to introduce windows 2003. The never ending race of ever increasing demand and the quest for satisfying them may continue leading to the vocation of more and more sophisticated with user-friendly products.

1.3   The Mouse

If you want to extract work from the computer, you have to input data. The input can normally be provided by the keyboard and the Mouse. You know the keyboard. If you want to move from one window to another, unless you know the keyboard combinations, it will be very difficult to move one window to another by using keyboard. But the mouse intuitively provides the idea.

As you have learnt in the earlier section, Windows XP uses GUI. That is, all information is displayed on the screen. You can use it by simply pointing to it and selecting. To do this you use the mouse. The mouse is an input device that you move on a flat surface (usually a mouse pad.). When you move the mouse, a pointer moves on the screen. This pointer, called the Mouse Pointer, is used to point to things on the screen. The mouse has either two or three buttons on the top. The left button is the most often used. Described below are mouse actions that you need to know to use Windows XP effectively.

Note - Click on and Click are used interchangeably for example you can write Click on the button or Click the button. Both forms are used in this chapter.

i) **Move:** Moving the mouse is simply dragging the mouse on the mouse pad so that the mouse pointer moves in the direction you want, without touching the buttons. This action allows you to point to things on the screen.

ii) **Click:** Clicking is used to select objects on the Windows screen. To click, ensure that the mouse is pointing to what you want and press the left button of the mouse once and release the button immediately.

iii) **Double-click:** Double-click is most often used to start applications. To double-click, point to what you want and press the left button of the mouse twice in quick succession. You should get used with Double-click; because new comers to the computer field find it difficult to cope with Double-click in the beginning.

iv) **Click and drag:** This mouse action is used to move an object from one place to another. When you click and drag an object, the object moves along with the mouse pointer. To click and drag, hold the left button of the mouse down and move the mouse to the place wherever you want.

1.3.1 Mouse after right click

The right click: Right Mouse button gains a lot of significance now-a-days. If you right click on an item, you will be provided with a context sensitive menu (context sensitive menu changes its contents depending on the situation). This is also called short-cut menu You can experiment with that menu. The context sensitive menu provides almost all the facilities offered by menu as well as toolbars. You can change left mouse button into right mouse button and vice versa. In this case the left click becomes the right click and vice versa. This action may be helpful to the left handed people.

1.3.2 Moving the mouse pointer via the Keyboard

Again you can create the effect of all the above operations by keyboard operations. In the beginning, people are very much attracted by the use of mouse, but when they have to write lengthy programs,

4

changing mouse and keyboard frequently is irksome . Therefore those people who are experts in typewriting prefer to make use of keyboard to bring the effect of mouse click.

The following keys can duplicate the mouse operations. If you want to use your keyboard to do the work of the mouse, you have to follow these steps:

(i)   Click the **Start** button
(ii)   Select the **Control Panel** in the menu and click it.
(iii)   Choose the **Accessibility Options** icon and click on it.
(iv)   It opens a screen , click on **Accessibility Option** under **pick a Control Panel icon.**
(v)   Open the **Mouse** tab.
(vi)   Activate use **MouseKeys** check box if it is not already activated.

Windows XP allows you to move the mouse pointer by using the arrow keys on numeric keypad of the keyboard.

Note 1: Make sure that you have **Num Lock** turned on.
Note 2: MouseKeys do not work with the separate arrow-key keypads found on most modern keyboards.

Besides the basic arrow movements, you can also use the numeric keypad keys outlined here. The following table gives you the equivalent keys for mouse operations.

Key	Equivalent Mouse Action
5	Click
+	Double-click
/	Select the left mouse button
*	Select both mouse buttons
-	Select the right mouse button
Insert	Lock the selected button
Delete	Release the selected button

These keys can be used as follows:

- To double-click an object, use the arrow keys to move the pointer over the object, press the slash key (/) to select the left mouse button, and press the plus sign (+) to double-click.
- To right-click an object, use the arrow keys to move the pointer over the object, press the minus sign (-) to select the right mouse button, and press 5.

- To drag-and-drop an object, use the arrow keys to move the pointer over the object, press the slash key (/) to select the left mouse button, press Insert to lock the button, use the arrow keys to move the object to its desired destination, and press Delete to release the button and drop the object.

- To click an object, use the arrow keys to move the pointer over the object, press the slash key (/) to select the left mouse button (if it isn't selected already), and press 5 to click.

- To right-drag-and-drop an object, use the arrow keys to move the pointer over the object; press the minus sign (-) to select the right mouse button; press Insert to lock button; use the arrow keys to move the object to its destination; and then press Delete to release the button to drop the object, and display the context menu.

Use MouseKeys when Num Lock is on. These options determine the relationship between MouseKeys and the Num Lock key. When On is activated (this is the default), for example, Windows XP will use MouseKeys whenever you have Num Lock is on. If you then turn off Num Lock, you can use the regular arrow keys.

Show MouseKey status on screen: When this check box is activated, Windows XP displays the MouseKeys icon in the system tray. Double-clicking this icon opens the Accessibility Properties dialog box.

1.4   Logging In

If the computer is not already turned on, turn on the computer. If you are the lone user you will be taken into the Windows XP desktop directly. You can continue your work.

Suppose there are multi-users, you will be shown a welcome screen similar to what is shown in figure 1.1 . The aim of logging in is to take you to Windows XP desktop.

Fig 1.1 Logging screen

You can select your account by clicking the appropriate icon or username. (Icon is a small picture / image representing an application, Icon literally means a statue.) If you do not have a user account, do not worry, there is guest account which you can make use of.

You all know PC means Personal Computer. A single person or his family used a computer earlier which made them to store his / their data secretly. When many different people start to work in the same computer, secrecy cannot be maintained. We cannot afford to provide each one with a computer. Such proposition is a costly affair. How about making the people to believe that they are working in their own computers even though they work in the same computer?

7

Note - Windows XP is used to refer Windows XP Professional

Suppose you do not have password you will be taken directly to the Windows XP. Suppose you do have a password (otherwise it will defeat the purpose of having User Account. User Accounts determine who the actual users are. Others may enter in to the system as Guest user) an entry box appears, click the entry box and enter your password into it. Caution must be taken to enter your password because it is case sensitive. Your 'A' is different from 'a'. Your password is determined by the administrater. The administrater governs the computer. Consult your teacher for further details.

This trick is achieved by a user account. If you have a user account in Windows XP Professional, you will be provided with a separate My Computer, My Documents, and some other folders. The only drawback is that your work can be watched by the administrator or administrators, he / she / they has / have special powers to control the activities related to that personal computer with Windows XP Professional.

If you have forgotten your password (which you cannot afford, especially if you are the administrator, in this case nobody can help you) and if you select the help icon for hint, the password hint appears (if you have one).

If you commit mistake while you enter your password, you will be prompted to enter your password again, with some help from the computer. When you are in the welcome screen, Ctrl + Alt + Del key combinations, provide you the dialog box for entering the username and password. If you successfully enter your password, you will be taken to Windows XP Desktop. If your computer is on a network, you may be shown a dialogue window that requests your ID and Password. Provide them.

1.5 Logging Off and Shutting Down

Logging off is the process of closing the desktop and returning to the Windows Log In screen.

Suppose you want to come out of your work, you can do either of the following two. You can close your session or you can shut down the computer.

Suppose you want to Log off without shutting down the computer, follow the following steps :

(1)   Save all your unsaved documents
(2)   Click the Start button (or press Winkey or Ctrl + Esc;
        Winkey pronounced as Win-key that lies between
        Ctrl and Alt keys, the Start menu will be displayed.
(3)   Click the Log off button (or press L or l key)
        Log Off is at the bottom of the Start menu.
(4)   You will be shown Log off Windows. Click Log off
        button (or press L or l key). refer fig.1.2

Fig 1.2 Log Off Screen

Note: Do not try to use **Switch User** button. It may lead to dangerous consequences. It may lead to fatal error; you should restart your computer. You may lose unsaved data in this process.

Suppose you want to shut down the computer you have to follow these steps,alert the other users at that time.

(1) Save all your unsaved documents
(2) Click the **Start** button.

(3) Click **Turn Off Computer** button (or press U or u key).You will be Shown **Turn Off Computer** Window with three options, along with cancel.

(4) If you have changed your mind not to shut down the computer click **Cancel,** button at the bottom**.**

(5) If you want to shut down the computer click **Turn Off** button (or press U or u key).

Some computers especially new ones will automatically shut down the computers**.** Other computers will show you ' It is now Safe to Turn off your computer' message (It is specific to the Configuration). You can switch off the computer. Alternatively you can press the **power** key from the keyboard, if that key is available in your computer. You can also Turn Off the computer by the key combinations of Alt+F4 and then click **Turn Off** button. Another method for turning off is given in customizing the Taskbar.

(6) Suppose you have dual operating system, if you want to switch over to the other operating system then you can click **Restart** button. This will be useful when you install new software also.

(7) There is yet another Choice **Stand By**. This may be very useful for notebook computer. This action will save power. If you have **Hibernate** facility, you can make use of it. Place the cursor on **Stand By** button and press Shift key, **Stand By** will change into **Hibernate**. If you click the Hibernate button (Shift + Click **Stand By**) and Switch off computer, the computer can be started comparatively quickly, in the next time, when you open the system.

1.6 Working with Windows XP

When you switch on your computer, Windows XP automatically starts loading from the Hard disk, if it is your default operating system. While loading, it performs a series of diagnostic tests to check the memory and hardware components such as keyboard, disk drives etc. Once the diagnostic tests are over, Windows XP starts loading files and graphics necessary for the GUI interface. This takes a few

minutes, after which it displays a screen, similar to Fig 1.3.

Fig 1.3 The Desktop

The opening screen of Windows XP is called the **Desktop.** The desktop of your computer may look different from what is seen in Fig 1.3. This is because Windows XP allows you to change the appearance of the desktop.

Your computer may or may not be connected to other computers. Computers that are not connected to any other computers are called Stand-alone computers. Two or more computers can be connected together to form a network. If your computer is in a Network, you have to do some more actions to Start your computer.

1.7   The Desktop

In Windows XP, the basic working platform is the Desktop. Let us understand the desktop with an example. When you study, you use a table, don't you? Usually, you keep all the books and note books that you may need on the table in front of you. You may also keep your pencil box, colour box, a dictionary and a few other things on the table. When you want a particular notebook, you simply reach out to that note-book and pick it up. Window's desktop is very similar to the tabletop. All the programs in your computer are available on the desktop. Here, instead of your hand, you use the mouse pointer to point to things and select them.

The desktop has several **Icons**.Each icon has a label telling you the name of the application it represents. **My Computer,My Documents, My Recent Documents** are some of the standard icons that you can see on the Windows desktop. Each of these icons represents an application that is frequently used. For example, My Computer allows you to see the contents of your computer, install and use new software and hardware. Apart from the standard icons provided by Windows, you can also create icons for the applications that you use frequently and place them on the desktop.

The desktop also contains the **Taskbar** as in Fig 1.4 The taskbar is usually a narrow strip, present at the bottom of the screen. On the left, it has the **Start** button. When you click on the Start button, the **Start menu** appears on the left side of the screen. Using the Start menu, you can start any application that you have currently installed. Next to the Start button is the **Quick Launch Toolbar.** One advantage of using Windows XP is the easy access it provides to the Internet, through the quick launch toolbar which contains icons that allow you to select some commonly used Internet-related applications. On the extreme right is the **Systems Tray** that contains the **Clock** and icons for other utilities . The empty space between the Quick Launch Toolbar and the Systems Tray is used to display buttons for the applications currently being used.

Quick Launch
Toolbar

Notification Area

Start Button
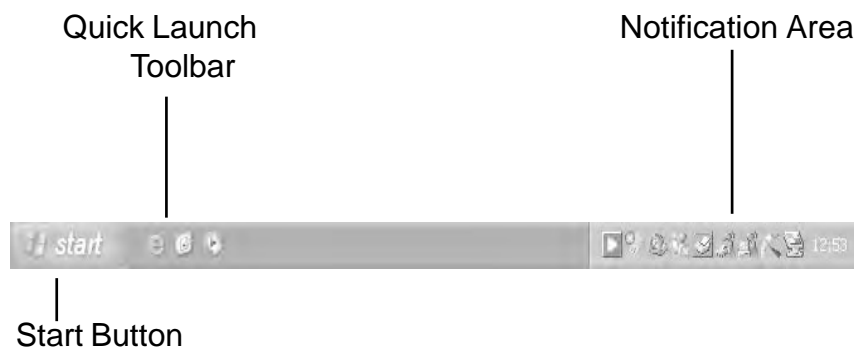
Fig 1.4 The Taskbar

12

1.8 The Start Menu

The Start menu acts as a launch pad for most of the things you want to do with Windows XP. Using this menu, you can start applications, change the settings of your computer, find files, get help and do much, much more. The Start menu appears when you click on the **Start** button on the taskbar.

You can have two different Start menus, one is your usual Start menu and another one is Classic Start menu, which is explained later.

Fig 1.5 The Start Menu

You can select an option from this menu by using the mouse. As you move the mouse pointer over the options, they get highlighted. Simply click the mouse when the option you want is highlighted.

All Programs on the Start menu has an arrow on the right. A right arrow

, whether you are in Start menu or Classic Start menu indicates the presence of one or more levels of submenu. A submenu is shown in Fig 1.6

13

Fig 1.6 All Programs option on the Start menu leads to a submenu

Note - The particulars in the screen may differ from your screen but the general features are the same.

To select an option on the submenu, slide the mouse pointer sideways. One option on the submenu will get highlighted. Now, move the mouse pointer up and down till the option that you want is highlighted and click. Note that some of the options in the submenu also have an arrow. Selecting these options will display another submenu as shown in Fig 1.7

Fig 1.7 Start, Programs, Accessories menus

14

1.9   Starting an Application

Windows XP allows you to start an application in many ways. The most frequently used ones are:

    i)   Using icons on the desktop
    ii)   Using the Start menu

Using icons on the Desktop: The easiest way to start an application is to use its icon on the desktop. When you want to start an application, look for its icon on the desktop. If you find the icon, double-click on it to start the application.

For example, to start the card game Solitaire, look for its Short-cut on the desktop and double-click on it (For creating Desktop shotcuts refer   2.11.2). The game appears on the screen as shown in Fig 1.8
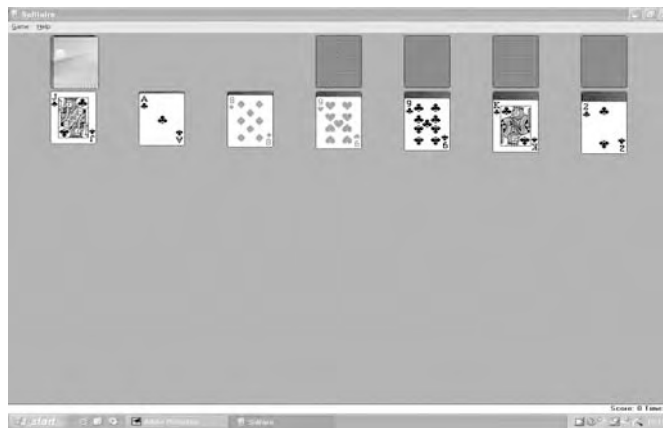
Fig 1.8 Application started using an icon on the desktop

Using the Start Menu: Though Windows XP gives you a few icons on the desktop and allows you to create your own icons for other frequently used applications; it is not possible to have icons for all applications on the desktop. To start applications, for which icons are not available on the desktop; you can use the Start menu. Click on the Start button on the taskbar and select the option that you want from any one of the menus or submenus that appear.

For example, to start the card game Solitaire

(If you have not made a Shotcut), click on the **Start** button, and then click on **All Programs**. Select **Games** from the submenu, which appears, then click on **Solitaire**. (Figure 1.9) .

You can also perform the above operation by keyboard operations alone.

(1)   Press Ctrl + Esc, or Window Key
(2)   Press **P** , this will highlight **All Programs** (Character **P** is
                underlined.
(3)   Press the **Enter** key, submenu will be displayed.
(4)   Press **G**
(5)   In the final submenu press **S** and **Enter** key.

Now you are in the card game Solitaire. Suppose you want to select an item of Microsoft in the submenu of programs, repeated use of M(or m) key will take you to different items that start with letter m (M). After the desired item is selected then follow these steps as given above.

Fig 1.9·Using the Start menu to start Solitaire

16

1.10 Windows

When you are using a table to study, you keep all the books you need on the table. Each book occupies some space on the table. Smaller books occupy less space and bigger books take up more space. The books may even overlap each other partially or completely. You can use these books by moving them around, closing some, opening others and so on. By doing this, you can ensure that the book you want is easily available to you.

Windows XP allows you to work with different applications in the same way. When you start an application, it occupies a rectangular area on the desktop. This rectangular area is called a **window**. You can have several windows on your desktop at the same time. These windows may be big (as big as the desktop) or small (as small as a button on the taskbar), overlapping others or one beside the other. Fig 1.10 shows you the desktop with three windows.

Fig 1.10 Desktop with three windows

1.10.1 Parts of a Window

To work efficiently with windows, it is important to learn to manage them well. Windows XP allows you to move them around, change their size, and hide them from your view and so on. Let us use the application WordPad, to manage windows well.

WordPad is one of the applications that comes as part of Windows XP. It is a simple word processor - you can enter and store text using it. To start WordPad, click on **Start →All Programs → Accessories → WordPad** (figure1.11). The above command means first click the **Start** button, then click **All Programs** in the menu , then click **Accessories** from the ensuing submenu, and finally click **Wordpad** in the last submenu that appears.

Fig 1.11 Starting WordPad

18
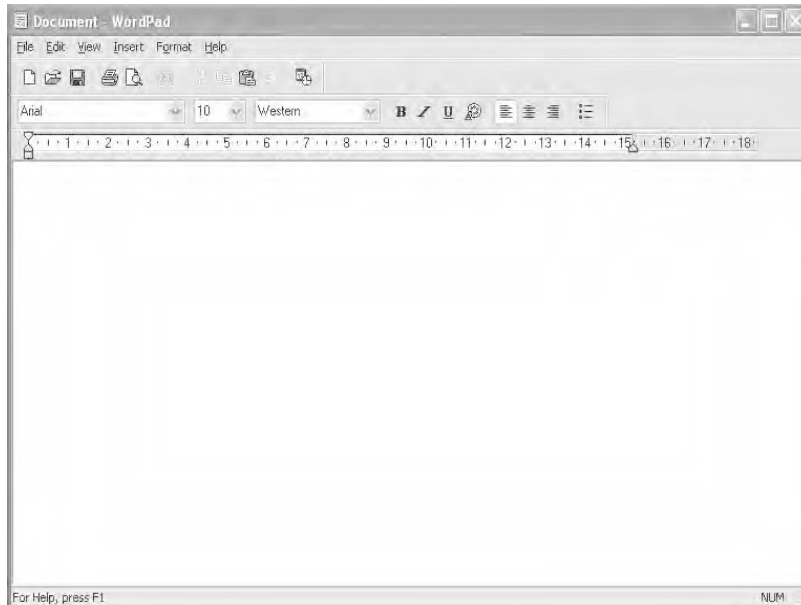
Fig 1.12 WordPad window

The Wordpad window opens. Windows XP is designed in such a way that all windows are similar. The methods used for sizing, moving and closing these windows are also the same.

At the top of each window is the **Title Bar**. As the name indicates, the title bar tells you the name of the application. There is an execption to this general rule. Even though Windows Explorer is an application, it will not show its name in the title bar (You see Windows Explorer in Art 1.19). It also contains three of the following four **Sizing buttons,** at the top of the right corner.

Minimize Button: The minimize button is used to reduce the size of the window to a button on the taskbar. Remember that minimizing a window does not close a window. It simply hides it from you. Fig 1.13 shows the WordPad window minimized.

19

Fig 1. 13 Minimized *WordPad window*

Maximize Button: Clicking on this button enlarges the window to fill the entire desktop. Fig 1.14 shows the WordPad window maximized.
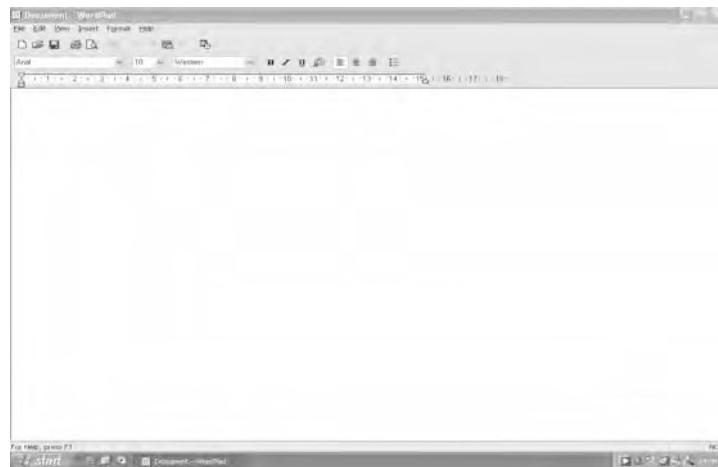
Fig 1. 14 Maximized WordPad window

Restore Button: This button is used to restore the win dow to its original size (that is, to the size before you maximized it).
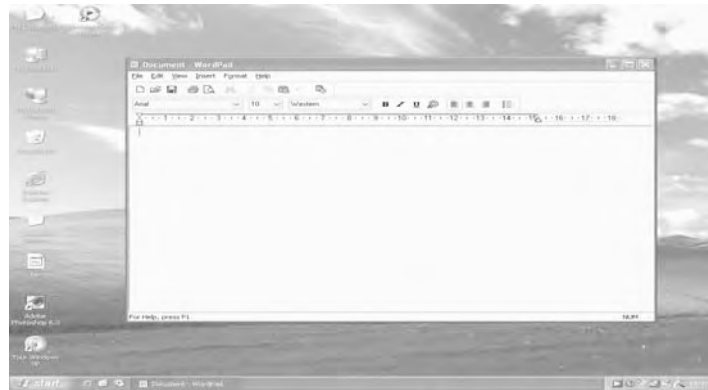
Fig 1. 15 Restored Wordpad window

Close Button: This button is used to close a window. Re member that closing a window will remove its con tents from memory and screen.

Below the title bar is the **Menu Bar.** This displays the different menus available to you. When you click on a menu option, say Edit, all the sub-options appear as a drop-down menu (Fig 1.16). You can select any one of them by pointing to it with the mouse pointer and clicking it.
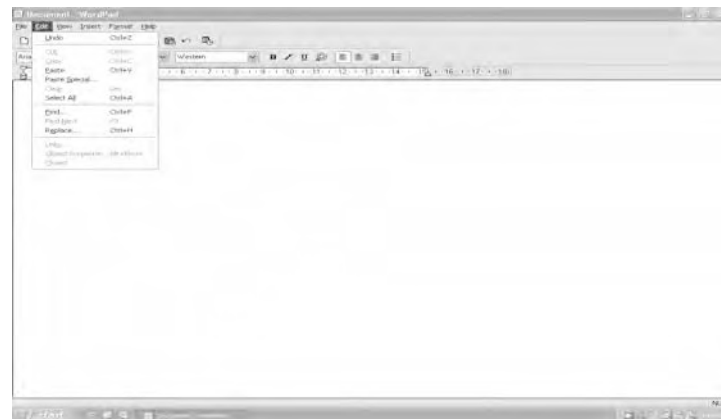
Fig 1. 16 Edit menu

One or more **Toolbars** appear below the menu bar. Toolbars consist of icons representing shortcuts for the most frequently used commands.

21

For example, to save a file, you can click on the **File** menu and select **Save** from the drop-down list. An easier method would be to click on the Save icon on the toolbar. (Ctrl + S (or Ctrl +s) combinations also will save the file). If you save for the first time, you will be prompted to enter the name of the file.

1.10.2 Moving a Window

Often, while working with multiple windows, you need to move a window to different area of the desktop to see one of the underlying windows. You do so by clicking and dragging the title bar of the window.

Note : You cannot drag a Window when it is either maximized or minimized.

1.10.3 Changing the size of a Window

Every window has a **Border** that can be used to change its size. Point to the window border with the mouse. The mouse pointer changes into a double-headed arrow. Click and drag this arrow to increase or decrease the size of window.

To change the length and breadth of the windows simultaneously, you have to move the mouse pointer to either of the bottom corners of the window. Now, the mouse pointer changes into a double headed arrow as said above. Click and drag the arrow to increase and decrease the length and breadth of the window simultaneously.

1.11 Windows Dialog Boxes

Windows XP is an inter-active operating system. Its GUI attempts to display as much information on the screen as possible. It uses dialog boxes to display the information and allows you to either type in your response or select from a list of choices. Listed below are some of the controls used in dialog boxes.

Text Boxes: Text boxes are used to allow the user to enter some data. Every text box is accompanied by a prompt or label that tells you what should be entered in that box. Fig 1.17 shows a window with a text box.
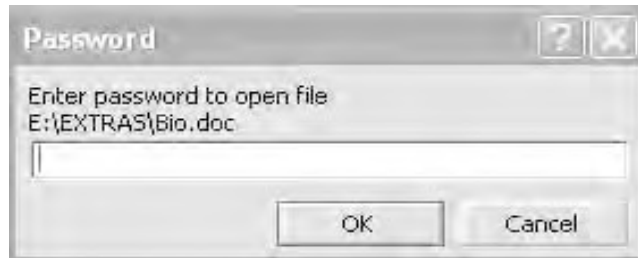
Fig 1. 17   A window with a Text box

List Boxes: These boxes display a list of choices. You can select the one you want by simply clicking on it.
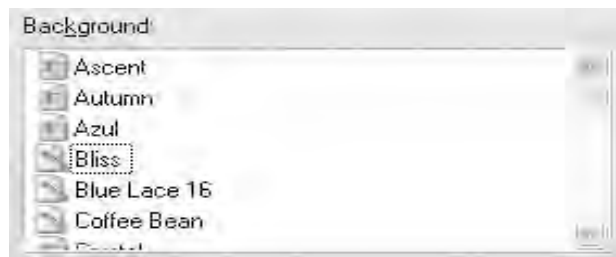
Fig 1.18   A List Box

Drop-down List Boxes: These are list boxes which have a small black inverted triangle at one end. When you click on this triangle, a list of options drops down in front of you. You can select an item from this list by clicking on it. This is used when there is limited space.
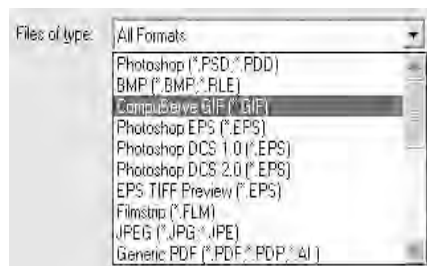
Fig 1. 19 A Drop-down List Box

Radio Buttons: Fig 1.20 displays a dialog box with 2 Radio buttons. These buttons are used to display multiple options. You can select one by clicking on the small white circle to the left of the option. A black dot appears at the center of the circle to indicate a selected option. In radio button option, you can select only one of the buttons. If you select a second radio button, the previously selected button is automatically deselected. If you have to answer multiple choice questions with several options in which you have only one correct answer then Radio Buttons are the suitable candidates.
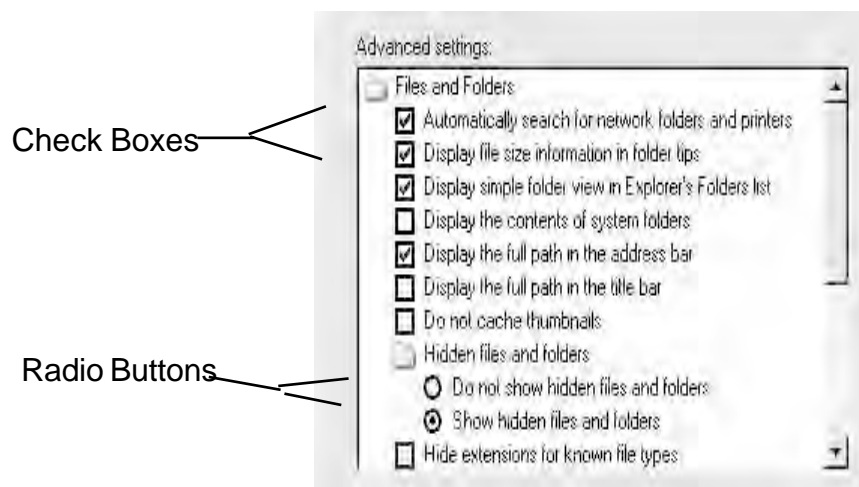
Fig 1. 20 Radio Buttons and Check boxes

Check Boxes: These boxes are used to enable or disable options. The options in these boxes have small white squares to their left. Clicking on a square enables the option and clicking on it again disables it. A tick mark in this square indicates that the option is enabled and a blank square indicates that the option is disabled. You can select any number of check boxes in the given option.

Buttons: The OK and Cancel buttons are the most frequently used buttons in Windows XP. When you click on a button, the related command is carried out. For example, if you click on the OK button in a dialog box, Windows will accept your choices and close the dialog box. Clicking on Cancel will make Windows ignore the changes and

24

close the dialog box. Some buttons are also used to display another dialog box.

Tabs: Tabs are used to display different sets of options in dialog boxes. Fig 1.21(a) and 1.21 (b) display a dialog box with five tabs. Clicking on each, displays an entirely different set of options. Fig 1.21 (a) shows the dialog box with the second tab **Desktop** selected. In Fig 1.21 (b), the third tab, **Screen Saver** has been selected.
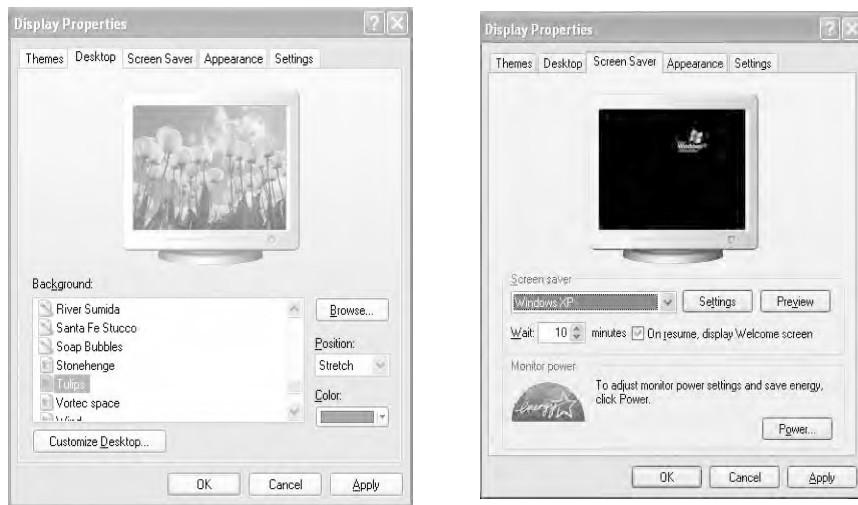
Fig 1.21 (a) Desktop Tab is Selected Fig 1.21 (b) Screen Saver Tab is Selected

Sliders : Sliders are used to enter a value by physically moving a marker over a slide. Fig 1.22 shows a dialog box with sliders to increase or decrease volume levels.

Fig 1.22 Sliders

25
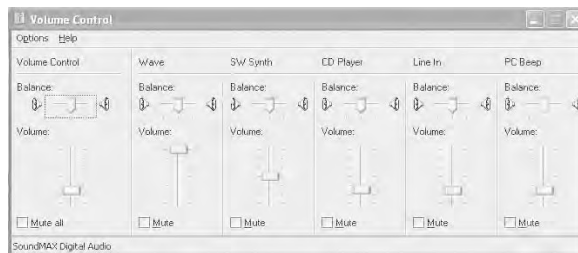
1.12 Help and Support Center

Even though this chapter tries to help you to make use of Windows XP Professional, it is impossible to include all the facilities available in Windows XP Professional in a tiny chapter. How can you access the remaining facilities offered by Windows XP Professional ? As you know, self-help is the best help. The Microsoft provides lot of help in its **Help and Support Center** in Windows XP.

Actually Windows "**Me**" introduced **Help and Support System** by substantially improving the help methods available in earlier versions of Windows. Windows Me, by combining many more external resources, introduced a Web-style interface to replace old-type Help-file interface of the earlier versions of Windows. Windows XP improved the help facilities of Windows Me remarkably. If you have an Internet connection, you need not use Internet Explorer to access the Microsoft knowledge Base. You can search it directly from **Help and Support System**.
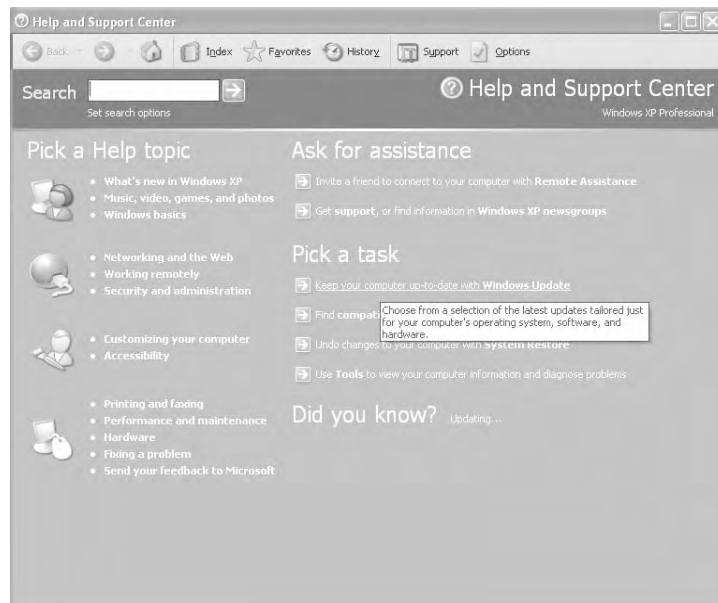
Fig 1.23 Home page for Help and Support Center

26

Microsoft knowledge Base is an online database of questions and answers **Start** → **Help and Support** (or press F1 key when you are in **Windows XP**) will provide with the help relevant to the program. Always make use of Winkey + F1 key combinations in order to avoid ambiguity.

This will take you to the **Help and Support Center** of XP without fail, wherever you are. The Home page may slightly be different from what you see here because of customization (Customization is the process of changing default setting to suit your needs and tastes). Except in dialog boxes, each **Help and Support Center** window has a title bar, which shows **Help and Support Center** as the title along with the **Minimize**, **Maximize** and **Close** buttons.

There is no Menu bar. It has a Toolbar, taking you to go around (navigate or travel) the Help topics. So it is called Navigating toolbar. Below the Navigating toolbar appears the **Search** bar. Below this bar, information is provided. If you are lucky enough, you can get the desired help by clicking a topic from **Pick a Help Topic** which may solve your problem.

If you pick a topic from the **Ask for Assistance** that will take you either to **Remote Assistance** or to **Support and Windows XP news groups**. Any one of these may solve your problem if you have a Internet connection. But beginners may find it difficult to understand the help provided by the above. So they should be content only with what they have with internal assistance . You should make familiar with **Pick a Task** and "**Did you Know?**" by yourself. If the item for which you need help may not be available at the home page, then enter the word or phrase into the search text box. Then press **ENTER** key or click the **Go** button (à) situated to the right of **Search** text box. Suppose you have entered "**view pictures**" in the **Search** text box, **Help and Support Center** displays **Search Results** pane on the left side and adds a toolbar containing **Add to favorites**, **Change view**, **Print** and **Locate in Contents** buttons in the right pan. Search results are shown below this Toolbar. This Help page is context sensitive.
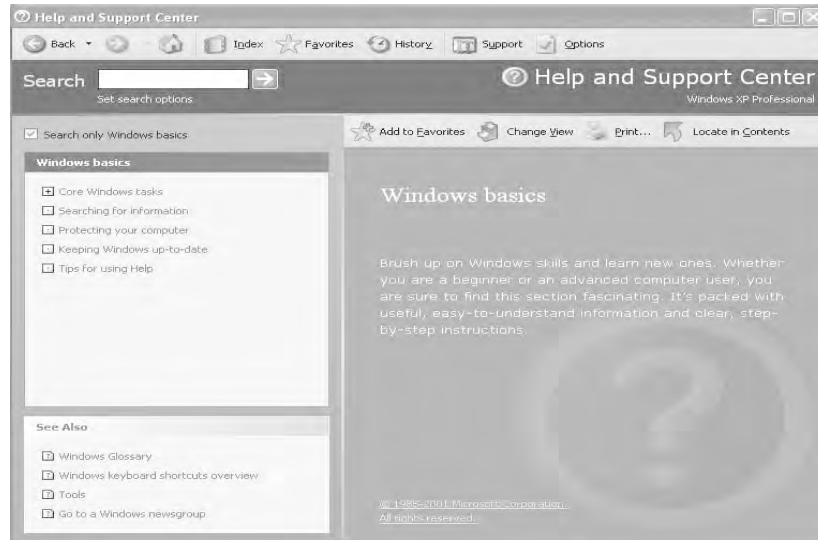
Fig 1.24 Windows Basics Help

If you have not customized the search results pane, it is divided **into Suggested Topics**, **Full Text Search Matches** and **Microsoft knowledge Base**. If you do not want to make use of Microsoft Knowledge Base or if you do not have Internet connection, you can hide Microsoft Knowledge Base. You can see only the first two options, the procedure for hiding Microsoft Knowledge Base will be given later.

Suggested Topics: **Suggested Topics** are keyword matches (these topics contain the word / words that you have entered in the Search text box as keyword / keywords). These topics are further classified into **Pick a Task** and **Overviews, Articles and Tutorials**.

Full-Text Search Matches: Full-text matches are topics that contain the word / words you entered into the Search text box , into the body of the text of the help topics. Here the word / words will not be treated as keyword / keywords.

Microsoft Knowledge Base: The results found in this category are from Microsoft Knowledge Base. Use it if you like. You should have Internet connection; in order access this knowledge base.

If you want to display the help content, first click the category and click a search result of interest, the result is displayed in the right pane. Some help text pages will have highlighted and colored text. You see their uses below.

Highlighted text: The matched word / pharse with what you had entered is highlighted. The highlighting serves no other purpose. If you click on those highlighted word / pharse nothing happens. If highlighted words occur often in a text, it is an annoying experience. You can get rid off those highlighting if do not like it. You will be shown the procedure later.

Blue underline text: If you click the blue underline text, it will open the item associated with the text.

Green underlined text: If you click on this term it will provide the definition of the term.

Already you have seen that three or four activated useful buttons on the right pane. Now you are going to see their usefulness.

Add to Favorites: If you see a help page and if you feel that will be useful to you for future reference, just click **Add to Favorites** button, that page is immediately copied and Windows XP Professional will announce that your wish is fulfilled. If you want to see the contents, you click **Favorites** in the navigation bar. In the left pane under the **Favorites** heading, opens what you have stored so far. If you double click any one of the topic, the contents will be displayed in the right pane (you can also single click on any one of the topic and click Display button at the bottom). You can use rename or remove buttons as usual.Rename is used to change the default name. This topic is explained later.

Change View: In order to reclaim more space, you can hide the left pane by clicking the **Change View** button. If you again click **Change**

View button, the left pane will appear once again. You can also perform the above action manually. You can drag the right pan to the left, so that a right pan may occupy the entire screen.

Print : you can print the help pages with this button.

Locate in Contents: If you click the **Locate in Contents**, it will display a table of contents for help and support in the left pane. The heading of the current help page is highlighted.

Help Index Button: Use if you know the first letter or first few letters of an item to be searched. You may feel a list that starts with that letter / letters may be helpful. If it is the case you click Index button on the toolbar. The left pane turns into an index.

Under this you can find **Type in the keyword to find** prompt, below this, there is the text area. You enter a letter or few letters into the text area, the index will automatically change to word / words that started with the entered letters. You can click the appropriate entry from the list and then click the Display button or double click the desired item. If necessary use the vertical scroll bar available at the right end of the left pane.

Fig 1.25 View Folders Help

30

Now concentrate on the navigation bar, refer Fig 1.25

Back: This is the first button in the navigation bar from left, after navigating to another page in help, if you want to move to the previous help page, clicking **Back** button will take you to the previous page. This process can be repeated until the back button is disabled. This button is disabled in the beginning.

Forward: This is the second button in the navigation bar from left. After you click the back button the forward button is enabled. You move forward by clicking the **Forward** button until it is disabled. This button is disabled in the beginning.

Home: This is the third button from left in the navigation bar. If you want to return to the home page, click on the **Home** button.   Y o u have already seen **Index** and **Favorites** buttons available in the navigation bar.

History: This stores a list of help pages you have visited recently, in the left pane. As usual double clicking any title will redisplay that help page in the right pane.

Support: It provides the other forms of technical supports available from Microsoft .

Options: This button is helpful in cutomizing the **Help and Support Center.**

1. Click the **Options** button on the navigation bar. **Help and Support Center** displays the **Options** screen.
2. Click **Set search options** in the left pane. **Help and Support Center** displays the set search options in the screen.
3. If you want to change the number of search results provided by **Help and Support Center** , you change the number in **Return up to 15 results per provider** by a number less than 100. The default value is 15 .

4. If you want to get rid off search highlights then deselect it. You make the other desired changes. If you do not want to access Microsoft Knowledge Base again you deselect it.
5. Similarly by clicking **change Help and Support Center options** in the left pane you can make the other changes.

Getting Help Online:

If you want to get help from Microsoft's web site, first of all you need an Internet connection and web browser. Microsoft website includes support for all the products, not just windows XP. Suppose you want to have help for " view folder " you have to give the command as XP+view+folder. The blank spaces should be replaced by +signs. XP indicates, you want to get help from windows XP in order to get help you have to undergo the following steps.

1. Make sure you are on line and use your web browser to go to http://search. microsoft. com.

2. In the Search text box that appears, you type XP+view+folder in Choose a Microsoft.com location, enter United State. Click the **Go** button. After some time the results will be displayed.

1.13 Customizing Windows XP

One of the most attractive features of Windows XP is that it allows you to customize the desktop. You can change the appearance of the desktop by changing the background, adding icons, moving icons, moving and resizing the taskbar and so on.You can also add Screen Savers.

1.13 .1 Customizing the Taskbar

The Taskbar is usually at the bottom of the desktop. But you can move it easily to any of the four sides of the desktop, unless it is locked. To do so , point the mouse pointer to any empty area on the taskbar. If you have opened many windows, then there will not be empty space on the taskbar. In this case you can make use of the space occupied by the

clock. Click and drag the taskbar to wherever you want it to be. Fig 1.26 shows a broad taskbar at the Top.

You can also change the size of the taskbar. Point to the edge of the taskbar. The mouse pointer will change into a double-headed arrow. Click and drag the mouse to increase or decrease the size of the taskbar.
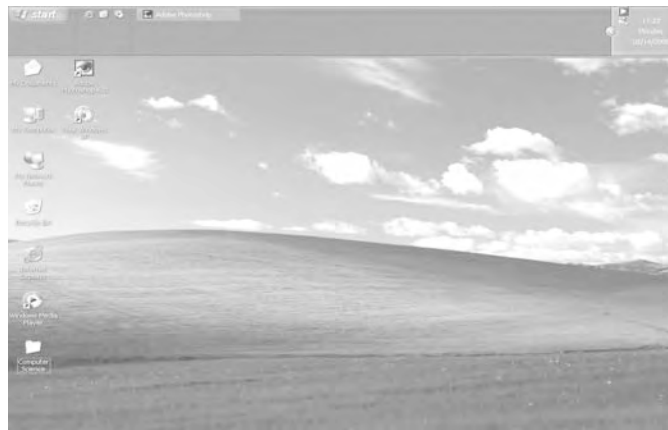
Fig 1.26 A broad taskbar

Taskbar Settings - Right click on the empty area of the taskbar. If there is no empty area right click on the clock. From short cut menu, by selecting Toolbars, you can add or delete tools. If you click **Address**, Address toolbar is created and increase its size by dragging with mouse. Then you can enter any command, that will be executed either directly or accessing the Internet. you can arrange the windows with **Cascade Windows**, **Tile Windows Horizontally** and **Tile Windows Vertically**. **Show the Desktop** is an substitute for **Show Desktop** button. With clicking Task Manager → Shutting down, you can perform Turn off , Restart, Hybernate, Stand by and Switcher user the computer. If you want to make the position of Task bar fixed, click **Lock the Taskbar**, then a tick mark appears against it. Now the taskbar cannot be moved. If you click on **Properties**, you will be shown **Taskbar and Start Menu Properties**. It opens under the Taskbar tab, you customize it to suit your needs and tastes. If you open **Taskbar and Start Menu properties** under **Start Menu** tab, and if you want to change

Start Menu into **Classic Start menu**, click Classic Start menu's Radio button, and click ok . A screen similar to the starting screen of Windows 98 will be shown.

1.27 A Screen similar to the starting screen of Windows 98

1.13.2 Changing the Wallpaper

Wallpaper is the background display that appears on your desktop. You can choose from several standard Wallpapers that are available as part of Windows XP. You can also use a picture that you have drawn, scanned or copied from somewhere. To do so, right-click anywhere in the blank area of the desktop. The menu shown in Fig 1.28 pops up.

Fig 1.28 Choose Properties to change Wallpaper,
Screen Savers etc.

34

Click on **Properties.** The **Display Properties** dialog box appears. Select Desktop tab (second from left) which will present a figure similar to the one shown below (Fig 1.29).

Fig 1.29 Display Properties dialog box

Browse through the list of wallpapers and click on the one you want. A preview in the top half of the window shows you how the wallpaper will look. Click on **Apply** and then on **OK**.

Fig 1.30 Desktop with Setup wallpaper.

35

1.13.3 Using Screen Savers

In old monitors, if you left the images on the screen unchanged for long, the characters would burn-in, leaving a permanent impression on the screens. To avoid this screen savers were used. Constantly moving technology has improved so much that screen savers are no longer necessary. But, they are still popular mainly because they are fun. To use a screen saver, click on **Screen Saver** tab in the Display Properties dialog box. Click on the drop-down list box just below the Screen Saver prompt. A list of available screen savers appears as in Fig 1.31

Fig 1.31 List of screen saver available

Select one. A preview appears in the top half of the window

36
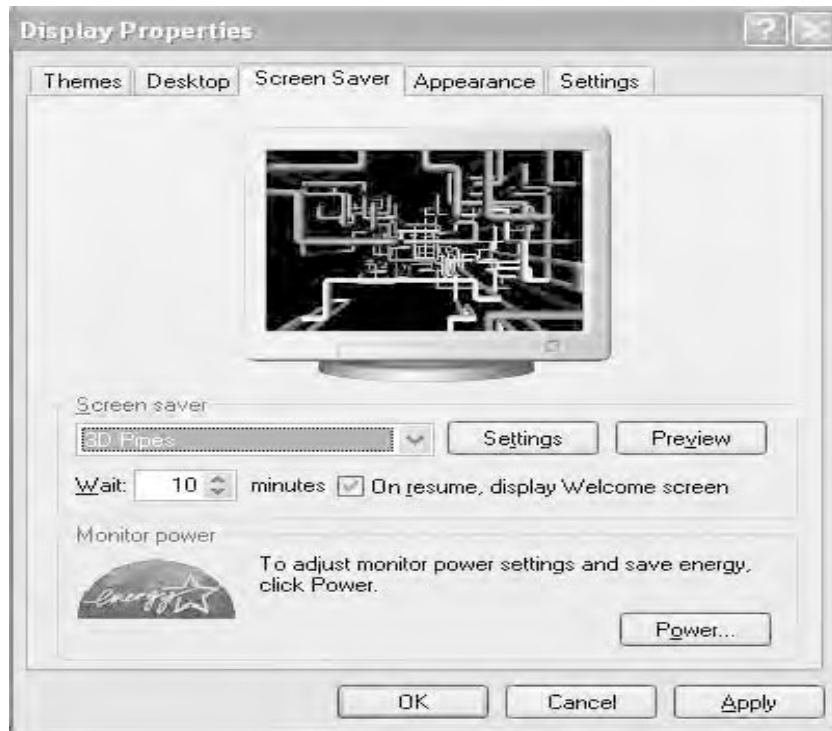
Fig 1.32 Preview of 3D Pipes Screen Saver

You can specify, in the **Wait** text box, the number of minutes the computer should wait before displaying the screen saver. According to the Fig 1.32, Windows will wait for 1 minute before displaying the screen saver.

you can protect your PC by giving Password to the screen saver.
Now, whenever your computer is idle for some time, Windows will automatically activate your screen saver. To remove the screen saver, just move the mouse or press any key on the keyboard.

1.14 The Control Panel

The Control Panel allows you to install and manage the different hardware components attached to your computer. You can open the Control Panel window by clicking on the **Start** button, and then **Control Panel.**

37

Fig 1.33 Opening Control Panel

You can also access Control Panel from My Computer window. Double-click on **My Computer** icon on the desktop and select **Control Panel** from the icons displayed in the My Computer window.

Control
Panel
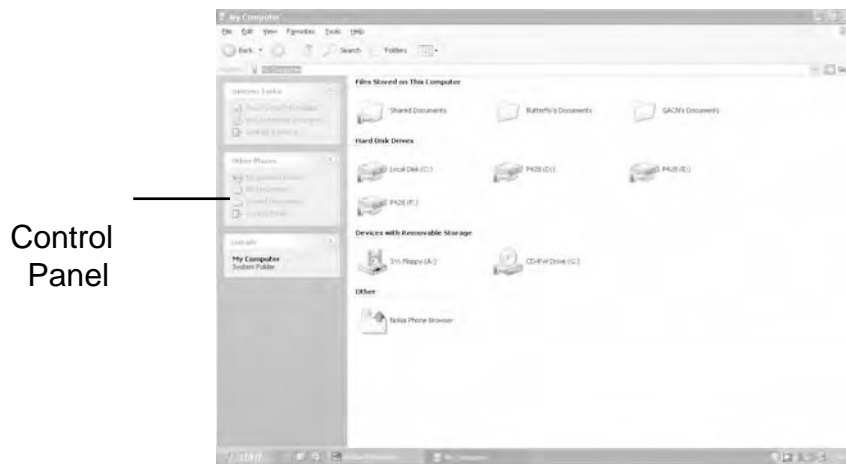
Fig 1.34 My Computer window

38

The **Control Panel** window opens in front of you. Winows XP Proffessional provides completely a new look to the **Control Panel**. It provides two views to **Control Panel**. The default view is **Category View** and the other one is **Classic View**. **Classic View** is similar to the one available in Windows 98. Both views are shown in Fig 1.35 (a), Whatever be the view all the **Control Panel** applets work in the same manner (applet is a small program). Many dialog boxes have new names, new tabs and new functionality. If you are in **Category View** you can click on **Switch to Classic View**. This will take you to the Classic View. When you are in **Classic View**, you will be shown **Switch to Category View**. If you click on **Switch to Category View**, you will be taken to **Category View**. **Category View** Fig 1.35(a) and **Classic View** Fig 1.35(b) of the Control Panel are shown below.

Fig 1.35(a) Category View   Fig 1.35 (b) Classic View

As you can see, the **Control Panel** window displays several icons. Using these icons, you can modify the system and hardware settings of your computer. Listed below are a few of these icons and their description.

Accessibility Options	This allows you to adjust your computer settings for vi sion, hearing and mobility deficiency
Date and Time	This allows you to set the date, time and the time zone for your computer
Display	This allows you to change the appearance of your desktop, such as the background, screen saver, colour, font size and screen resolution.
Fonts	This allows you to add, change and manage fonts on your computer.
Keyboard	This allows you to customise your keyboard settings such as the blink rate and character repeat rate.
Mouse	This allows you to customise settings such as the button configuration, double click, speed, mouse pointer and motion speed.
Printers and Faxes	This allows you to install printer and fax Printer and helps you add new ones.
Regional and Language ...	This allows you to customize setting for the display of languages, numbers times and dates.
User Accounts	This allows you to change user account setting and password to people who share this computer

You are provided with more icons in the **Control Panel**. To know how to utilize them, rest the mouse pointer for one or two seconds on the Icons, you will be shown their usage by means of pop-up message.

Summary

❖　　Windows XP is an operating system.

❖　　The opening screen is called the Desktop. It contains icons and Taskbar. Icons are small pictures representing applications. The Taskbar has the Start button, the Quick Launch toolbar and the Systems Tray.

❖　　The Start menu acts as a launch pad for most of the applica tions in the computer.

❖　　You can start applications using the icons on the desktop or the Start menu.

❖　　The rectangular area on the desktop that is used by an applica tion is called a Window.

❖　　Every window has a title bar with sizing buttons, menu bar, tool bar and borders.

❖　　A window can be moved, resized or closed.

❖　　Windows XP allows you to customize the desktop and taskbar.

❖　　The Control Panel allows you to install and manage different hardware and software components in your computer.

❖　　It is always a good practice to shut down the computer properly before switching the power off.

Exercises

I.   Fill in the blanks

1.   Windows XP is an.....................................

2.   Windows XP uses a...

3.   Clicking on the Start button opens the....................................
     menu.

4.   The clock is displayed on the...

5.   The.. also has buttons representing ap-
     plications currently being used.

6.   .... is one of the options on the Start
     menu.

7.   .... are constantly
     moving images that appears when the computer has been idle
     for some time.

8.   You can move a window by clicking and   dragging
     its............................

9.   The ....dialog box is used to
     change the wallpaper, screen savers, etc.

10.   The...icon on the Control Panel allows
     you to view, add or remove fonts.

II.   State whether the following are True or False

1.   Windows 4.1 was a very popular version of Windows.

2.   You can password protect your PC through screen saver.

3.   The contents of memory are not saved when you shut down your
     computer.

42

4.   When you minimize a window, it automatically closes.

5.   In Windows XP you have to type all the commands.

6.   The Start button is always visible on the desktop,

7.   Windows uses dialog boxes to display information.

8.   Check boxes are used to enable and disable options.

9.   You can access the Control Panel from My Computer.

10.   A minimized window can be restored by clicking on it.

III.   Answer the following

1.   What is Windows XP?

2.   Write a short note about the evolution of Windows operating system.

3.   What are the advantages of using Windows XP?

4.   What is a mouse? What are the different mouse actions that you are familiar with?

5.   What is desktop? What are the things you see on the desktop?

6.   How can you customize the desktop?

7.   What is the Control Panel? Describe briefly some of the icons found on the Control Panel.

8.   What is Shut down? Why should you shut down your computer properly?

9   Describe the different parts of a Window.

10   Write a short note about the different kinds of dialog boxes that you use in Windows.

1.15 Applications

All information in Windows is stored as files. These files are broadly classified into two categories:

i)   Application Files
ii)   Document Files

Application Files: Application files (also called Program files) are files with which you can do something. For example, files that allow you to draw and paint, enter and save text, calculate and play games are application files.

Document Files: Document files are files that are created by the user using an application. In the last chapter, you learnt how to start an application. You can start an application by clicking on its icon on the desktop or by using the Start menu. When you do this, the application appears on the screen in a window. At the same time, a button representing the application also appears on the taskbar. This button stays on the taskbar as long as the application is active and disappears only when you close the application. In this chapter, you are going to learn about some of the commonly used applications of Windows XP. You will also learn how to start multiple applications, how to switch between them and how to transfer data between them.

1.16 Using Applications in Windows

Several useful applications come as part of Windows. Using them, you can perform a wide variety of tasks. Discussed below are some of the commonly used ones

1.16.1 MS-DOS

Before the introduction of Windows, MS-DOS was one of the very popular operating systems among PC users. Hundreds of DOS-based

applications were available in the market. To start such programs or to use any DOS Command, the **Command Prompt** option of windows can be used. Perform **Start → All Programs → Accessories Command Prompt**.

Fig 1.36 Getting MS-DOS Window

A window as shown in Fig 1.37 appears on the screen.

45
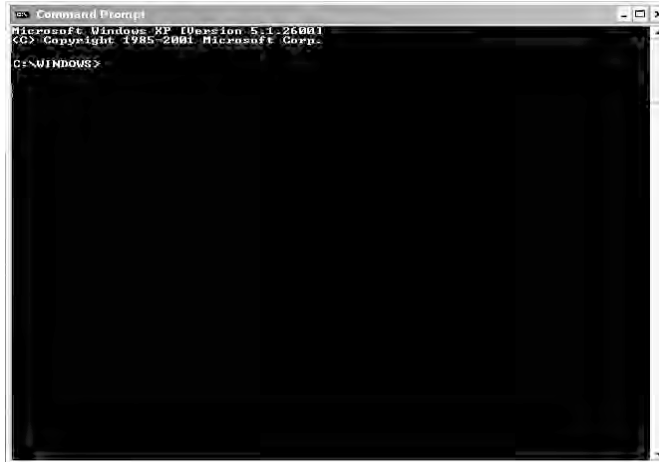
Fig 1.37 MS-DOS window

The MS-DOS window is like any other window; you can move, minimize, maximize or close it like any other window. Notice that after the copyright message, the window displays the familiar **C:\>** prompt. You can use any DOS command here.
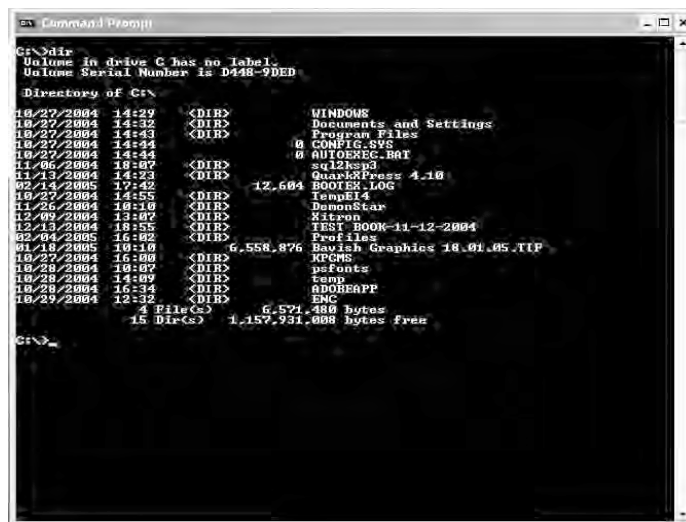
Fig 1.38 DOS window with the dir command

Start the MS DOS application and try out the Dir command. Also try out any other DOS commands that you are familiar with. Close the application window after you finish.

1.16.2 Clock and Calendar

Windows has an in-built clock, which is usually displayed on the taskbar. To change the date or the time, double-click on the clock on the taskbar. The **Date and Time Properties** dialog box appears on the screen. To change the date or time , you should have special privilege. Only administrator can undertake these activities.
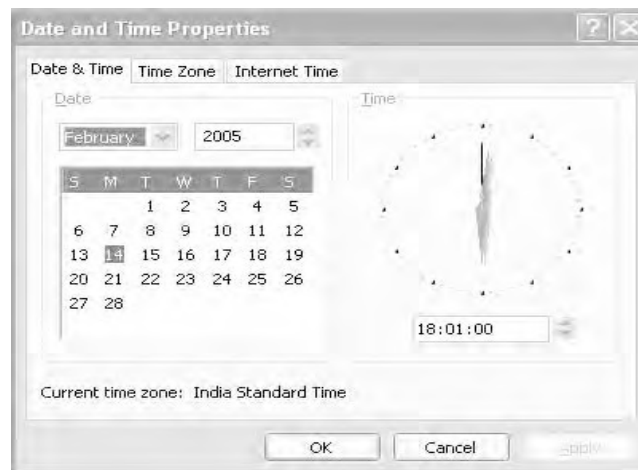
Fig 1.39 Date and Time Properties dialog box

On the left half of the dialog box, the current month's calendar is displayed. To view the calendar for some other month, click on the month and drop down list box and year spinner box and select the month and year you want.

To change the time, click on the digital clock seen on the right. Highlight the hour, minute or second by dragging the pointer over it. Increase or decrease the highlighted value by clicking on the up and down arrows in the box. Note that the time in the analog clock also changes correspondingly. Analog clock is the ordinary clock with hour hand minutes hand and second hand.
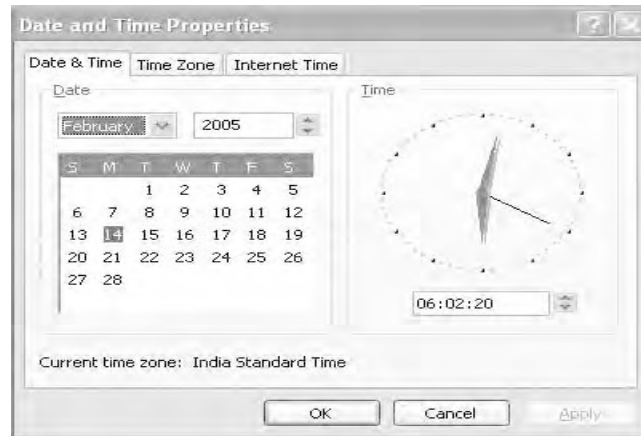
Fig 1.40 Clock showing a different time

Click on **OK** after you finish.

1.16.3 Calculator

The Calculator is a useful application that comes with Windows. It can be used to perform mathematical and scientific calculations. To start the Calculator, execute the following action.

Start → AllPrograms → Accessories → Calculator.

Fig 1.41 The Calculator

48

The Calculator can be used in one of the two modes - **Standard** mode or **Scientific** mode. Fig 1.41 displays the Calculator in the Standard mode. As you can see, this calculator is very similar to an ordinary calculator. You can use the keyboard and the mouse to enter numbers and operators. If you are using the mouse, click on the number and operator buttons. The numbers that you have entered and the results will be displayed in the display bar just below the menu bar. If you have selected **Digit grouping** under View menu, the numbers are separated by comma following the European convention. The numbers that appear to the left of decimal places are separated by comma for every three digits starting from the right . The leading comma (if any) is suppressed.

To use the Calculator in the **Scientific** mode, click on the **View** menu and select **Scientific.** Fig 1.42 shows the Calculator in the Scientific mode, with statistics box.

Fig 1.42 The scientific calculator

49

To calculate sum, average and S.D of given numbers execute the following steps

1)   Enter the first number.
2)   Click **Sta** button.
3)   Click **RET**
4)   Click **Dat** button.
5)   Enter the next number.
6)   Click **Dat** button.
7)   Repeat step 5 and step 6 until you exhaust all the numbers.
8)   Click the s button.
9)   The Standard Deviation of the entered numbers is displayed.
10)   Click **Sum** you will be shown sum of the numbers entered.
11)   Click **Ave** you will be shown average of the numbers entered.

Note: If you click **Sta** button, you will see Statistics Box. The entered numbers are in Statistics Box

 If you click the LOAD button of Statistics Box, the highlighted number in the display area of the Statistics Box will be loaded into the display area of calculator display area.

If you click CD button of Statistics Box, the highlighted number in the display area of Statistics Box will be deleted from the list of numbers. If you click the CAD button of Statistics Box, all the entered numbers are deleted.

1.16.4 Paint

Paint is an application that lets you draw and colour pictures. To start Paint, click on **Start →All Programs→ Accessories→ Paint**
 (Fig 1.43).
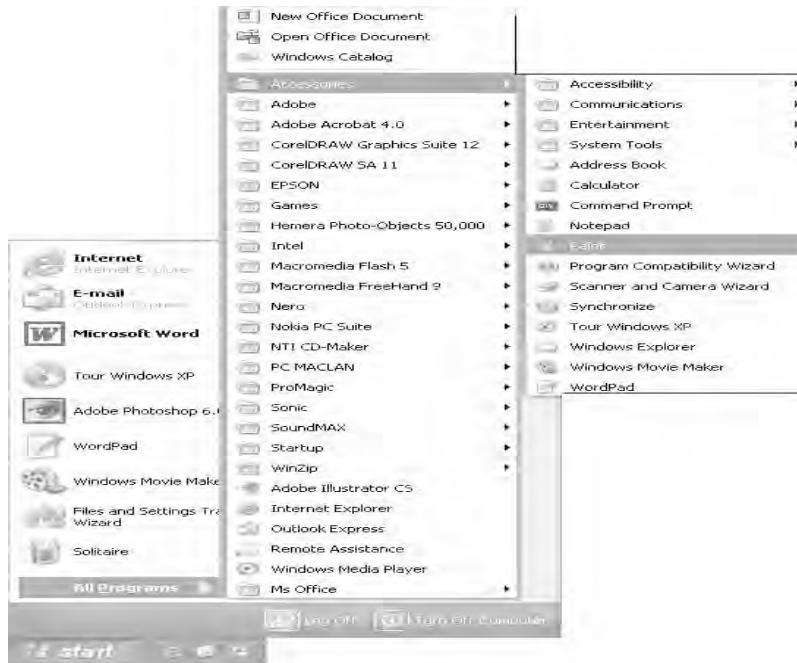
Fig 1.43 Starting Paint

The paint window appears on the screen.

Menu Bar

Toolbox

Drawing area
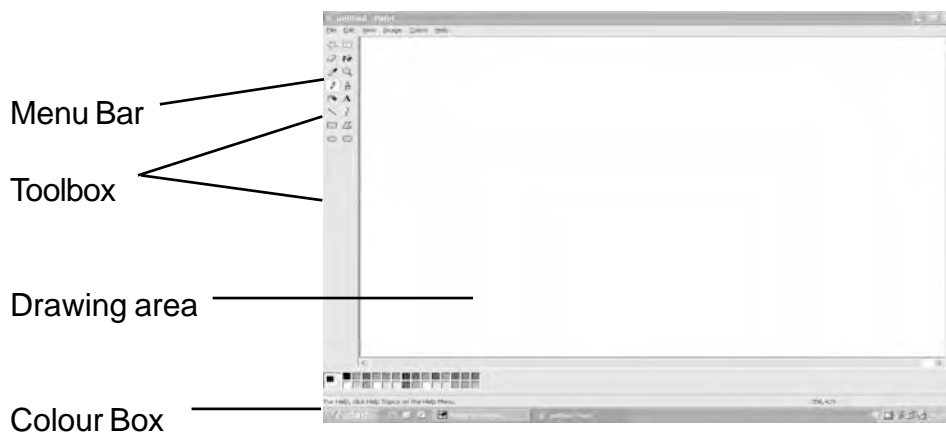
Colour Box

Fig 1.44 Paint window

51

Just like any other window, the Paint window also has a title bar with sizing buttons, a menu bar and a status bar. In addition, it has a **Toolbar** and a **Colour Box**. The Toolbar has various tools that you can use to draw and colour. Fig 1.45 shows the Toolbar with the different tools.

Free-form Select	Selection
Eraser	Fill with colour
Pick colour	Magnifier
Pencil	Brush
Airbrush	Text
Line	Curve
Rectangle	Polygon
Ellipse	Rounded Rectangle
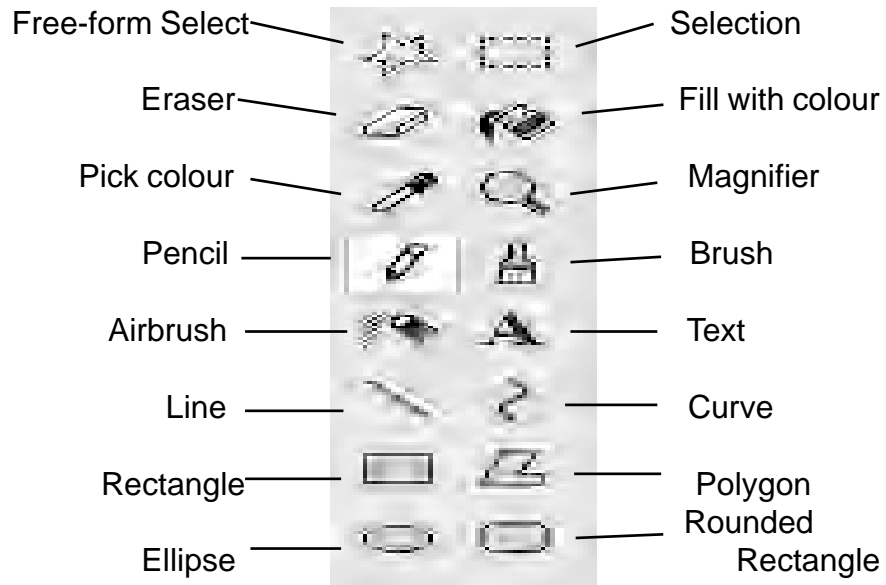
Fig 1.45 The toolbar

To use any of the tools in the toolbar, first click on the tool to select it. For example, click on the ellipse tool.
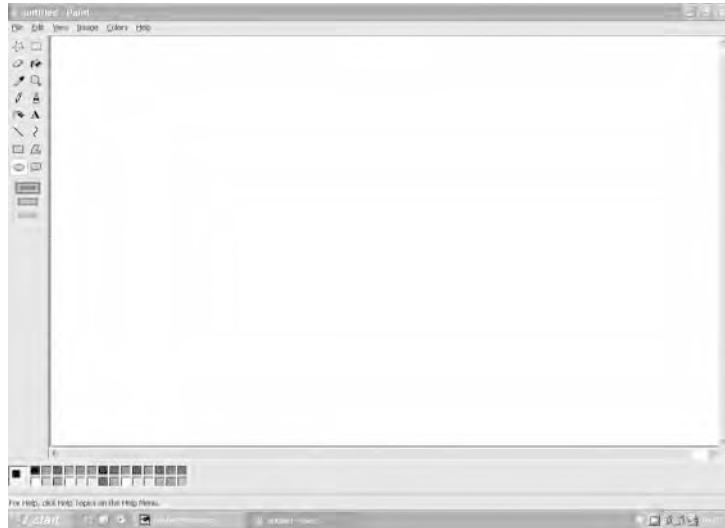
Fig 1.46 The circle tool is selected.

Then, move the mouse to the drawing area and click and drag to draw the figure you want.
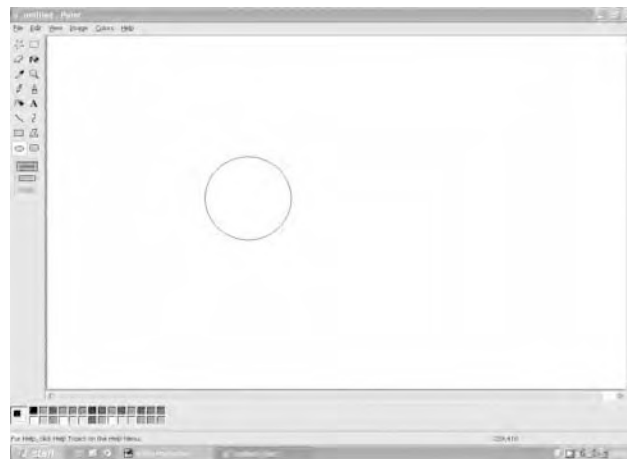
Fig 1.47 Click and drag the mouse to draw

The Colour Box contains the colours that you can use. Click on the colour of your choice and use the Fill with colour tool, the Airbrush or the Brush to colour your pictures.

You can close Paint by clicking on the **Close** button on the title bar or clicking on the **File** menu and selecting **Exit (**or **Alt+F4** keys)**.**

Do-it-Now Exercises

1.   Open the Paint application and draw a colourful bunch of balloons.
2.   Draw a simple house and colour it.
3.   Draw a colourful kite.
4.   Draw a flower of your choice and colour it.

1.16.5 WordPad

WordPad is a simple word processor that comes along with Windows. A Word processor is a program that allows you to type and store text. To start WordPad, click on
Start → All Programs → Accessories → WordPad.
The WordPad window appears on the screen.
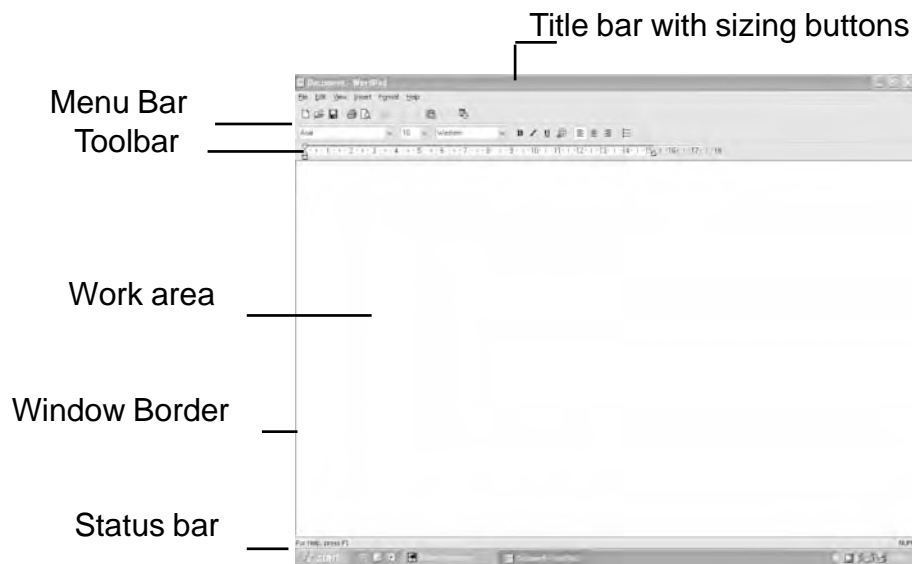
Fig 1.48 WordPad window

As you have already learnt, the WordPad window has a title bar, menu bar, toolbar, work area and a status bar.

A small vertical blinking line appears at the top left corner of the work area. This is the **Cursor**. It indicates your current position on the screen. Some users refer the cursor as the insertion point because it shows, on the screen, where the next text you type will be inserted. Use the keyboard to type in the text. Note that as you type in the text, the cursor moves. When you reach the end of a line, WordPad automatically moves the cursor to the beginning of the next line. This feature is called **Word wrap**. The Enter key on the keyboard is used to start a new paragraph to enter short line or a blank line. Fig 1.49 shows the WordPad window with some sample text.

You do not have to press the Enter keywhen you reach the right margin Do not think that the appearance in the screen will be the appearance of the output. If you want to set the margin that can be done with **Page Setup** of the **File** menu. You can use the following key or key combinations for editing the text.
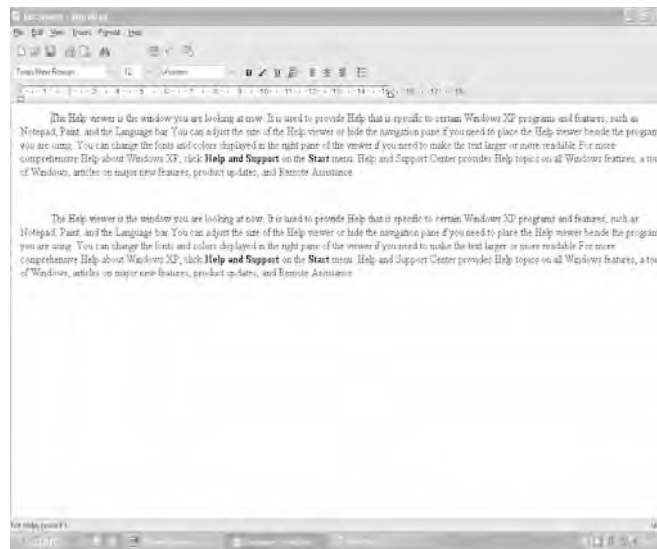
Fig 1.49 WordPad window with some text

Some useful Editing keys are given below.
Keys for Moving the Cursor through Text

Key	Where It Moves the Cursor
→	One character to the right
←	One character to the left
↑	Up one line
↓	Down one line
Home	Beginning of the line
End	End of line
Ctrl + Home	Top of document
Ctrl + End	End of document
Page Up (PgUp)	Up a page (or screen)
Page Down (PgDn)	Down a page (or screen)
Ctrl + ←	One word to the left
Ctrl + →	One word to the right
Ctrl + ↑	Up one paragraph
Ctrl + ↓	Down one paragraph
Ctrl + Page Up (PgUp)	To top of previous page
Ctrl + Page Down (PgDn)	To top of next page

After you have finished typing in the text, you can correct it, add or delete text. To do so, first move the cursor to the place where you want to edit, using the arrow keys on the keyboard. You can also use the mouse to move the cursor. To do so, place the mouse pointer at the place and click.

Once you have moved the cursor, you can delete text using the Backspace and Delete keys on the keyboard. Backspace key deletes the character before the cursor and Delete key deletes the character after the cursor. If you are in insert mode,you can insert new text by simply typing it. If you are in overwrite mode the text you enter will overwrite the existing text (if any).

Pressing Insert key will take you to either of the modes. If you are in insert mode, pressing the Insert key will take you to the overwrite mode and vice versa. Note that when you type in new text, the existing text moves to the right, if you are in Insert mode.

To close WordPad, click on the Close button on the title bar or select **Exit** from the **File** menu.

1.17 Working with Multiple Applications

When you are using multiple applications, it will be very time consuming if you have to close one application before starting the next one. Moreover, transferring information from another application is very difficult if not impossible. For example, in MS-DOS, a file created using a word processor cannot contain a graph created using a spreadsheet program. Windows overcomes this problem by allowing the user to work on multiple applications at the same time. In Windows, a WordPad file can contain data or a graph created using Excel, a picture created using Paint and so on.

1.17.1 Starting Multiple Applications

Starting multiple applications is very simple. First start one application. The application appears on the screen in a window. At the same time,

a button with the name of the application appears on the taskbar. Now, start the second application.

Several things happen –

❖ the window of the second application appears on the screen overlapping the first window,

❖ the button of the second application appears on the taskbar,

❖ the title bar of the first application and its button on the taskbar become dim.

You can start more applications in the same way. Fig 1.50 (a), (b), and (c) will help you understand this better.
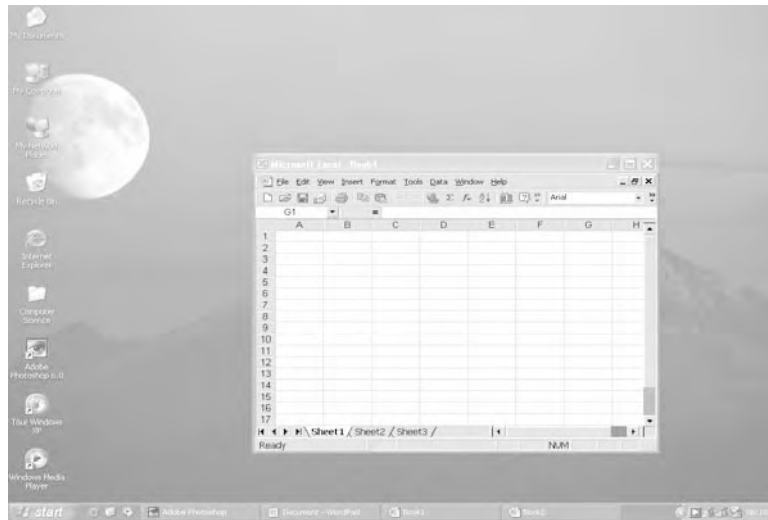
Fig 1.50 (a) Desktop with Excel application.
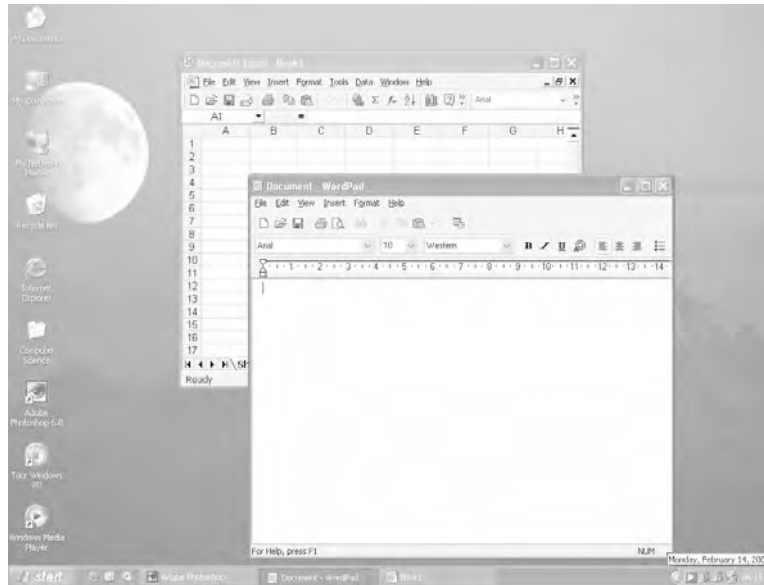
Fig 1.50 (b) Desktop as it appears after Word is also started.
Note the dimmed title bar and button of the Excel application.
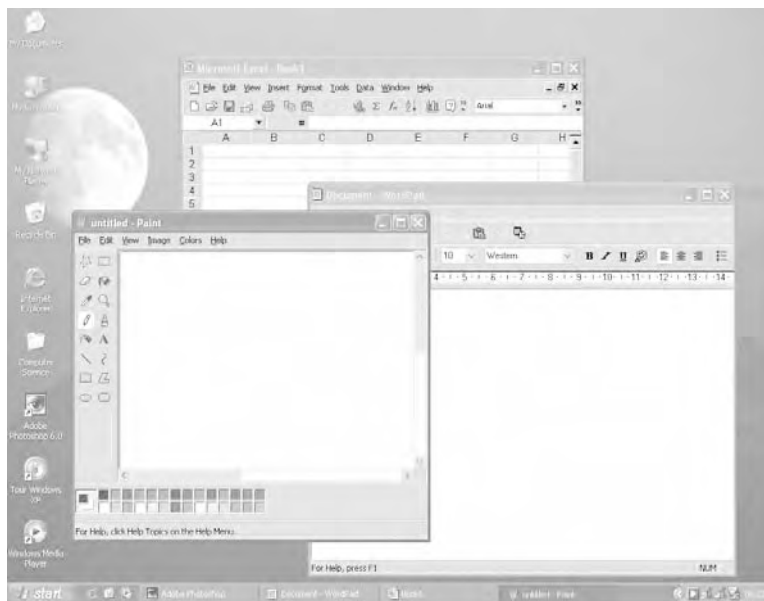
1.50 (c) Desktop with 3 application windows

1.17.2 Switching between Multiple Applications

The buttons on the taskbar are used to switch between the different applications. Remember that every time you start an application, its button appears on the taskbar. The button of the application you are currently using is highlighted and its window is called the **Active Window**. In Fig 1.50(c), **Paint** is the active window. To switch to another application, click on any part of that application's window that is visible. If no part of the window is visible, click the button of the application on the taskbar. The application window is moved in front of all the other windows and its button is highlighted. Fig 1.51 shows WordPad as the active window. Windows Explorer (Which you are going to learn shortly) enables you to create only one button per application. If you click on the button, it will show you a list, from which you can select any one of them.

1.17.3 Transferring Information between Different Applcations

Windows allows you to transfer data between the different applications you are running simultaneously. To do this, Windows uses a temporary storage location called the **Clipboard.** You can use the clipboard to store any kind of data. You can store text, pictures, numbers, group of files and so on.
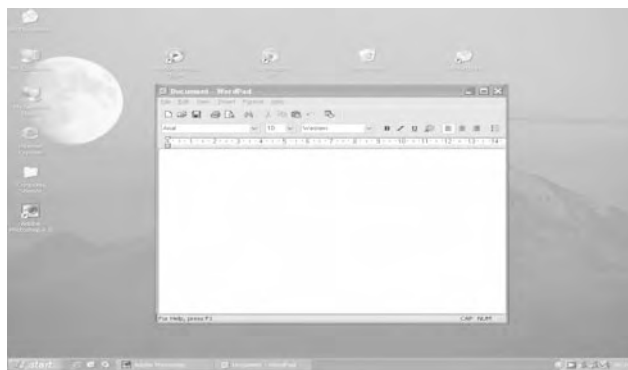
Fig 1.51 WordPad window is the Active Window

60

The information to be transferred is first copied from the source application to the Clipboard and from there to the destination application. Windows also gives the option of either copying or moving data. The difference between copying and moving data is that moving removes the data from the source location and places them in the destination location. Copying leaves the source data untouched and makes a new copy in the destination location.

Let us understand this better with an example. Suppose, you have drawn a picture in Paint and want to include it in a document created using WordPad. To do so, first start both the applications.

You may recall that windows allows multiple applications to be started at the same time. However, you have to switch between the Applications by activating the application of your choice.
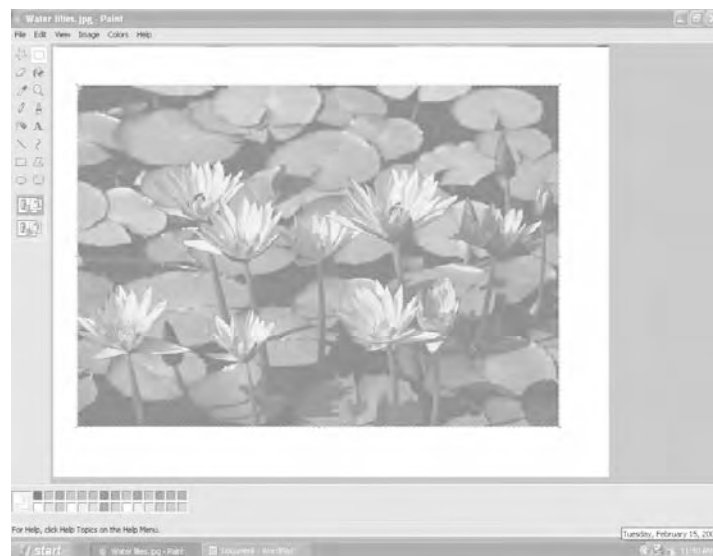
Fig 1.52 Desktop with Paint and WordPad

Click on the Paint window to make it active. Use the **Select** tool to mark the picture you want to move or copy.

Fig 1.53 Paint with the picture selected

Click on the **Edit** menu and select **Copy** or press**(Ctrl+c)**. (If you want to move the picture select **Cut or** press **(Ctrl+x**)
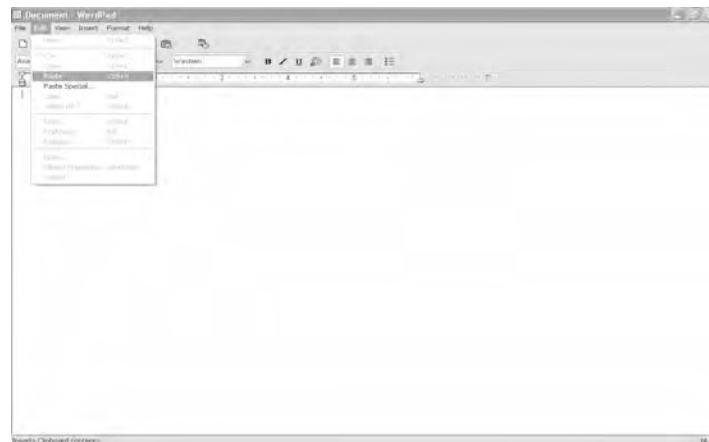
Fig 1.54 Select Edit, Copy

Click on the WordPad window to make it active. In the WordPad window, click on the Edit menu and select **Paste (Ctrl+v).**

Note: Moving and copying will be dealt elaborately later.
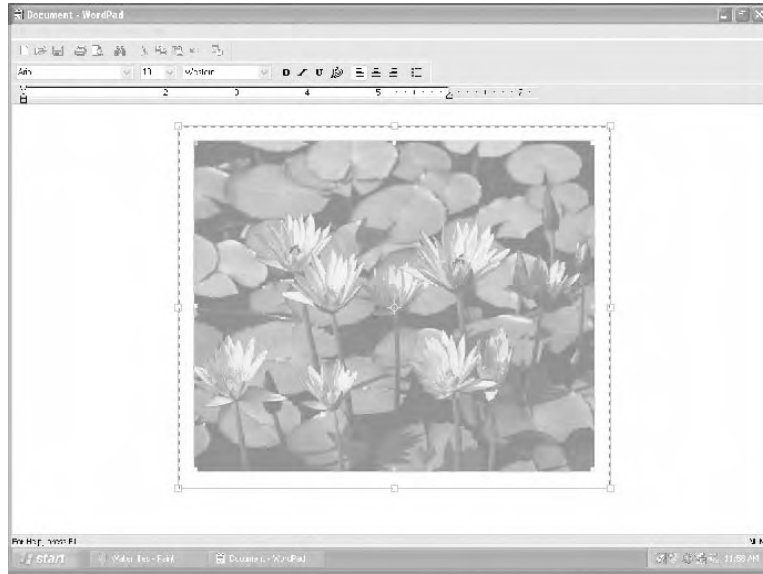
Fig 1.55 Click on Edit; Paste in the WordPad

The desktop will look as shown in Fig 1.56. Note that the picture in the
Paint window remains untouched.
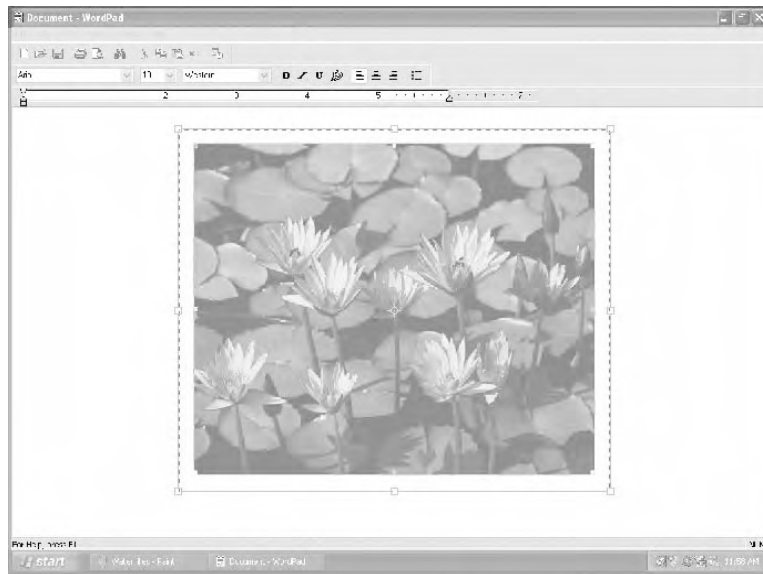
Fig 1.56 WordPad after pasting the picture.

In the same manner, you can transfer data between any two Windows-based applications.

There is, however, one important point that you should remember while using the clipboard. At any time, the Clipboard can hold only one set of data. When you copy or move a file or folder to the Clipboard, it overwrites whatever was stored there earlier.

In addition to clipboard, windows XP provides with ClipBook. Without any effort on your part, now, you can store 24 different items in the ClipBook and you can paste them one by one. For more information, click Start and click Run on the Start menu, enter clipbrd.exe in the text box and press enter. In the ensuing ClipBook Viewer, click Help and then click contents. After going through the help you can do whatever you like with ClipBook

Summary

❖ Files can be of two types - Application and Document files. Applications are used to create data files.

❖ Command Promot option on the start menu allows you to use DOS commands and run DOS-based programs.

❖ The Clock on the taskbar is used to change the date and time.

❖ The Calculator is like an ordinary calculator. WordPad is a simple word processor that is used to enter and store text. Paint is used to draw and colour pictures.

❖ Windows allows you to use multiple applications at the same time

❖ You can switch between applications using the buttons on the taskbar.

❖ You can also transfer data between two applications.

❖ The Clipboard is a temporary storage for data being copied or moved.

Exercises

I.   Fill in the blanks
1.   _____ are used to create data files.
2.   The _____ option allows you to use DOS commands.
3.   The _____ on the taskbar allow you to switch   between
        applications.
4.   Windows uses the _____ to store data being moved or
        copied
5.   Every time you start an application, a _____ appears on
        the taskbar.
6.   The two modes of Calculator are _____ and _____
7.   You can colour your pictures using the _____   _____
        in Paint.
8.   In Paint the _____tool is used to mark the picture to be copied
        or moved.
9.   In WordPad, the_____ key is used to delete the character
        after the cursor.
10.   You can close the WordPad application by clicking on Exit in
        the _____ menu.

II.   State Whether the following are True or False

1.   You cannot use DOS-based files in Windows.
2.   The Calculator is used for performing arithmetical calculations.
3.   You can draw using WordPad.
4.   You can use WordPad and Paint at the same time.
5.   You cannot move a picture painted in Paint to a WordPad
        document.
6.   While working with multiple applications, the button of the active
        application is highlighted.
7.   When you are working with multiple applications, you can close
        them only in the sequence in which you opened them.
8.   The Date and Time Properties dialog box only shows the
        calendar of the current month.
9.   In Windows, you can start a maximum of 10 applications at any
        time.

III.   Answer the following

1.   Explain with an example how to start multiple applications ?
2.   How do you switch between multiple applications ?
3.   What is the Clipboard? How is it used ? Explain.
4.   What are the two different types of files ?
5.   How does the computer display the correct time? How can you change time ?
6.   What is Paint? Describe briefly the different parts of the Paint window.
7.   Where is the MS-DOS Prompt available? How do you use it ?
8.   Where is the Calculator available? How do you use it? Explain briefly.
9.   What is WordPad? How do you start WordPad ?
10.   Describe briefly how to edit text entered in WordPad .

CHAPTER 2

WINDOWS EXPLORER

2.1 Files

You have learnt earlier that you can store a lot of information in your computer. Most computers in the market today have hard disks with capacities of several gigabytes. But how is all this information stored internally? And more importantly, how do you find what you need from all the information that is stored?

Let us answer these questions one by one. All information in computers is stored in Files. Every file has a unique name that helps you to identify it. A file name is made up of two components:

i)   Main Component
ii)   Extension

Main Component: The first part of the file name is the main component. This part precedes the dot and is also called the primary name. This is the name given to the file by the user. The dot (or full stop) separates the main component from the extension. The main component can contain alphabets, numbers, spaces and other characters like @, $, !, {, (,), [,] . However, there are a few characters that a file name cannot contain. They are: \, /,* , ?, ", <, >. Comma and full stop are not included in the set.

Extension: This is the second part of the file name. That is, the part that succeeds the decimal point is called the extension or the secondary name. The extension is used to identify the type of the file and is normally up to three to four characters long. When a file is created using an application, the extension is automatically added to the file's main component by the application itself. Some examples of file extensions are .DOC, .BAS, .XLS and .java.

The file name, including the extension, can be a maximum of 255 characters long. Though you can assign any fancy name to your file, it is always better to use a name that reminds you something about the contents of the file. The aim of naming a file is to retrieve the contents easily. If you assign a fancy name to a file then you may probably not be able to associate the contents with the name of the file. For example, you have written a letter to your friend Ashok. You can call this file **AAA** or **A8124343**, but **Ashok-letter** would be a better choice. This concept is often refered to as the nameing convention.

What is the main use of the (file) extension? When you click on a document icon, this action not only opens the document but also an application, with which it was created. But, how does the application know that its services are required? File extension is the key to solve the problem. The extension of the file name simply announces the format in which the data in the file is stored. Based on this, suitable application opens the file. Normally the file extension is hidden. If you like, you can make it visible. But, it is a good practice to leave it hidden or else you may try to rename it. Renaming the extension, may lead to dangerous consequences and you may not be able to open the document.

It is to be noted that MS-DOS, another operating system, follows a different set of rules for assigning a name to a file. The main component of files created or used on DOS-based computers can have a maximum of eight characters and cannot contain spaces. The extension should not exceed more than three characters in DOS.

2.2 Data Organization

In a computer having a 40 GB hard disk or an 80 GB hard disk, you can store several thousand files. But in these cases, finding one file will be very difficult. You will have to go through all the file names one by one till you locate the file that you need. This is like looking for a book in a library in which the books have not been arranged in any order. Many people never bother to store their files properly.

Windows XP (and DOS) overcomes this problem by using **Folders** (DOS calls them Directories). A folder is nothing but a collection of related files or subfolders. Let us understand this with an example. Consider an organization. Its office will have hundreds of papers relating to products, customers, suppliers, personnel, finance and accounts, and so on. Normally, these papers are filed into different folders and stored in a filing cabinet. Labels on the folders and the cabinets make it easier to find what you need. So, when a person wants some information about a supplier named Shah and Co., he has to look only in the cabinet marked suppliers and search for the folder marked Shah and Co. In the same way, Windows XP allows you to organize the files on your disk by grouping them into folders.

2.3   Windows Explorer

Windows Explorer is a program that helps you to manage your files and folders. To start Windows Explorer, click on **Start → All Programs → Accessories → Windows Explorer**
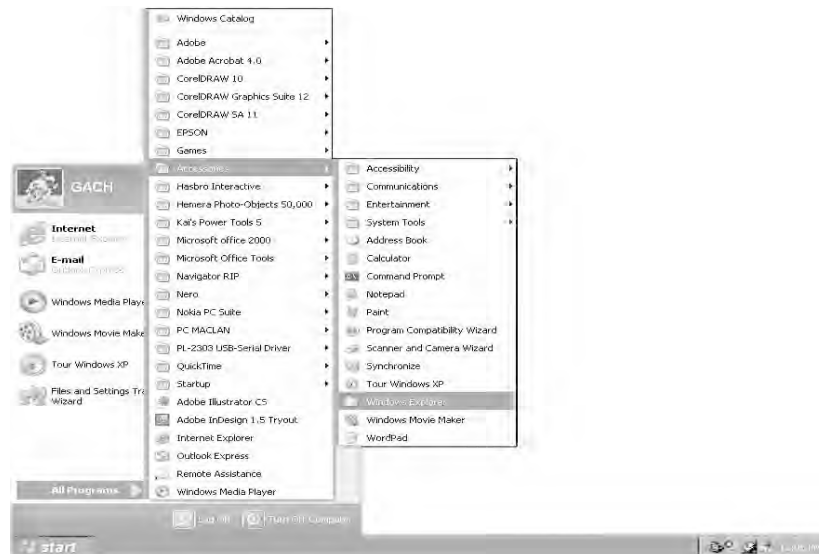
Fig 2.1 Starting Windows Explorer

69

You can also start **Windows Explorer** by right clicking on the **Start** button and then selecting **Explore** from the short cut menu.
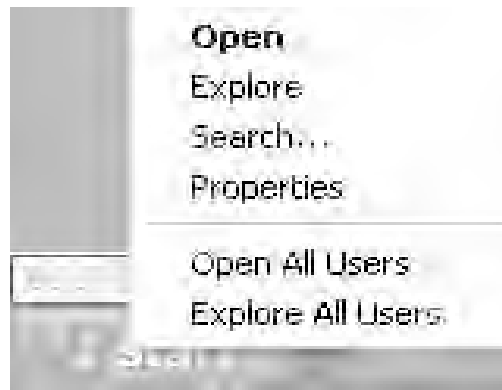
Fig 2.2 An alternate way of starting Windows Explorer

The Explorer window opens on the screen as shown in Fig 2.3
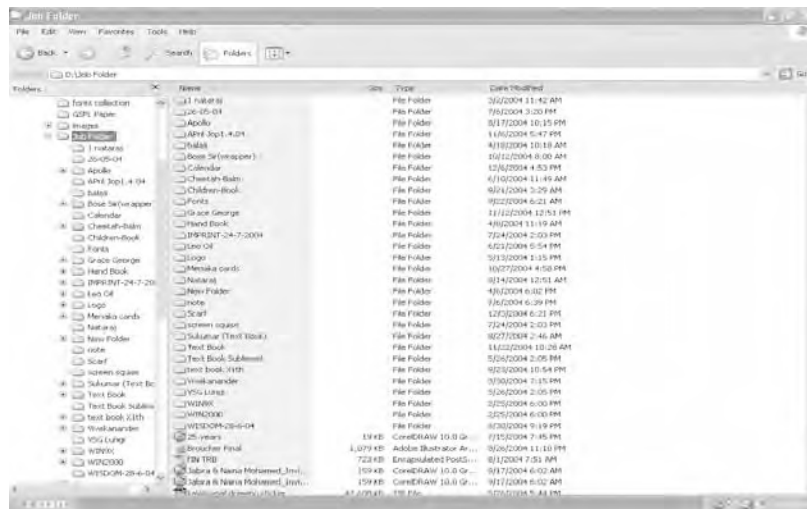
Fig 2.3 Windows Explorer window in Folder Bar

Like any other window, the Explorer window also contains a title bar, a menu bar and a toolbar. But unlike the other windows, the Explorer window is selfless. It never shows its own name in the title bar; instead,

it shows the name of the current folder whose contents are visible in the right pane (Main Pane). The only time that it shows its name is when related files are grouped in the taskbar. If you open up to four folders, separate buttons are created on the Task Bar, but if you open the fifth folder or any application, folders are grouped and only one button is created whose name becomes Windows Explorer. You can see the number of folders preceding the name Windows Explorer. Again a single button is created for each application program. This action not only preserves the space in the task bar but also if you right click on it, it will allow you to use **Close Group**. This will close all the folders at one stroke. Even the audio, video files are treated as folders under this context. If you want to open / close one of the grouped folders, you can left click the button. This action displays all the folders within the button. You can either open the folder or close it. If the number of buttons exceeds four then normally a single button is created for each application. Suppose you want to open five **Word** applications, five separate buttons will not be created in the task bar, only one button is created in the task bar. Below the toolbar is the display area. As you can see, this area is divided into two panes. The left pane displays either the **Explorer Bar** or the **Folder Bar** and the right pane always displays the contents of the currently selected folder in the left pane. If you click the **Search** button from the toolbar, the left pane neither shows **Explorer Bar** nor **Folder Bar**, instead it shows the **Search companion**. If you just double click a folder, you will see an **Explorer Bar** in the left pane. If you right click on the same folder and select **explore** from the ensuing shortcut menu, you will see the **Folder Bar** in the left pane. You right click a folder and choose **open** from the ensuing short cut menu, you will see only the **Explorer Bar** in the left pane. But you can switch over from one to the other just by clicking folder icon on the toolbar.

2.4 Working with Folders

A small yellow icon represents each folder. Note that the disk drives on the computer are also treated as folders. A plus sign to the left of the folder icon in **Folder Bar** indicates the presence of subfolders within

this folder. You can see + or – sign only in the **Folder Bar**. You can click on the plus sign to display a list of the subfolders. When you do this, the plus sign changes to a minus sign. Clicking on the minus sign will hide the details. Fig 2.4(a) and 2.4(b) show the folder representing the hard disk C: in the expanded and collapsed forms.

Fig 2.4(a) Folder Bar   Fig 2.4(b) Folder Bar Expanded
                                                     form c:> Collapsed form in Folder Bar

If there is no plus sign to the left of a folder icon, it means that the folder does not have any subfolder. Scroll bars in this part of the window allow you to browse through the list of folders.

To see the contents of a folder you have to select the folder. To do so, just click the folder. The yellow file icon next to the folder changes to look like an open folder. The selected folder is highlighted and its contents are displayed in the right pane of the Explorer window. In Fig 2.5 the Windows folder has been selected in the left pane and its contents are displayed in the right pane.

2.4.1 The Explorer Bar

Old habits die hard!. What you have seen is the only facility available in Windows 98. But Windows XP provides additional facility in the name of **Explorer Bar.** The **Explorer Bar** with its sophisticated, more useful tools lets you navigate (travel) and to work with icons contained within the current folder. This bar is divided into three categories. They are :-

(1) File and Folder Task
(2) Other Places
(3) Details

File and Folder Tasks: As the name suggests **File and Folder tasks** allows you to work on files and folders, for example, by clicking **Make a new folder** under **File and Folder Tasks**, you can create a new folder. **Share this Folder** option shares the chosen files among the group users. It provides web facilities also. **File and Folder Tasks** is context sensitive (depends on what folder you choose). Whatever you find here is not visible in the **Folder Bar**.

Other Places: If you want to switch over to other folders, you can select options in other places.

Details: Details provides some detail about the open folder.
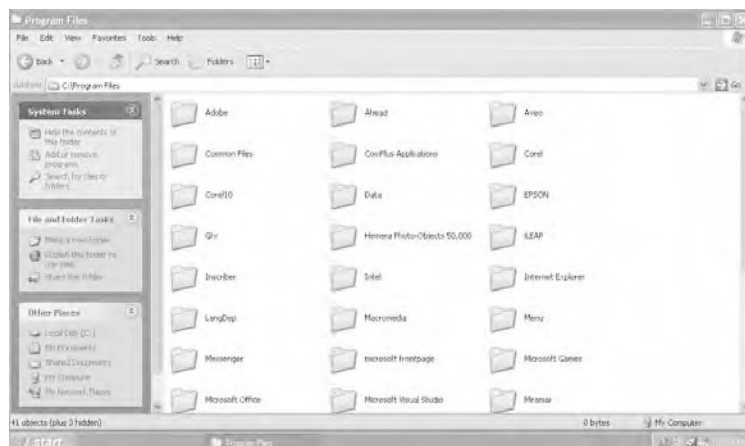
Fig 2.5 Windows Explorer Window in Explorer Bar

73

You can expand or collapse any of the categories by clicking any where in the caption (including Show / Hide button two down / up arrows). You can also navigate to any folder by means of the **Address** bar. Click the down arrow situated in the right most area of the Address text box. From the drop down list box you can navigate to any folder from your current position.

2.5   Changing the View

Windows Explorer allows you to change the way in which information is displayed in the right pane. You can display the list of files and folders using any of the following views. You click view in the menu bar or click view button in the explorer's toolbar. The view affects only how the information is displayed not what is displayed. Icons work in the same manner, whatever be the view. For example Double-clicking an Icon in any view will open it.

The Icons View - The **Icons View** shows each file or folder's icon and its name. This view will not provide any more detail.
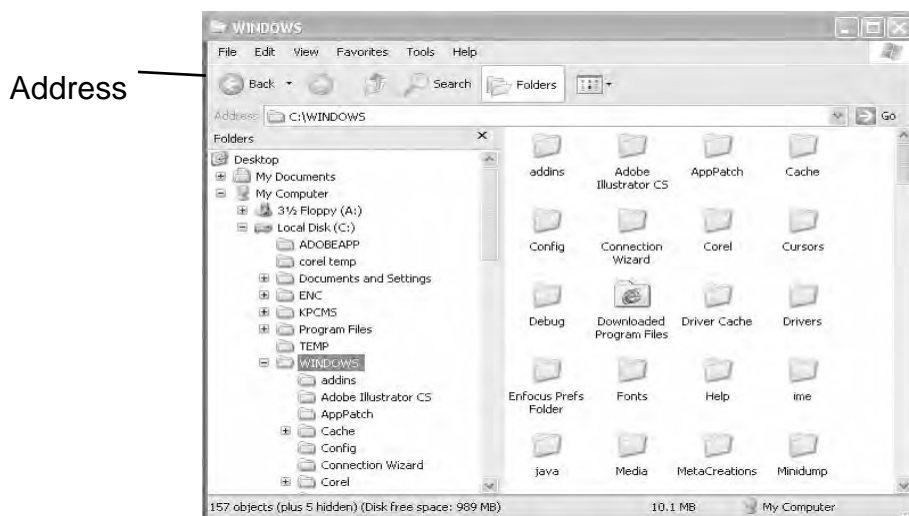
Fig 2.6 Contents of windows folder with Icons View

The Tiles View - In this view, the icons are a little bigger. In addition to icon and its name, this view provides some more additional information for some icons.

74
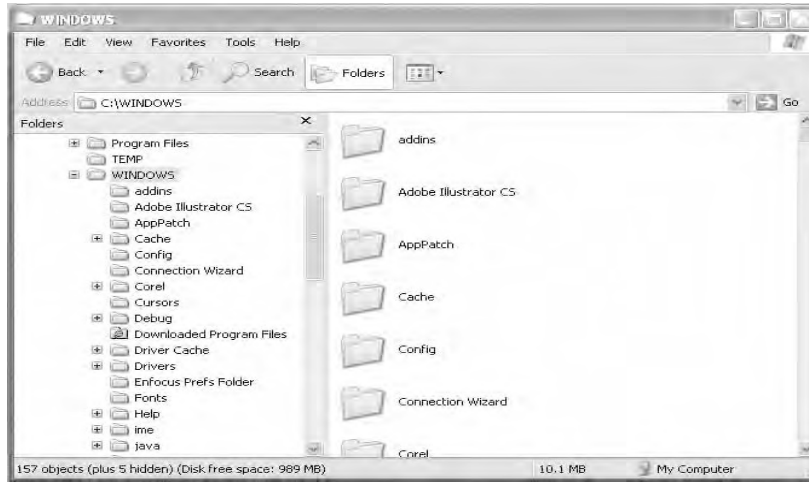
Fig 2.7(a) Icons with Tiles View

The Thumbnails View - The **Thumbnails View** works well in folders that contain pictures. Documents that contain pictures are not shown as icons but they are displayed as minimized pictures.
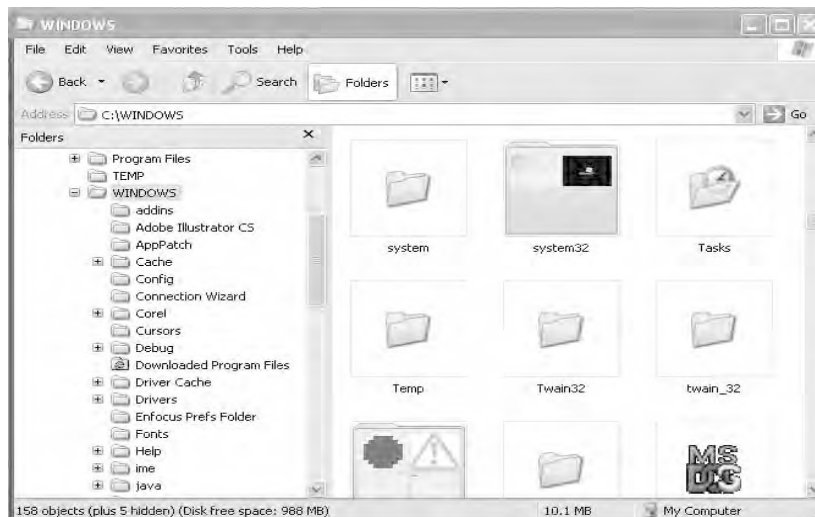
Fig 2.7 (b) Thumbnails View

The Filmstrip View - This view is available only in folders that contain pictures such as **My Pictures Folder.** This view is similar to the

75

Thumbnails View. When you click or point to a picture, an enlarged copy of the picture appears in preview area.
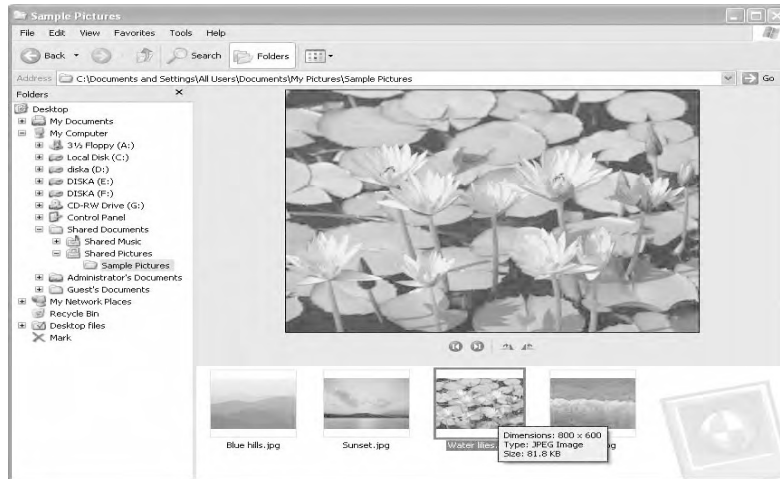
Fig 2.7(c) Film Strip View

List View - This view retains the small icons but displays the files and folders one below the other in columns.
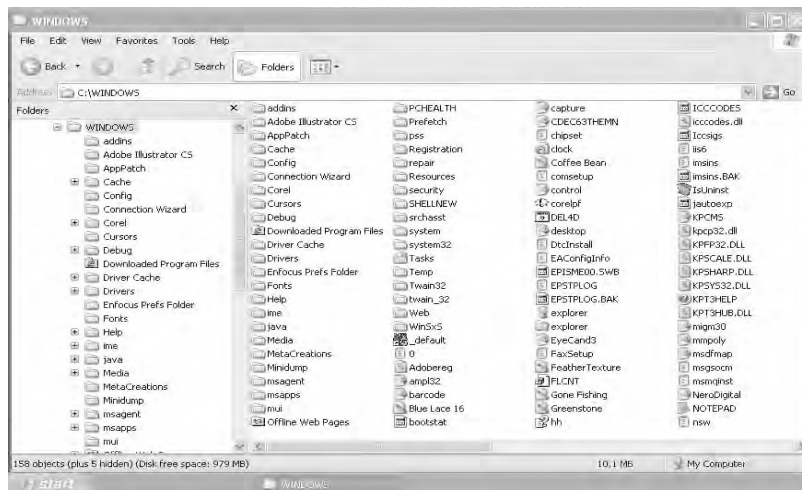
Fig 2.7(d) List View

Details - This view displays details like file size, type, last modified

76

date and time along with file names and small icons. If you cannot find all the information, use the scroll bar available in this view.
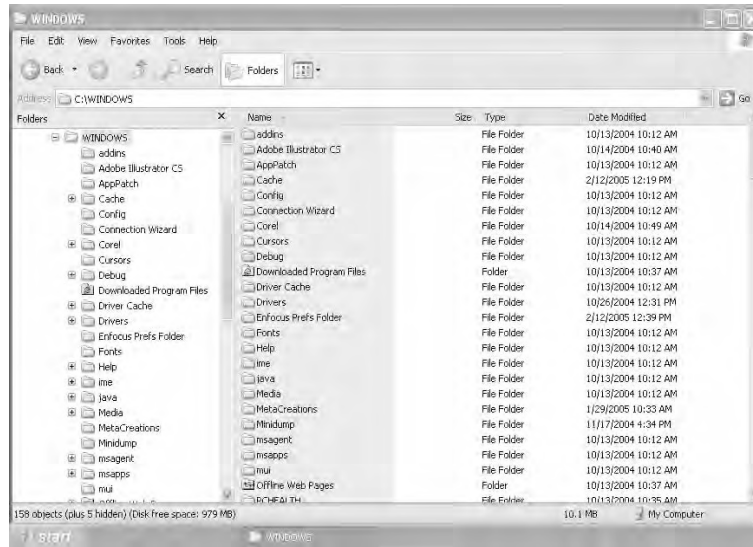
Fig 2.7(e) Details View

Note- All the View figures have Folders Bar in the left pane. You can have Explorer Bar in the left pane also.

You may wonder why we need several views. Views will help you in finding a forgotten file. You may forget the name of a file, but you may remember, the date of creation or the extension of that file. Then sort, the contents of a folder that you expect your file will be, by date or file type (extension), in Windows Explorer. You can see whether the file is there or not at a glance.

Note: A better way to search a file is by means **Search** option. You will see it later.

2.6 Creating a New Folder

Often, you may want to create a new folder to store some of your files. Creating new folders using **Windows Explorer** is very easy. Already

you have seen a method to create a folder by simply clicking **Make a New Folder** under **File and Folder Task** in the **Explorer Bar**. It will create a new folder in the right pane. You can enter the name that you have chosen for the folder in the highlighted box and then press **ENTER** key ↵. First, select the folder under which you want to create the new folder. Then, right click anywhere in the empty space in the right pane of the **Explorer window**. Click on **New** from the menu shortcut. Select **Folder** from the submenu that appears. You can also obtain the same result from the menu bar by Clicking **File → New → Folder.**
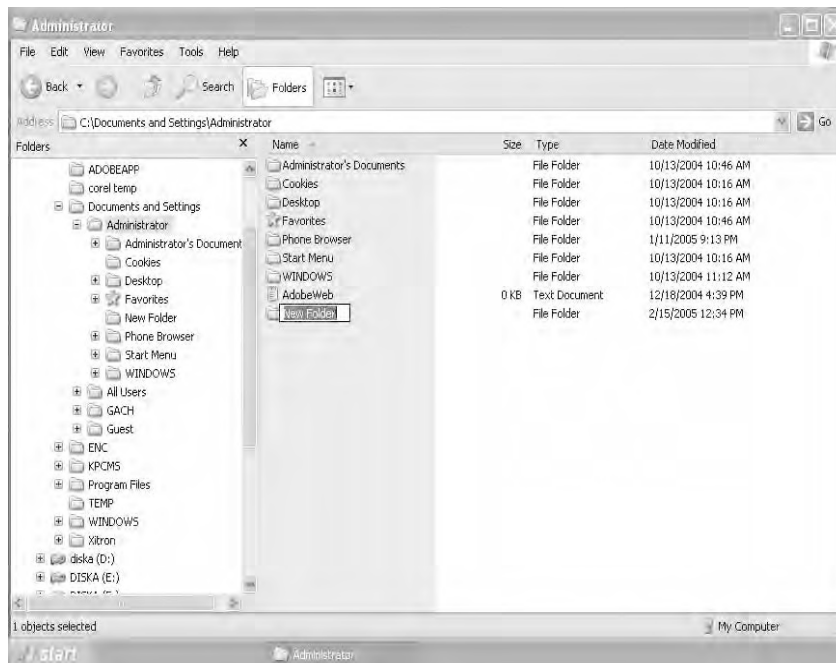
Fig 2.8 Explorer Bar to create new Folder

In the folders bar also you can right click on the empty space on the right pane. Click **New** in the ensuing shortcut menu then click the **Folder** in the submenu.

78
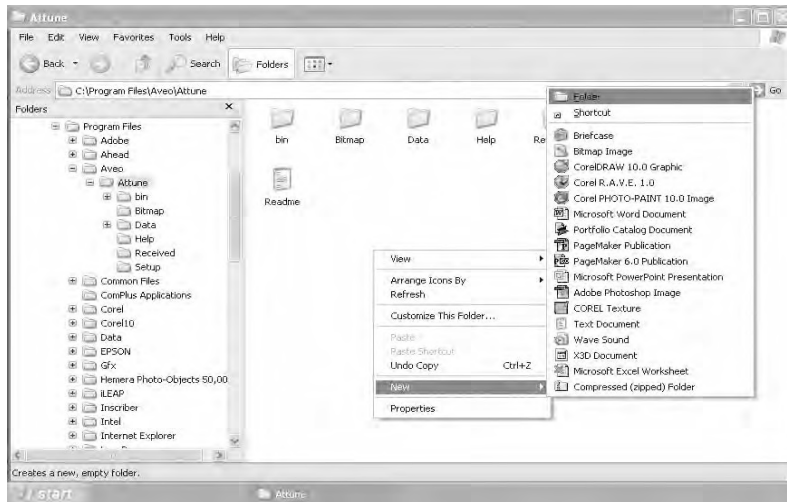
Fig 2.9 (a) Click on New and select Folder

A new folder with the temporary name "*New Folder*" is created as in figure 2.9(b)
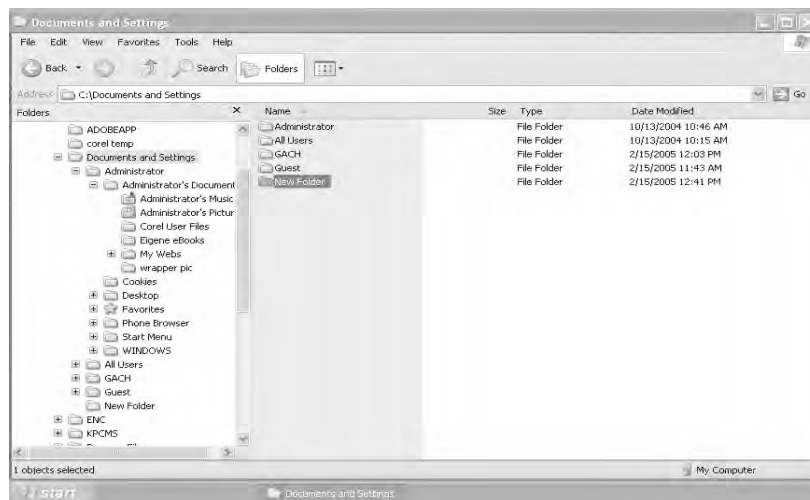
Fig 2.9 (b) " New Folder " is created

Simply type the name you want to give this new folder and press **EN-TER**. Here Test is the chosen name and My Documents is the chosen folder, under which TEST folder is created.

79
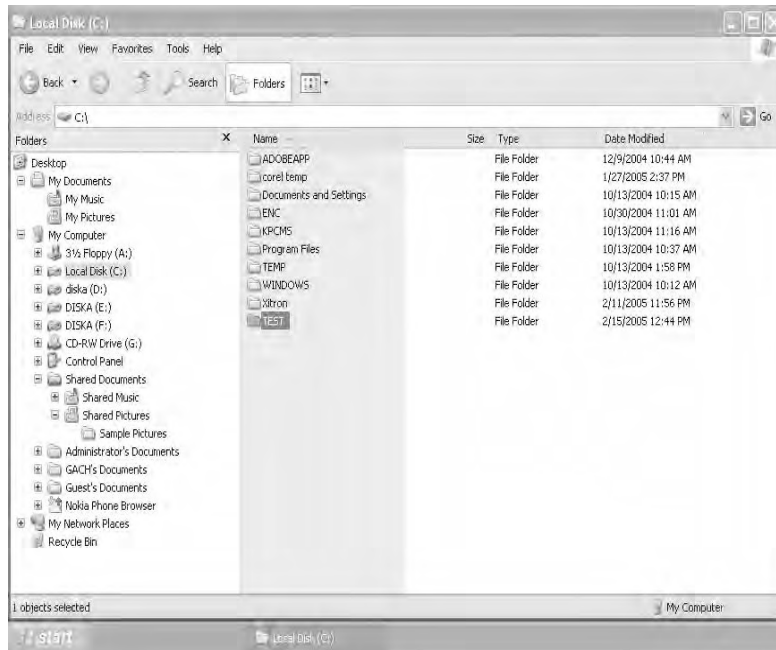
Fig 2.10 A new folder called TEST is created under My Documents.

2.7   Selecting Files and Folders

Windows Explorer allows you to copy, move and delete files and fold-
ers. But, before you can do any of these, you have to select the files or
folders that you want to copy, move or delete. Selecting one file or
folder is very simple. Just click on the file or folder and it gets high-
lighted. If you want to select more than one file or folder, you can do so
in any one of the following ways:

i)   If the files or folders to be selected appear consecutively on the
screen, then, click on the first file or folder. Using the scroll bars (if

necessary), point the mouse pointer to the last file or folder in the list, hold the **Shift** key down and click. Fig 2.11 (a) shows a list of six consecutive files, which have been selected.
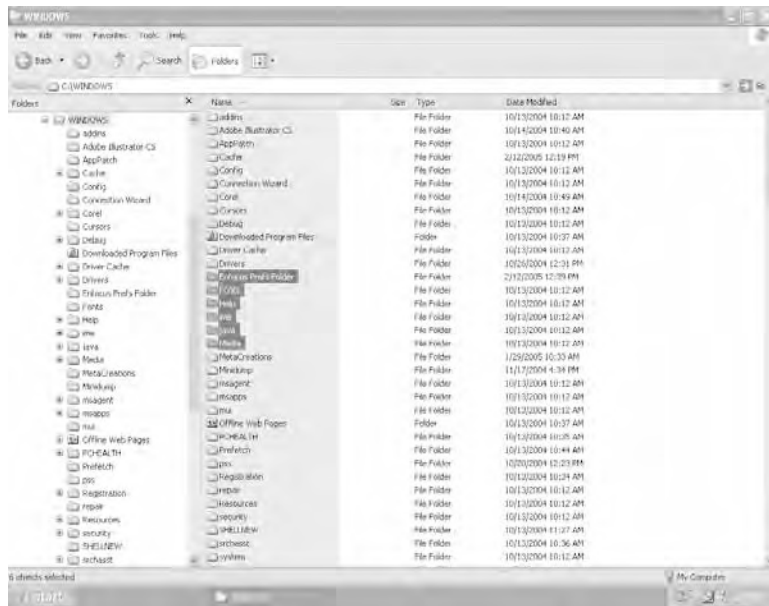
Fig 2.11 (a) Six consecutive files selected

ii)   If the files or folders to be selected are not displayed consecutively, then, click on the first file, move the mouse pointer to the second file to be selected and click while holding the **Ctrl** key down. Repeat the procedure for each of the other files to be selected. Fig 2.11 (b) shows five selected files that are not displayed next to each other. If you select a file wrongly, in order to deselect it from the group, press **Ctrl** and click on the file. Above selection can also be made with Explorer Bar in the left pane.
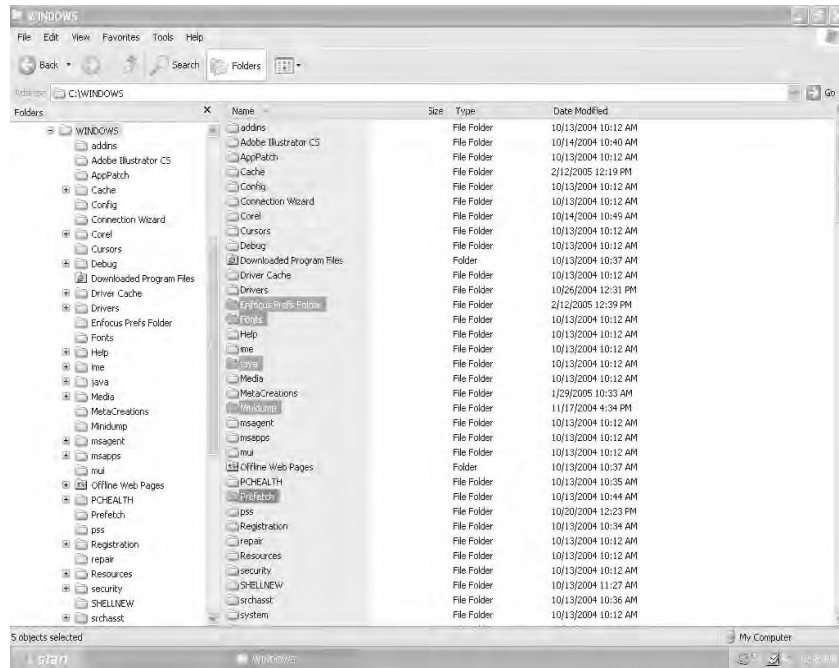
Fig 2.11 (b). Five non-consecutive files selected

2.8   Moving and Copying Files and Folders

Once the files are selected, you can move or copy them using **Cut, Copy** and **Paste** in three different ways.

1)   Click on the Edit menu and make appropriate choice.
2)   Right click on any one of the selected folders or files, in the ensuing short cut menu and make suitable selection.
3)   You can use the keyboard combinations Ctrl+X (Ctrl+x) to cut, Ctrl+C (Ctrl+c) to copy and Ctrl+V (Ctrl+v) to paste.

The difference between copying and moving files is that moving removes the files or folders from the source location and places them in the destination location. Copying leaves the source files or folders untouched and makes a copy in the destination location.

82

The Windows Explorer copies or moves files using the Windows XP Clipboard. A clipboard is a temporary storage area where files or folders are stored before being copied to the new location.

Let us understand copying and moving files with an example.

2.8.1 Moving Files and Folders

You are going to see how to move the files or folders by using Folder Bar. Explorer Bar may be used to move the files with ease. But **Folder bar** is considered first. If the left pane of Explorer Window is not in **Folder Bar** then click **Folders** button on the toolbar. Now you are in **Folder Bar**. If the Folders button in the toolbar is highlighted, you are in Folders Bar, otherwise you are in Explorer Bar.

Consider the folder My Documents. It has 17 files and folders. Suppose you want to move the files **Student, Raj, kumar** and **Exam** from My documents to the folder Test. Select the files as explained above.
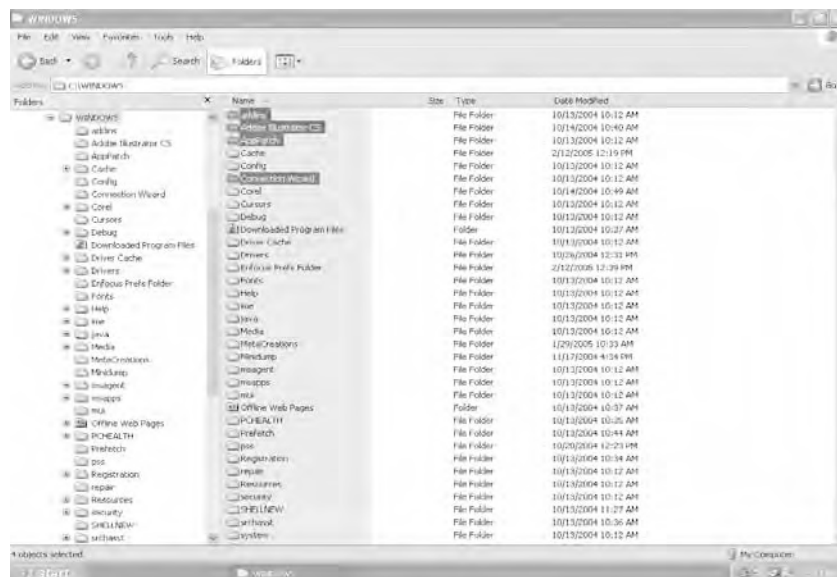
Fig 2.12 Four files selected

Then, **Cut** the items by using anyone of the three methods explained above.

Next, click on the folder or disk drive to which you want to move these files. In this example, click on the folder **Test.**

TEST Folder

Fig 2.13 The Test folder

Now, **Paste** the items by using anyone of the three methods explained above.

Figure 2.13 shows the folder **Test** after the files have been pasted. Remember that they have been removed from the folder **My Documents.**

Now let us see how to move files and folders with Explorer Bar . You can move or copy selected files easily in the **Explorer Bar.** If you are not in **Explorer Bar**, click the F**older** in the toolbar**.** You will be shown the **Explorer Bar** with the following options under **File and Folder**

Tasks.
Move the selected items
Copy the selected items
Publish the selected items on the Web
E-mail the selected items
Delete the selected items

Let us suppose you want to move the selected items. To do so you click on **Move the selected items**. You will be provided with **Move Items** list box; you can browse and select the desired destination. You can even create a new folder by clicking **Make New Folder** button found at the bottom of the **Move Items** list box. Click **Move** button adjacent to **Make New Folder** button to store the selected items in the newly created folder. That is all; you have successfully moved the selected items under TEST. If you want to move a single item, clicking on the item will show you 6 options one among them is **Move this folder** under **File and Folder Tasks.** Follow the same procedure for moving group items.
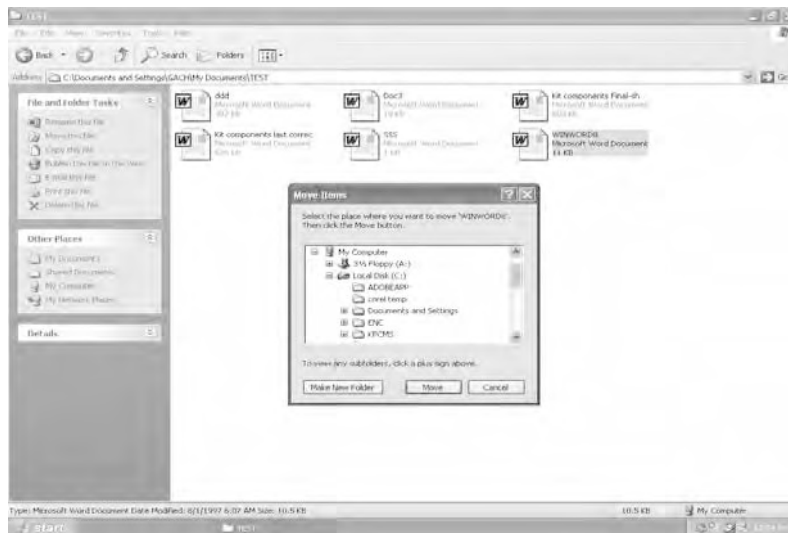
Fig 2.14 Moving Files and Folders with Explorer Bar
You can also move selected items with Edit menu also.

85

From the menu bar, click Edit®Move To Folder as in fig 2.15
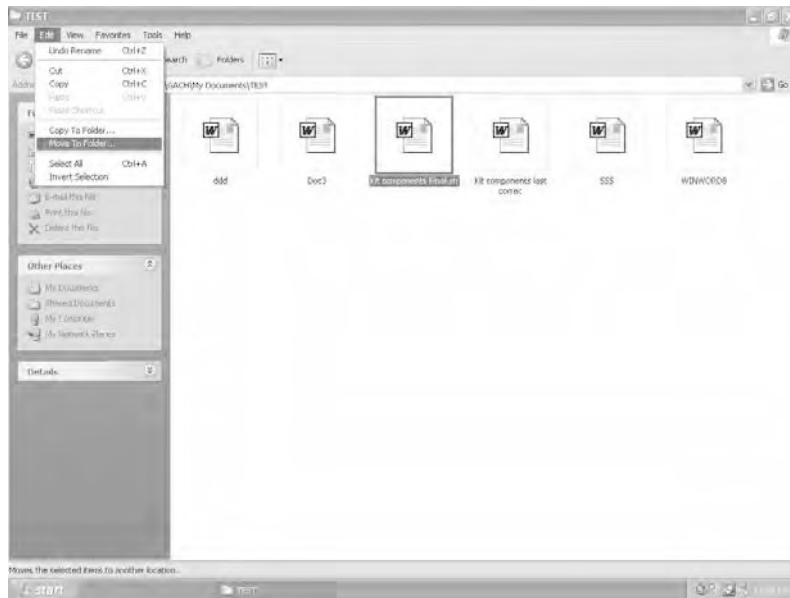
Fig 2.15   Moving through Edit submenu

You will be taken into **Move Items** drop down list box. Then follow the steps explained in the previous paragraph.

2.8.2 Copying Files and Folders

When you copy a file, the original file is left untouched and a fresh copy of the file is placed in the destination location. For example, you want to copy the files **INAUG** and **GACN** from the folder **My Documents** to the folder **TEST.**

Again, first select the files
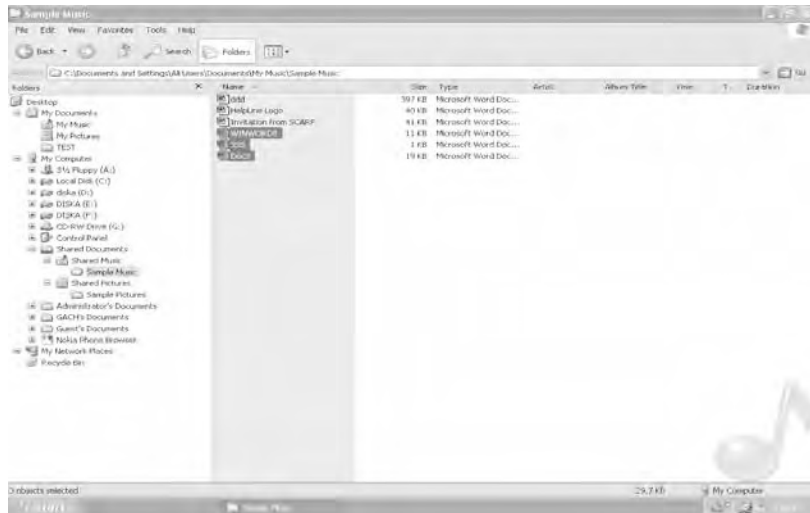
Fig 2.16 Files selected for copying

Then, Copy the items by using anyone of the three methods explained in 2.8

Click on the folder **Test.**
Now, **Paste** the items by using anyone of the three methods explained in 2.8
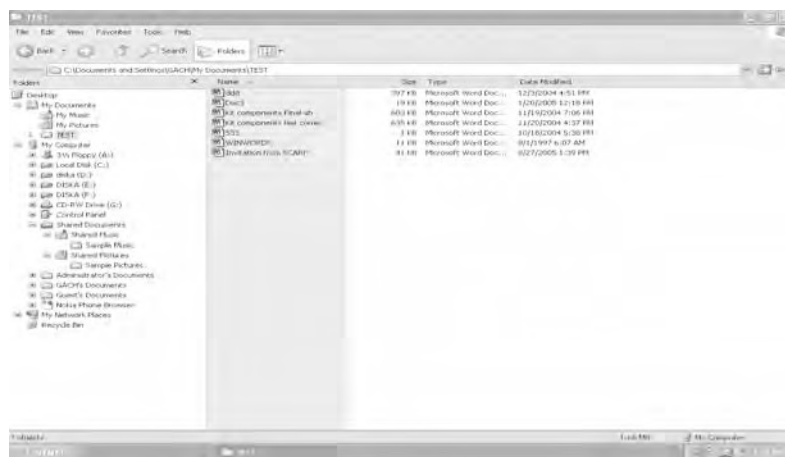
Fig 2.17 The Test Folder after copying

87

When you want to copy files or folders to a disk in A or B drive, you can use the **Send To** option in the pop up menu, which appears when you right click on the file or folder. For example, you want to copy the files **Kumar** and **Raj ,** from the folder **My Documents,** to a floppy in drive A. To do so, first select the file and then, right-click on the selected files.

A shortcut menu as shown in Fig 2.18 appears.
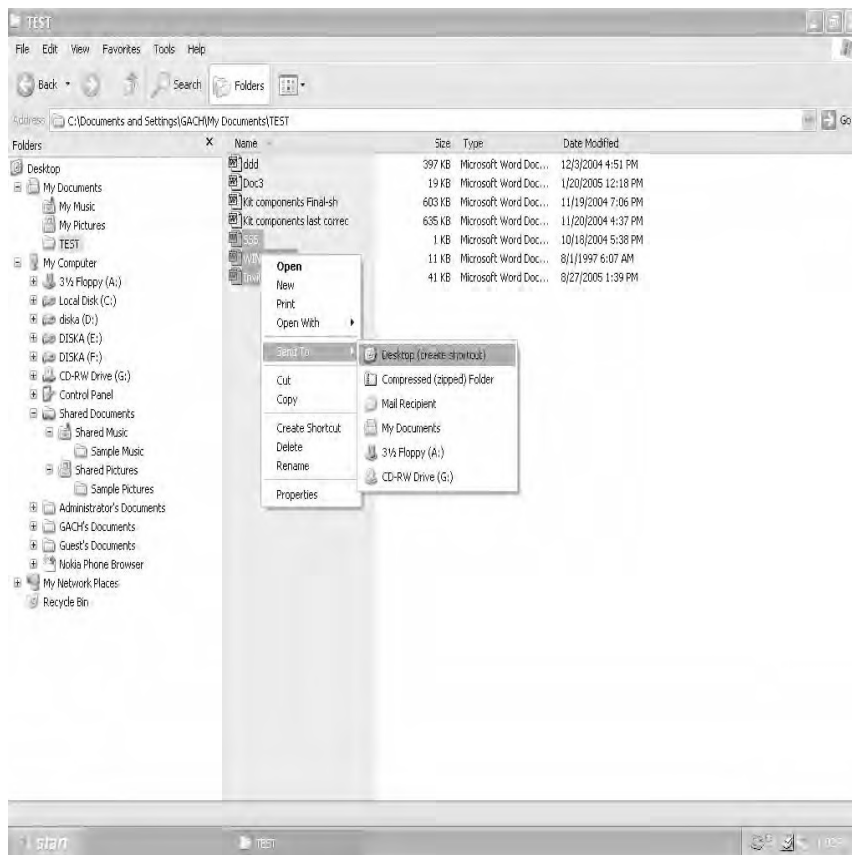
Fig 2.18 Select Send To

Click on **Send To** and 3 ½ **Floppy [A]**
Click on 3 ½ **Floppy [A]** in the left pane to check that the files have been copied.
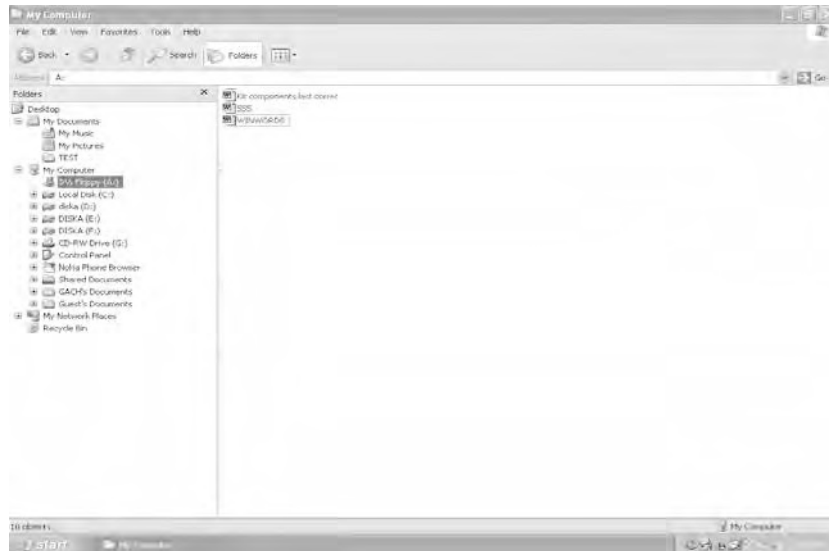
Fig 2.19 Contents of floppy disk in drive A: after copying

Let us see how to copy the files or folders by using Explorer Bar.

You can copy selected files easily in the **Explorer Bar.** If you are not in **Explorer Bar**, click the F**olders** in the toolbar**.** You will be shown the **Explorer Bar** with Five options under **File and Folder Tasks,** if you select more than one item**.** Already you have seen the options under **Moving and Copying Files and Folders**

Let us suppose you want to Copy the selected items you click on **Copy the selected items**, you will be provided **Copy Items** drop down list box (Fig 2.20), you can browse and select the desired destination. As already stated, you can even create a new folder by clicking **Make New Folder** tab found at the bottom of the **Copy Items** drop down list box. Click Copy button adjacent to **Make New Folder** button to store the selected items in the newly created folder. That is all; you have successfully copied the selected items to the desired location. If you want to copy a single item, clicking on the item will show you **Copy this folder** under **File and Folder Tasks.** Follow the same procedure for copying group items.
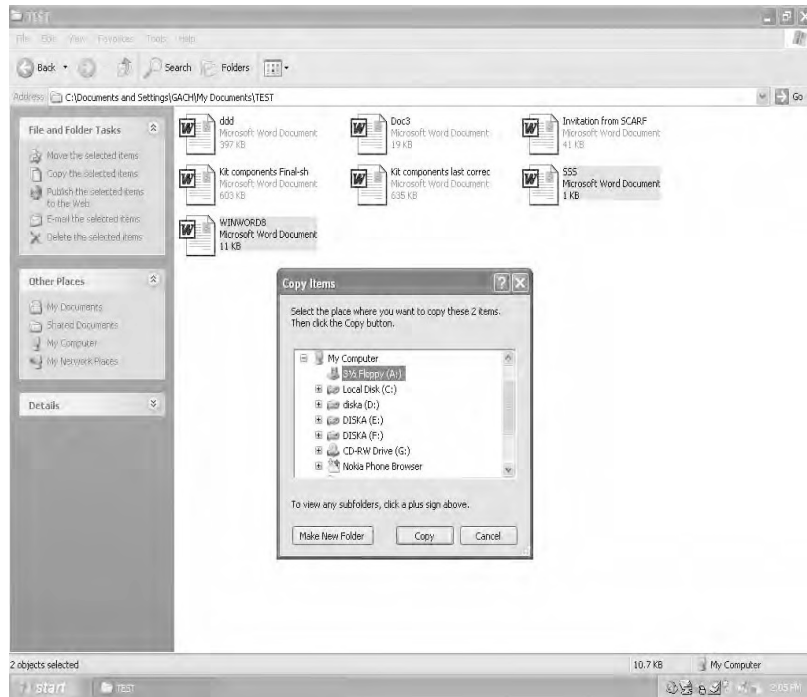
Fig 2.20 Copying Files to 3 ½ Floppy [A] with Explorer Bar

You can also copy the selected items with Edit menu also. Refer Fig 2.21

From the menu bar, click Edit →Copy To Folder…
You will be taken into **Copy Items** drop down list box. Then follow the steps explained in the previous paragraph.

If you want to copy the selected files from the Explorer Bar to 3 **½ Floppy [A],** if you have not inserted the Floppy into the Floppy drive, do it so now. You just click the option **Copy the selected items** under **File and Folder Tasks.** Choose 3 **½ Floppy [A]** from **Copy Items** drop down list box, under My Computer. Click 3 **½ Floppy [A]** then click the copy tab at the bottom of **Copy Items** drop down list box**.** That is all

90

you have copied the selected items into the Floppy disk. If you want to copy only a single file you follow the same procedure.

Note: Remember that the clipboard can hold only one set of items at a time. When you copy or move a file or folder to the clipboard, it over-writes whatever was stored there earlier. If you need to store the items you place in the clipboard permanently, you should make use of clipBook. The clipBook has 127 pages and you can store an item in each page. The clipBook gets items through the clipboard. You can transfer item from the clipboard to ClipBook. The items stored in clipBook can be shared with the users through the Internet. For using clipBook, click Start → Run. In the Run textbox ,enter clipbrd viewer and click OK.You will be taken to clipBook Viewer. Click Help → Contents. In the ensuing help click Related Topics at the end of the help and click Save the contents of the Clipboard to the local clipBook. Follow the instruction in the ensuing Help.

2.8.3 Copying Files to CDs

Coping files to a CD is often referred to as burning the CD, because a laser actually burns the information on to the disk. If you write files on CD you should have CD burner installed, of course you should have blank disks. There are two types of CD burners and two types of blank CDs in the market. CD-R, CD-RW are the two types of CD burners. There are CD-R, CD-RW disks also.

CD-R burner is used to burn data to blank CD-R disk. You can make use of the resultant disks in any computer that has CD drive in it. If it is an audio CD you can use it in any standard stereo.

CD-RW burner is used to burn data to either a blank CD-R or CD-RW disk. The resulting disk can be used only in computers that have a CD drive. The CD-RW disk can be used as an ordinary floppy. You can

add or delete files from it. But CD-R can be written only once.

General Method for Copying to CD
Insert a suitable blank CD into the suitable drive and wait for a few seconds.
In the ensuing dialog box click, open writable CD folder using **Windows Explorer** and click OK.
Here CD-RW Drive is used.

1. If dialog box does not appear on the screen within a few seconds of inserting the blank   disk, open your My Computer folder. Then right –click the drive's icon and choose Open, then follow the previous step.

2. Go to the source folders.

3. Select items you want to copy to the CD. Right click any selected item and choose Send to à CD –RW Drive.

4. Each item to be copied will appear as a temporary file, with black arrows pointing downwards as shown in fig 1.78

5. Check whether all files that you want to copy are there and verify that the data capacity of the combined files is less than the capacity of the disk. Then click write these files to the CD   under CD Writing Tasks in the Explorer bar of the CD's folder Window.

6. In the first page of CD writing wizard, you can enter a new name for CD. It is just like a label to the floppy disk. Delete the date that appears.

7. Wait until the wizard burns the data to the CD. Then click Finish button on the last page of the CD Burning Wizard. That is all.
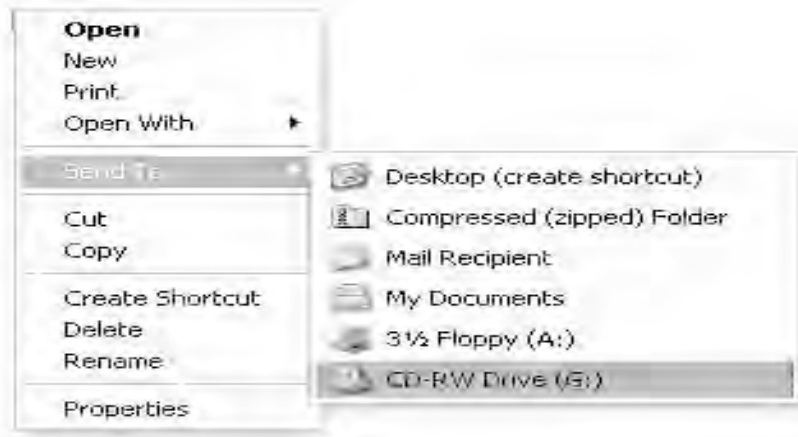
Fig 2.21 Selecting CD-RW Drive

When you click **Send To**, if you have CD-RW drive you will be shown **CD-RW Drive** otherwise the last option will be there. In Fig 2.22 you can see CD Writing Tasks, this category will appear only in the computers which have CD-drive.
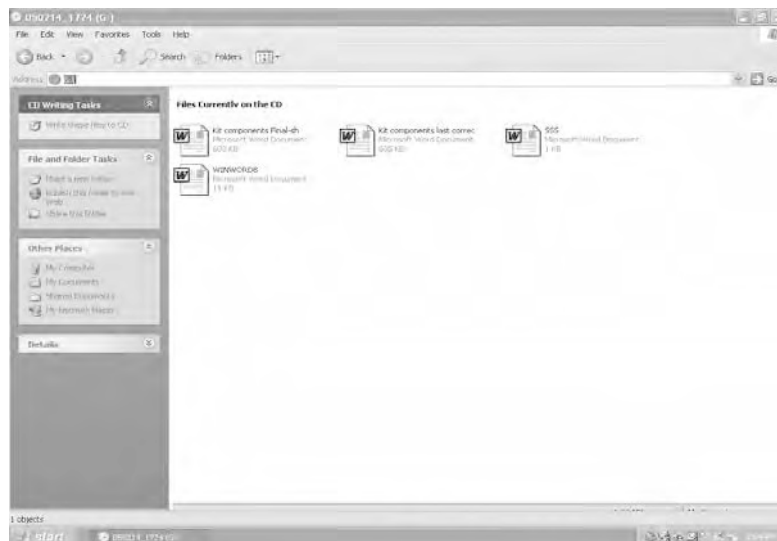
Fig 2.22 Files ready to be written to the CD

93

2.9   Renaming Files and Folders

Normally you Rename only one file or folder. In this case, you can Re-name the file in any one of the following ways.

1) Click the file or folder. When you are in Explorer Bar, you can choose **Rename this folder** from **File and Folder Tasks.** The name of the selected file MANI gets highlighted. Now, type the new name (SHIVA) and press **Enter.** The new name of the file SHIVA appears in the window . Fig 2.23 (a) shows the renaming process.
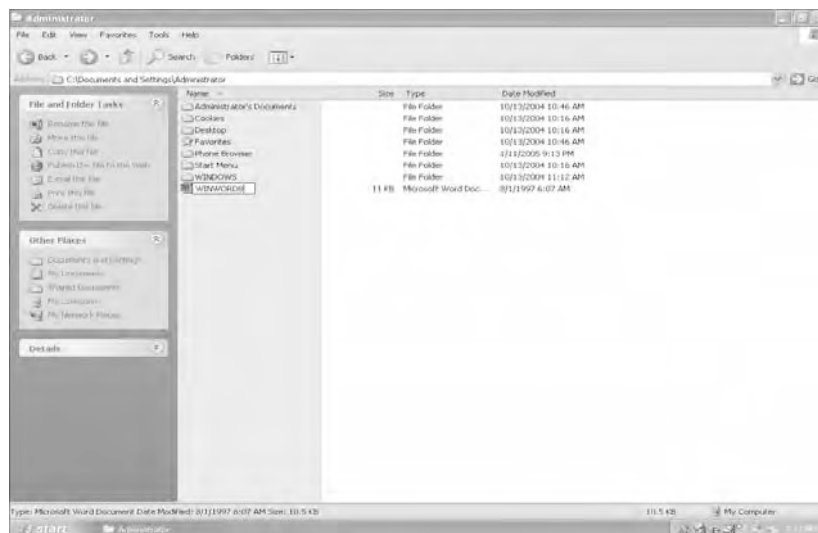
Fig 2.23(a) Single file Renaming option (Explorer Bar)

You can follow the following methods when you are either in Explorer Bar or folder Bar.

 2) To rename a file or folder, right click on the file or folder. Select **Rename** from the   shortcut list, which pops up on the screen. Now change the name as given above.
 3) From the menu bar, click **File → Rename** and rename the file as explained in method 1.

94
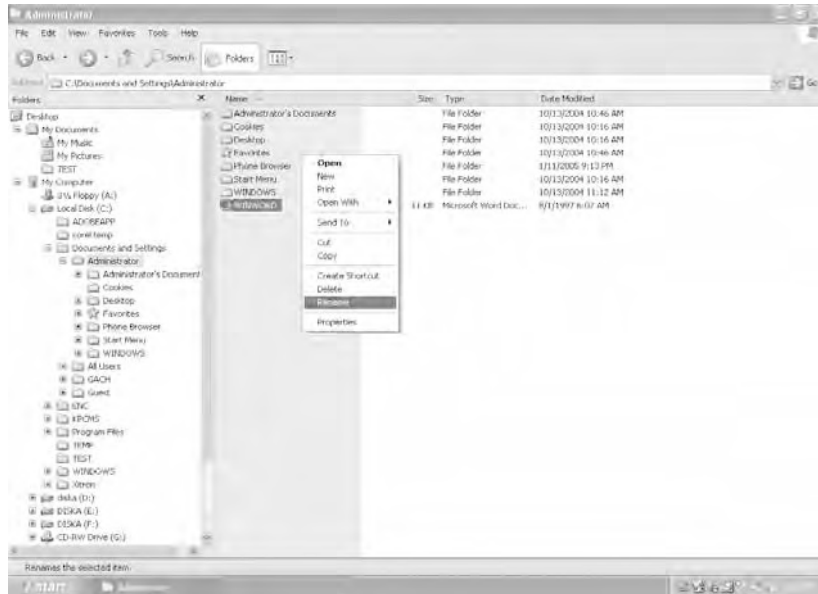
Fig 2.23(b) Single file Renaming option with Folders Bar
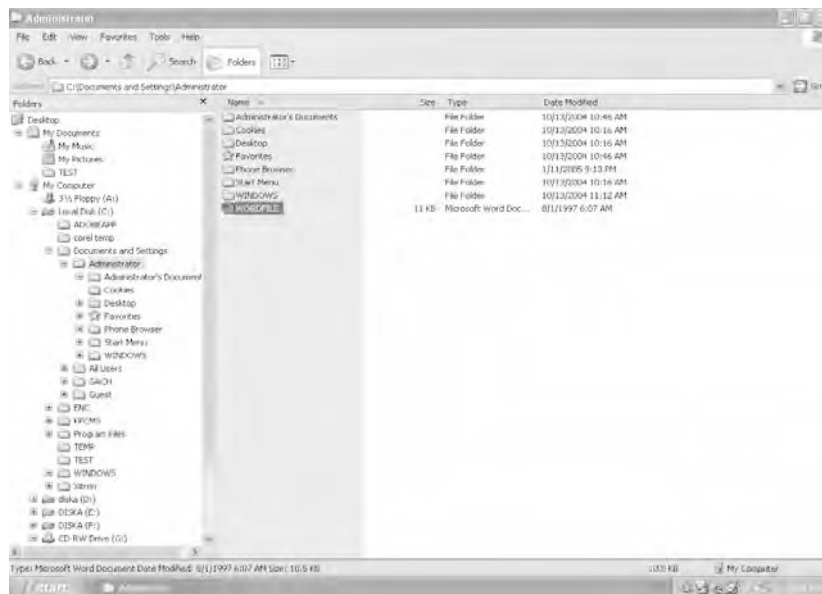
Fig 2.24 The file MANI has been renamed as SHIVA (Folders Bar)

95

If you want to Rename a group of files or folders, there is no special help from Explorer Bar. In fact, Explorer Bar misleads you. When you select files or folders to Rename, Explorer Bar will not show anything about renaming the group of files or folders. But you can follow method 2 and method 3 explained in the previous page. If you use the right-click method the file or folder that you have chosen to right-click, will get the name that you have chosen. For example, you have selected kumar.doc, student.doc, exam.xls and raj.xls. If you have chosen the name "rajan" to rename the group of files or folder, the item that gets focus is named as rajan, the other file are named as rajan (1), rajan (2) and rajan (3). If you follow the menu method, you will have the same result.
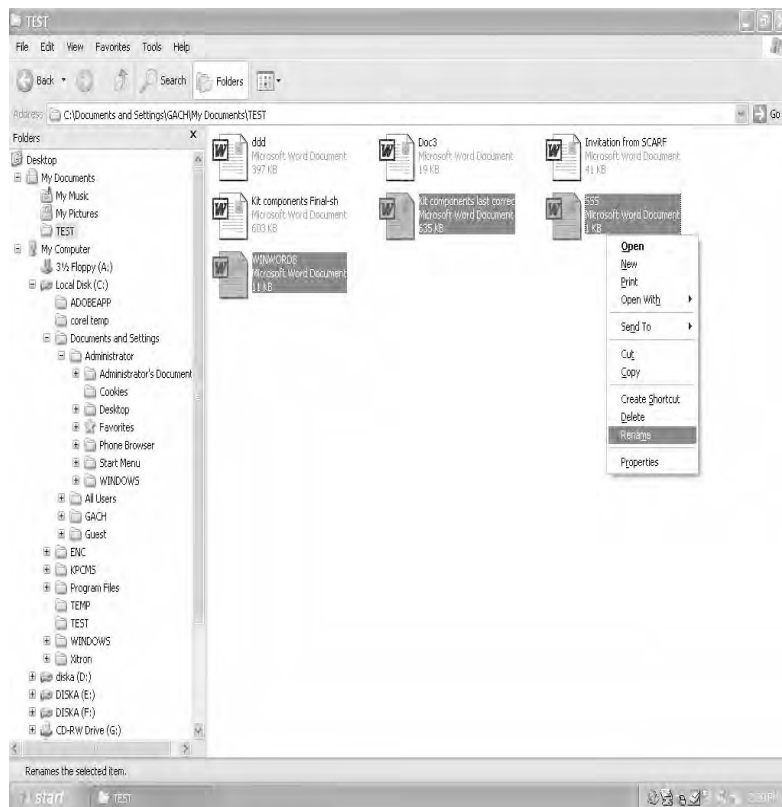
Fig 2.25 Renaming Group of Files

96
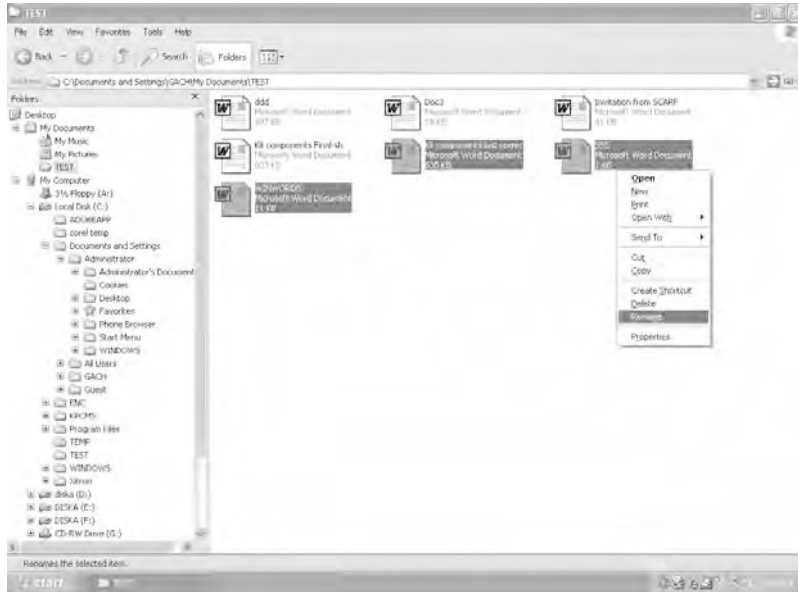
Fig 2.26 Group of Files Renaming process
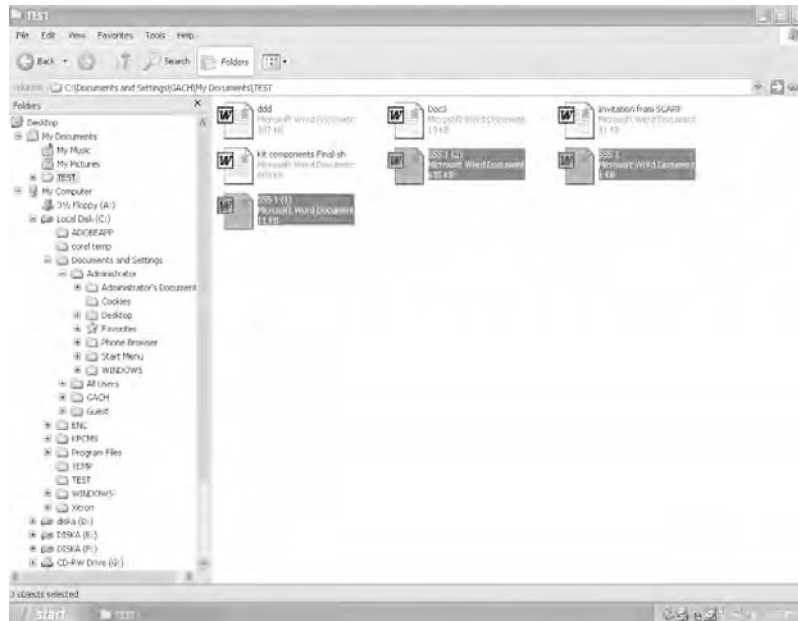
Fig 2.27 Group of Files Renamed

97

2.10 Deleting Files and Folders

Windows Explorer uses a special folder called the **Recycle Bin** to hold deleted files. The Recycle Bin is like the garbage can in your house that you empty once it is full. In the same way, you can empty the recycle bin when you want. Using the recycle bin gives you a chance to get back files that you have deleted by mistake.

To delete files, first select them. Then right click on the files and the shortcut menu appears.
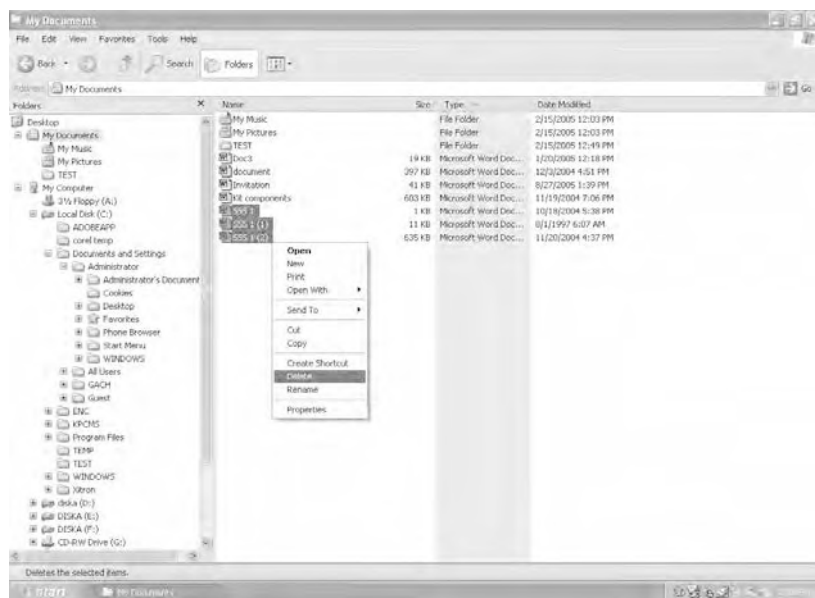
Fig 2.28 The shortcut menu with the Delete option

Select **Delete** from the shortcut menu and the files will get deleted. (In reality, they are moved to the Recycle Bin). You can drag the selected file / files to the Recycle Bin or to its Explorer Windows

When you are in **Explorer Bar**, you can delete selected files by just clicking **Delete the selected items** from **File and Folder Tasks.**
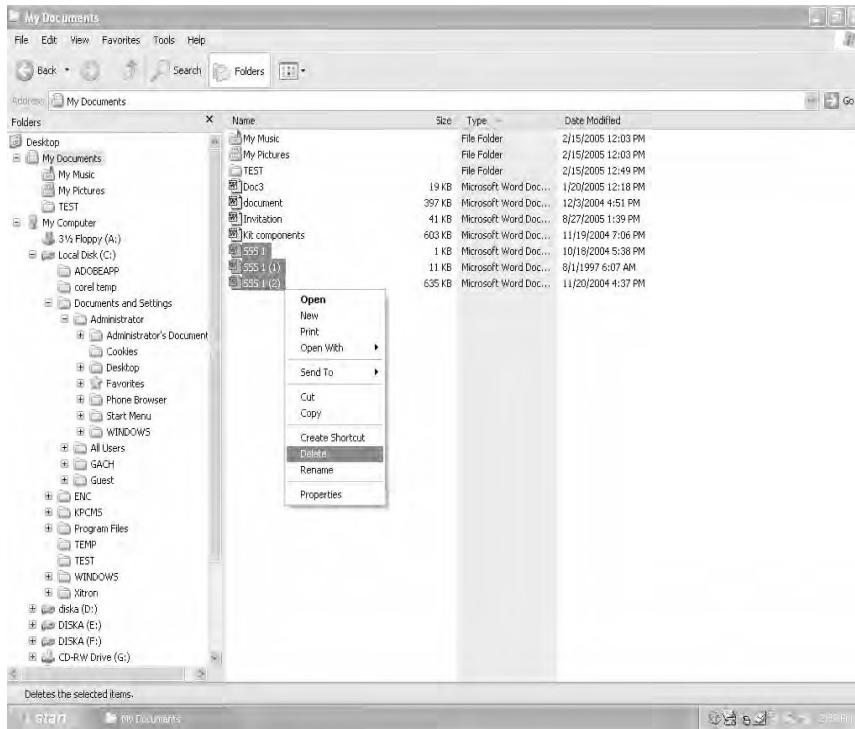
Fig 2.29 The My Documents folder after the files have been deleted

The **Recycle Bin** folder is available on the Desktop and can be used like any other folder. Double-click on the icon to open it and check if the deleted files are present. If you do not want to send the deleted items to the Recycle Bin, Shift + Delete key combination will achieve your goal. If you delete some items from floppy or from CD-RW, the contents will be deleted for ever. The contents will not go to the Recycle Bin.
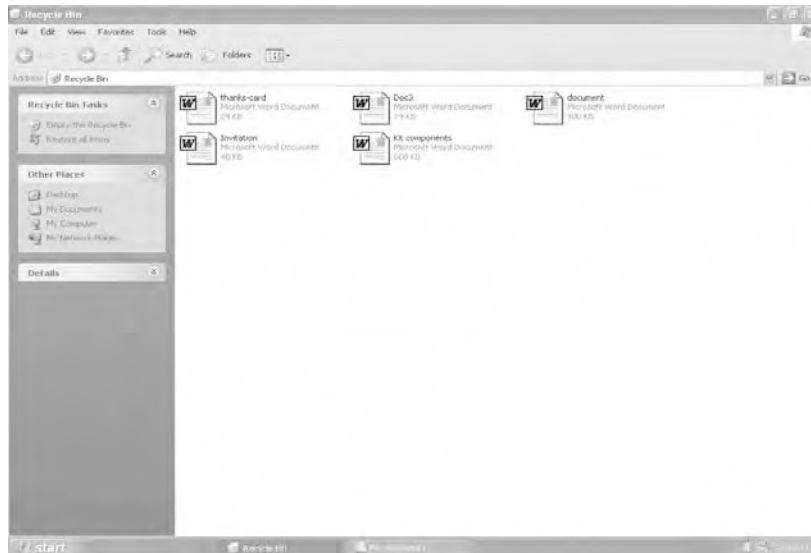
Fig 2.30 The Recycle Bin

To empty the Recycle Bin, click on the **File** menu and choose **Empty Recycle Bin**. If you double click the Recycle Bin, it will open in the **Explorer Bar**. You can see the Explorer bar in the left pane. Contents of the Recycle Bin are shown in the right pane. Under the **Recycle Bin Tasks** you are shown two alternatives. If you want to restore some item, you select them. Then click the **Restore**, all the selected items are sent to their former destinations. If you click **Empty Recycle Bin**, even if you select a few items, all the items whether they are selected or not will be eliminated from your computer's storage. First you select those items, which you want to recover from the right pane, and click **Restore** all items. You can then click **Empty the Recycle Bin**. Remember that once the **Recycle Bin** has been emptied, you cannot get back the deleted files.

Deleting Files and Folders from CD-RW disk :

You already know that a CD-R disk cannot be modified and CD-RW can be used as a floppy. If you want to delete the contents of CD- RW

100

disk, you have to follow the following steps. First you should open the disk, next you should delete the contents.

1)   Insert CD-RW disk into your CD-RW drive. Then one of the three possibilities will happen.

           i)   The Windows XP Professional may provide you with a dialogue box asking you **what you want to do**. Choose open folder to view files using Windows Explorer. Go to step 2.

           ii)   A program opens and starts playing the CD, close the program and then choose open folder to view files using Windows Explorer. Go to step 2.

           iii)   If nothing happens, open your My Computer folder, right-click the icon for the CD-RW drive and choose open folder to view files using Windows Explorer. Go to step 2.

2)   In the ensuing Explorer Window, click **Erase** and follow the instructions on the screen.
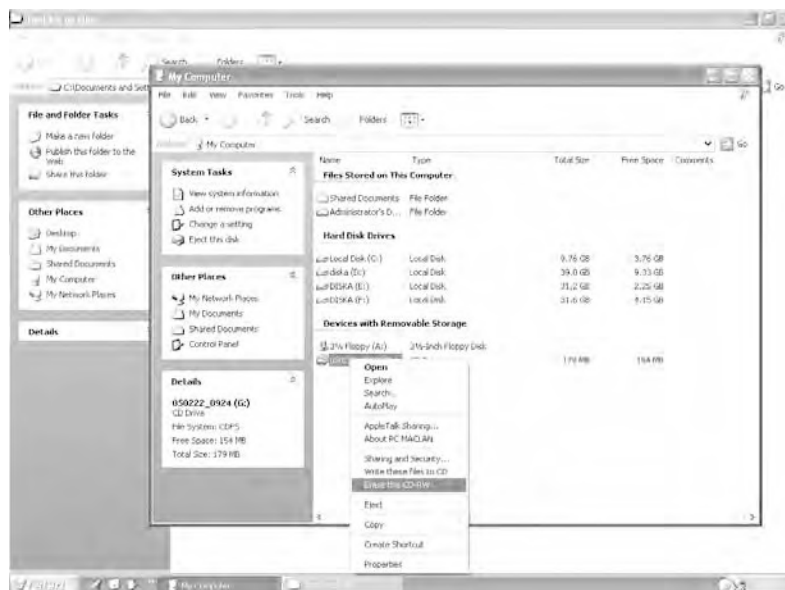
Fig 2.31 Using Erase

101

Fig 2.32(a) Using Erase -1   Fig 2.32(b)Using Erase -2
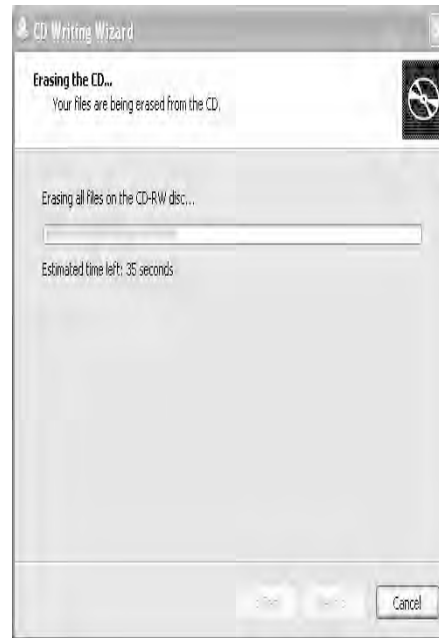
2.11 Creating Shortcuts

Among the many applications available on your computer, there will be a few that you use frequently. For example, you enjoy painting and frequently use Paint. To start Paint, you should click on **Start → All Programs → Accessories →Paint**. It would be more convenient if you could start Paint directly from the desktop. Windows XP allows you to create such shortcuts for frequently used applications. When you create a shortcut, Windows XP creates a link which points to the physical location of the program.

Windows XP allows you to create two kinds of shortcuts.

i)   Keyboard shortcuts

ii)   Desktop shortcuts

102

2.11.1 Keyboard shortcuts

You can create a keyboard shortcut for any program by using the Properties dialog box of that application. Let us understand this better with an example.

Suppose you want to create a keyboard shortcut for Paint. To do so, first click on **Start → All Programs → Accessories → Paint** and right click on it then select **Properties.**

Fig 2.33 The shortcut menu with Properties

The Properties dialog box opens on the screen. Click the **Shortcut** tab**.**
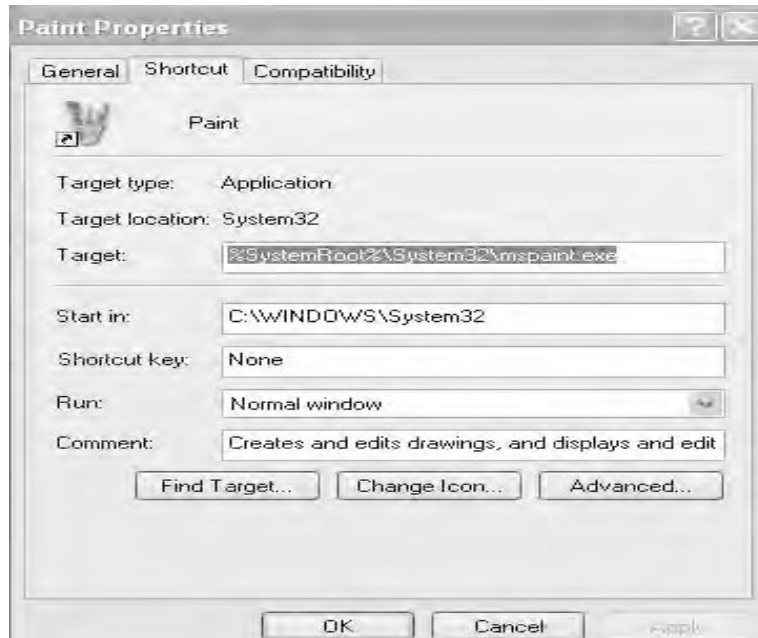
Fig 2.34 Paint Properties dialog box

In shortcut key text box type a letter of your choice, say **P** and click on **OK.** Now to start Paint, press **Ctrl + Alt + P** together.

Note: The shortcut key box will display " None " until you select the key and then the box will display Ctrl + Alt + "the key you selected", you cannot use ESC, ENTER, TAB, SPACEBAR, PRINT SCREEN, SHIFT or BACKSPACE keys.

2.11.2 Desktop Shortcuts

To create a desktop shortcut, first locate the application using Windows Explorer. For example, to create a desktop shortcut for Paint, start **Windows Explorer** and go to the **Windows** folder (Fig 2.35).

104
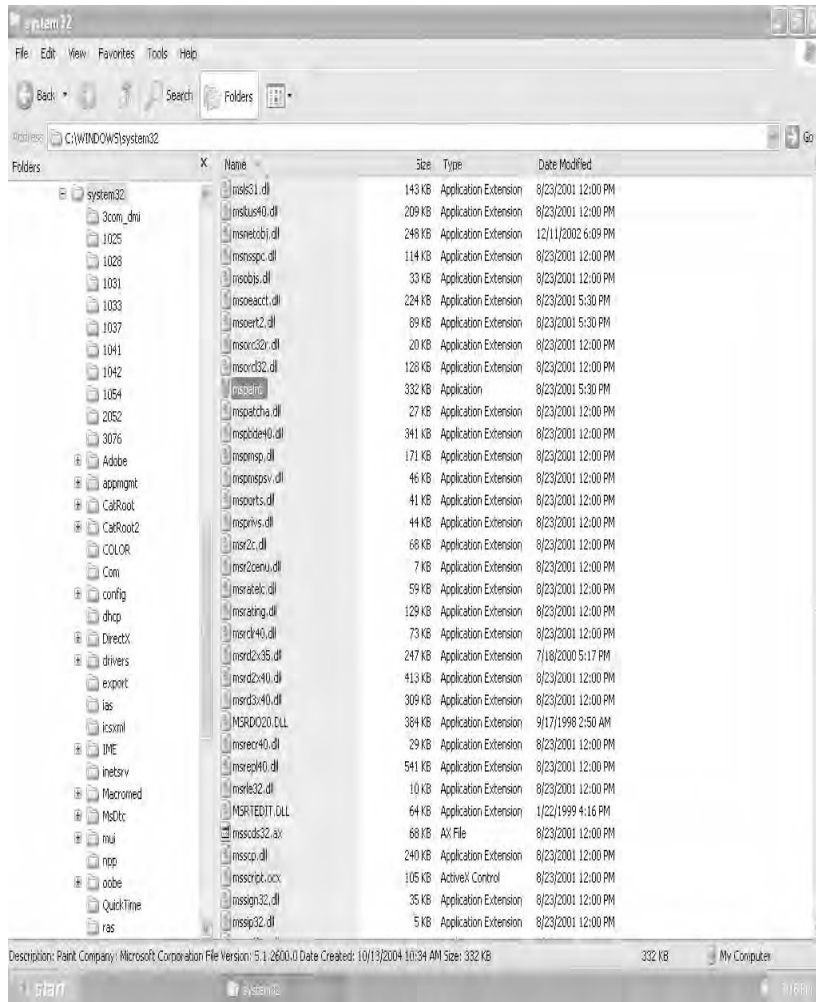
Fig 2.35 Paint application file in the Windows folder

Right click on the application file and select **Create Shortcut** from the
menu. A new file called **Shortcut to . . . .** is created. Restore the Ex-
plorer window so that you can see a part of the desktop. Drag and
drop the shortcut file. Now your desktop will look as shown in the figure
below.

105

Fig 2.36 Desktop with the Shortcut to Paint icon

Note – The small arrow to the left of the icon indicates that the icon is a shortcut.

You can start Paint by clicking on the shortcut icon.

2.12 Search

If you ask a novice computer user where his/her files are, the most probable answer will be "in the computer". This is just like saying my book is somewhere in the world. Even the experienced users sometimes lose their files; no matter how well they organize their files into folders on their hard drives. The computer will not eat the files. So the items must be in the computer, unless you deliberately removed them. The Search facility of Windows XP allows you to find the so-called lost item. Suppose you want to find a lost file. You cannot find something out of nothing. So you should know something about the file that are being searched for.

You may inform the **Search,** all or part of the file name, approximate date (or with in a week, month etc.) on which the file is saved or modified or downloaded.

If you search for a document containing text you should provide a word or phrase that appears in the document. You click **Start** → **Search** (or you open any Explorer windows click search button in the toolbar or click View → Explorer Bar → Search). The left pane of the Explorer windows becomes Search Companion. In the right pane you can see **To start your search, follow the instructions in the left pane**. In the left pane you can see what do you want to search for ? the first choice is Pictures, Music, or video. The meaning is self-explanatory. You select this under appropriate conditions. The next choice is "Documents (Word processing, spread sheet, etc.)". Select this, under appropriate conditions. The next choice is **All files and folders** (remote). If you want to search in All files and Folders, select this. If you want Computers or People choice, you should have the Internet connection. Here you are going to find a lost file, so you have clicked **All files and folders**. You have shown the dialogue box expecting you to furnish the information about **All or part of the file name**. You can make use of the wild card entries such as ?,*. The ? Stands for a single letter and * stands for zero or more letters. For example, if you know the document to be searched starts with "pur" and it is a picture document then you should enter a pur* in the text box. In the next text box you are expected to provide a word or phrase that appears in the file. You enter .jpg there. **In look in**: drop down list box you browse through it to select appropriate entries. Here click *Local Hard Drives (c::d:;e:;f:),* you click it. (You may have to select the above differently, you better consult your teacher.)
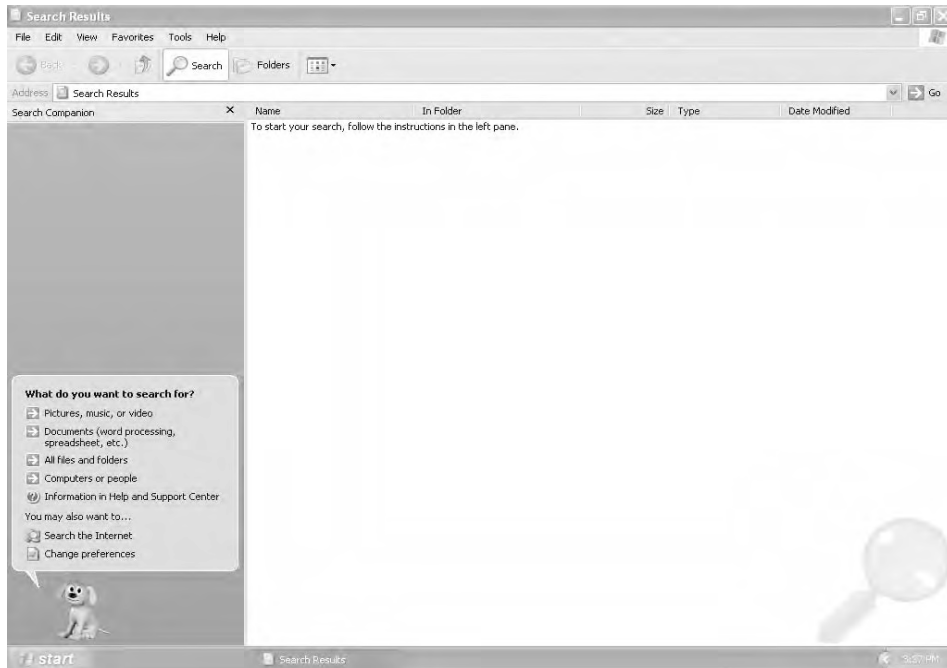
Fig 2.37 Search options

To answer the query **when was it modified**, if the required file is modified within the last week click on appropriate radio button. If you do not remember any thing about the period of modification, copying or downloading, leave the default selection as such. Click **what size is it** . It will display five options. You have to specify whether the size of the file is small, medium, large and another options is **specify size**. If you do not remember the size, leave the default selection as such. You do not disturb more-advanced options. Then click **Search**. Then in the ensuing dialog box click **Yes, finished searching** or else you follow the instructions given on the screen or click **Back** button to repeat the **Search**. The result is shown below. Even though, All Files and Folders is chosen for the Search, to introduce you many options, the natural option for this Search is Pictures, Music, or video.

108
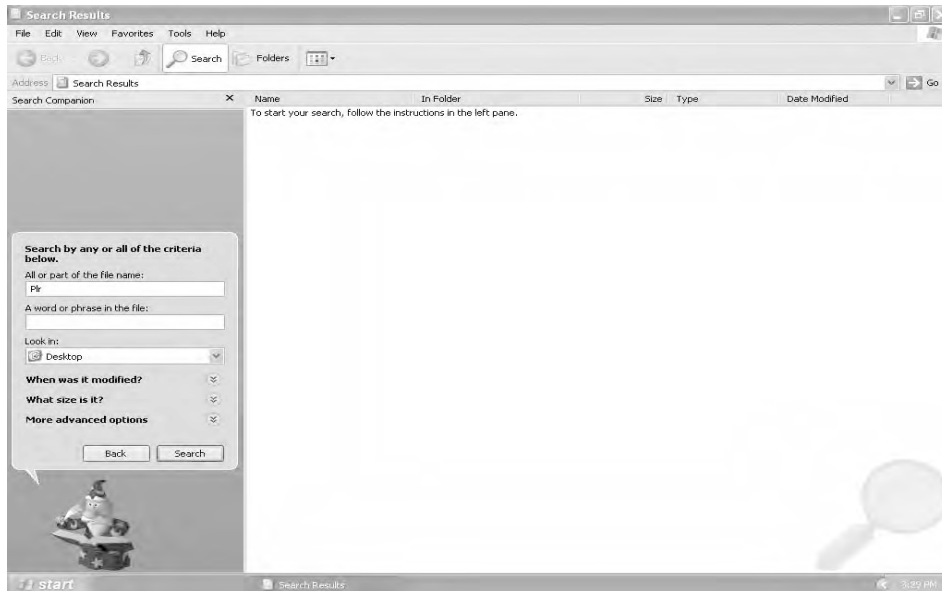
Fig 2.38 Expanded Search options

2.13   The Run Command

The **Run** command on the Start menu offers an alternate method to start applications or open data files. There is one advantage in using the Run command. When you use the Run command to open a data file, say a word processing document or a paint picture, it automatically starts the corresponding application also. The Run command is most often used to install new software or games from a CD or a floppy disk. The disadvantage of using the Run command is that you should enter the complete file name along with the Path. Path is the location of the file. Path names always start with the drive followed by folder names and end with the file name. The drive, folder names and file name are all separated by \ (backslash). For example, c:\My Documents\Project Report.doc refers to the document file named Project Report in the folder My Documents in the C: drive.

To use the Run command, click on the **Start** button and select **Run**. Type the file name in the **Open** box.

109
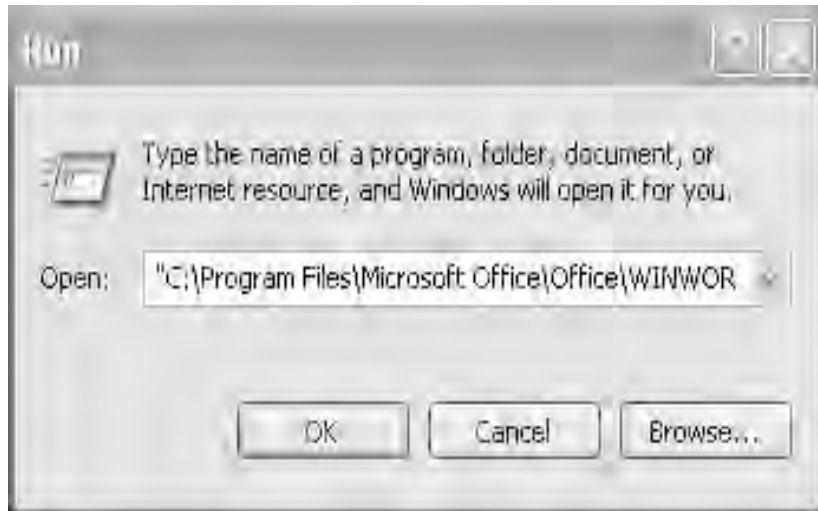
Fig 2.39 The Run command

The inverted triangle in the **Open** box displays a list of recently used pathnames. The **Browse** button lets you search for a file or folder.

2.14   What is new in Windows XP?

1)   Easier Installation and Updating

Roughly speaking, installation means the addition of program files and folders to your hard disk. Windows XP includes several features designed to make it easier to install and to keep up-to-date, the program files and folders

2)   Effective Multi-user Capabilities

Windows XP keeps each user's files separate so that no user can see another users files unless they have been shared deliberately. It lets multiple users Log on at the same time. End users run their applications.

110

3) Redesigned Start menu

Windows XP provides a redesigned start menu that is easier and quicker to use. The start menu appears as a panel containing two columns. The lower part of the left column automatically reconfigures itself to show your most used applications. The Start Menu can also be customized to the show Classic Start Menu (similar to the start menu of Windows 98).

4) Taskbar Changes and Enhancements

These improvements are designed to help beginners. Experienced user may switch back to how it was in the earlier versions of Windows, if they like.

Taskbar locking: By default, Windows XP Professional locks the Taskbar. This prevents taking the taskbar to an inaccessible area.

Taskbar scrolling: Taskbar locking prevents flexibility. If the taskbar is of a fixed size, buttons for the running applications must become very small and useless when 10 or more applications run. To tackle the situation Windows XP provides a scroll bar on the taskbar when required.

Taskbar Button grouping: Windows XP provides only one button per application when there is not enough space to accommodate buttons on the Taskbar. This too prevents minimizing the size of buttons displayed on the taskbar. It shows the name of the current active window along with the number of windows and a drop-down arrow. If you click the button, it will show you the list of Windows by title, you can select any one of them.

5)   Notification area

The status area (system tray) is known as notification area. Notification area shows a few icons of the programs which are automatically executed at start up

6)   Better Audio and Video Features

Windows XP includes a set of new features and improvements for audio and video.

7)   CD Burning

Windows XP provides built-in CD burning capabilities, which reduce the effort taken by the user while writing something into the CD.

8)   Search Companion

Windows XP includes Search Companion, an enhanced search feature to search for finding information both on your PC and in the World Wide Web.

9)   Enhanced Autoplay Feature

If you insert a CD and if it starts playing the music from it or installing any software it contains, immediately, this facility is called Autoplay. This feature is enhanced considerably in Windows XP.

10)   More Games

Windows XP includes more games than the previous versions of Windows. This may be a welcome move for young people

11)   Remote Desktop Connection

This improved feature lets you use your computer to access a remote computer with less effort.

12) A more Useful Winkey

One or two winkeys may be provided in modern keyboards. Normally the key is situated between Ctrl and Alt keys. This key possesses the Windows logo. Windows XP includes more functionality for the Winkey. You are provided table with the uses of Winkey.

WINKEY COMBINATIONS

Winkey Combination	What it does
Winkey	Toggles the display of the Start menu
Winkey+B	Moves the focus to the notification area
Winkey+Break	Displays the System Properties dialog box
Winkey+D	Displays the Desktop
Winkey+E	Opens an Explorer window showing My Computer
Winkey+F	Opens a Search Results window and activates Search Companion
Winkey+Ctrl+F	Opens a Search Results window, activates Search Companion, and starts a Search for Computer
Winkey+F1	Opens a Help and Support Center window

113

Winkey+L	Locks the computer
Winkey+M	Issues a Minimize All Windows command
Winkey+Shift+M	Issues an Undo Minimize All command
Winkey+R	Displays the Run dialog box
Winkey+Tab	Moves the focus to the next button in the Taskbar
Winkey+Shift+Tab	Moves the focus to the previous button in the Taskbar
Winkey+U	Displays Utility Manager

13) Improvement for portable computers

Windows XP includes several improvements for portable computers (such as Note book computers).

14) More Help

Windows XP delivers more Help-and more different types of Help-than any other version of Windows. You have already seen some help topics of interest.

15) Network Connectivity

Windows XP provides various improvements in network connectivity.

16) Multiple Monitor Support-For Both Desktop and Laptop.

Windows XP Professional also introduces a new technology called Dual View, which offers excellent opportunities to multiple monitor support especially to laptops.

114

The above characteristics can apply to both Windows XP Professional and Windows XP Home.

The following Characteristics strictly belong to Windows XP Professional

17) Backup and Automated System Recovery (ASR)

Windows XP Professional includes a Backup utility and an ASR feature that can be activated from boot up to restore a damaged system.

18) Offline Files

Offline files allows you to store copies of files located on network drives on your local drive so that you can work with them when your computer is no longer connected to the network.

19) Remote Desktop

Remote Desktop allows you to access the Desktop of the computer connected remotely as if you are accessing the Desktop of your own computer. If you need to connect to your computer remotely via Remote Desktop Connection, you need Windows XP Professional rather than Windows Home. So far, you have seen features that caught your eyes. Now, you are going to see the facilities hidden in Windows XPProfessional.

20) Protected Memory Management

Windows XP offers fully protected memory management. With this facility, Windows XP can handle memory errors effortlessly.

21) System File Protection

Windows XP offers a feature called System File Protection that protects your system files from inadvertent mistakes on your part.

22)   System Restore

Windows XP provides a System Restore feature. This is more effective than System Restore feature found in Windows Me.

You can use System Restore to rollback the changes to an earlier point at which the system was working properly..

23)   Device Driver Rollback

Windows XP tracks the drivers you install and lets you roll back the installation of the driver. In other words, you can revert to the driver you were using before.

24)   Compatibility with Windows 9x Applications

Windows XP runs all applications that would run on Windows 9x, Windows NT and Windows 2000.

2.15   Guarding Against Viruses

The literal meaning of virus is poison. Virus enters into the living things and passes its code to the cells of the host. The host cell forgets to undertake its own work, it becomes the industry for producing viruses. Computer virus is a mischievous program designed to damage the Software, Hardware and / or data.

The technique of the biological virus is employed by the computer virus also. It enters your computer as innocuous software and multiplies many times. In that process, it takes the lion's share of the memory normally, erasing your own useful programs.

Though virus started from the Bell Laboratory in the name of core wars, it showed its ugly head to the world by the handiwork of a self taught Software Engineer. But still the method of creating viruses was kept as a secret. One of the eminent computer professionals, while receiving

a prestigious award, revealed the secret of creating viruses to the audience. The entire computer world was shell-shocked. This opened the Pandora box. From then on, the computer world is cursed with many viruses. Most of them are created by the students to just show their intelligence to the world, thus causing a loss of millions of dollars. The virus designers mainly attack windows OS.

Viruses come in three basic flavours. They are File infectors, Boot sector viruses and Trojan horse viruses

◀ File infectors attach themselves to executable files and spread among other files when you run the program.

◀ Boot sector viruses replace the hard disk's master boot record (or the boot sector on a floppy disk) with their own twisted version of the bootstrap code. This lets them load themselves into memory whenever you boot your system (the famous "Michelangelo" virus is one of these boot sector beasts).

◀ Trojan horse viruses, which appear to be legitimate programs at first glance but when loaded, proceed to viciously damage your data:

◀ Viruses are, by now, an unpleasant fact of computing life, and you just have to learn to live with the threat. But somehow in the beginning, the Microsoft chose to ignore this ugly threat, but now Microsoft deals with this crime more seriously in Windows XP. There are vendors who provide antiviral vaccines that will protect you from the hazards of this threat. Antivirus is a program to safeguard your system from the virus programs.

There are many such antiviruses, which make the life of the programmers somewhat easy.
Here are two tips to keep your system virus-free:

1. The main source of the viruses is the floppy disk. So, one should be very careful about the floppies.

117

2. Now-a-days, the Internet is the major source of producing viruses. One should be very careful while downloading files from the Internet. Keep your virus utility's virus library up-to-date. By some accounts,   more than 100 new virus strains are released each month, and they just get nastier and nastier. Regular updates will help you keep up-to-date.

The Economical Explorer Keyboard

If you want to have alternative methods for the mouse click, here is the table.

Alt+Enter	Display the properties sheet for the selected objects.
Alt+F4	Closes Explorer (actually closes the active window).
Alt+left arrow	Takes you back to a previously displayed folder.
Alt+right arrow	Takes you forward to a previously displayed folder.
Backspace	Takes you to the parent folder of the current folder.
Ctrl+A	Selects all the objects in the current folder.
Ctrl+C	Copies the selected object to the Clipboard
Ctrl+V	Pastes the most recently cut or copied objects from the Clip
Ctrl+X	Cuts the selected objects to the Clipboard.
Ctrl+Z	Reverses the most recent action.

Delete	Sends the currently selected objects to the Recycle Bin.
F2	It helps to rename the selected object.
F3	Displays the Find dialog box with the current folder as the default.
F4	Opens the Address toolbar's drop-down list.
F5	Refreshes the Explorer window. This is handy if you have made changes to a folder via the command line or a DOS program and you want to update the Explorer window to display the changes
F6	Cycles the highlight among the All Folders list, the Contents list, and the Address toolbar.
Shift + Delete	Delete the currently selected objects without sending them to the Recycle Bin.
Shift+F10	Displays the context menu for the selected objects.
Tab	Cycles the highlight among the All Folders list, the contents list, and the address toolbar. F6 does the same thing.

Summary

◄   All information on disks are stored as Files. Every file has an unique file name.

·   A collection of files is called a Folder.

◄   Windows Explorer is an application that allows you to manage your files and folders.

◄   Windows Explorer provides two Bars. They are Explorer Bar, Folders Bar. Explorer Bar provides easy way to move, copy or delete.

◀ Using Windows Explorer, you can, View the files and folders on your disk·

◀ Create new folders, Copy and move files and folders. Rename files and folders

◀ Delete files and folders

◀ Create shortcuts for frequently used files and applications.

◀ CD-RW can be used as a floppy.

◀ The Search feature allows you to search for files or folders.

◀ The Run command provides an alternate way to start applications and open data files.

◀ Viruses are ugly programs that spoil work. One has to be care ful about them.

Exercises

I.   Fill in the blanks

1.   Information is stored as _____ in your computer.

2.   Every file name has two components: the _____. and the _____.

3.   _____ is collection of files.

4.   _____ allows you to manage your files and folders.

5.   The _____pane in Windows Explorer displays a list of   folders.

6.   The display in the Explorer Bar is   sensitive.

7.   The _____ command allows you to search for files or folders.

8.   The _____command provides an alternate method to start applications and open data files.

9.   The Search command is available on the Start menu. It is also available on the   _____toolbar.

10.   A plus sign to the left of a folder in the left window of Explorer indicates the presence of _____.

11.   The _____ key is used to select a group of files whose names are not displayed next to each other in the Explorer window.

12. The CD can be used as a Floppy.
13. Alt + F4 key combination is used to
     active window.

II. State whether the following statements are True or False

1 . In Windows XP two files in the same folder can have the same
     name.
2.   Windows XP allows you to give file names with spaces.
3.   A folder can contain several subfolders:
4.   Windows Explorer allows you to work with only one file at a time.
5.   You can start Windows Explorer by right-clicking on the Start
     button.
6.   There is no difference between copying and moving files.
7.   In Windows XP, files cannot be deleted. They can only be moved
     from their folders to the Recycle bin.
8.   The Clipboard is a temporary location for files being copied or
     moved.
9.   The Send To option is used to copy files to a floppy disk.
10. You can search files based on file type using the Search com
     mand.
11. In Windows XP, the file name should not exceed more than eight
     characters.
12. Shift + Delete deletes the selected item / items permanently.

III. Answer the following

1.   Write short note about opening Explorer Window in Explorer
     Bar and Folders bar ?
2.   What is the Run command used for ?
3.   What are files and folders ?

4. Write a short note on file names.
5. Explain the Recycle Bin. How is it used?
6. How do you select files in Windows Explorer ?
7. Describe briefly the different ways in which you can view information in Windows Explorer.
8. Describe the different parts of the Windows Explorer window.
9. How do you create a new folder?
10. What is the difference between copying and moving files?
11. Describe the different methods to copy the selected files.
12. How will you Rename a group of Files ?
13. How will you Copy files to CD ?
14. Distinguish between CD-RW and CD-R ?
15. Explain the different ways in which the selected files are moved.
16. What are the special features available in Windows XP Professional alone ?

CHAPTER 3

LINUX

3.1 History of Linux

While Linus Torvalds was studying in the University of Helsinki, Finland, he had to do a project according to the norms of the University. At that time, students used a version of Unix called Minix created by Prof.Andrew S.Tannenbaum. After studying Minix, Linus was very much attracted by the elegance and effectiveness of its parent Unix. The Unix operating system was created mainly by the efforts of Ken Thompson. Linus Torvalds decided to develop an effective PC version of Unix for Minix users. He called it Linux by combining his first name with the last letter of Unix (also Minix) and he released version 0.11 in the year 1991. Linux was widely distributed over the Internet. The other programmers, in the subsequent years, refined the new operating system, Linux and added some features found in standard Unix systems. Interested programmers all over the world contributed something to the improvement of Linux. In this way, it has got a distinctive advantage over the other Operating Systems. Normally, the Operating Systems are developed under a closed environment. A very limited number of people were allowed to remove the errors in the entire code within a fixed deadline, whereas the Linux code was available to one and all, they could fix the errors, under friendly atmosphere without any constraint. All these produced useful code. Linux has all the utilities needed for the the Internet.

In its simplest format, it can run effectively requiring just 4MB of memory. It is amazing that this operating system with all its features, occupies such a small memory. This has not affected its stability or speed.

Even though Linux was developed by the contribution of many people throughout the world, it is not unwieldy. Linux was developed from the beginning according to the ANSI standard for Unix called the Portable Operating system Interface for Computer Environments (POSIX). Linux is specifically designed for Intel-based PCs.

123

There are two versions for each release of Linux .One is a stable version and the other is a trial (or beta) version. In a **n. x .y** version, the first number **n** specifies version number, if second number **x** is even then this is stable otherwise it is a   beta version. For example 2.2.5, here x = 2. Therefore this is a stable version.

The Internet is a boon for Linux development; it enables the people through out the world to interact with others to develop Linux. Today, many companies   provide support for Linux over the Internet. There are many Linux groups on the Internet, and registration to these forums is free. You can subscribe to and get the latest information from these forums.

3.2   Logging in / Logging out of Linux

 Linux systems allow many users to work simultaneously. A user normally works at a user terminal. You have to establish connection to the Linux system, the system after showing some information; will show Login prompt(prompt is a helping message), which is the location where you enter your user name. Your Login name is nothing but user (your) name. You need not be frightened by the word Login, which simply means user here.

Assume a house with many rooms and one person occupies one room and each room contains invaluable treasure. How will you safeguard each room of that house? The entire house should be locked inside and a watchman should be employed. The arrangement is that you should tell him your name and your identity code. He has a list of names and matching identity codes for each individual. If the name and the identity code match with any of the name and its identity code, then and only then, he will allow you to enter into your room. If the name and the identity code do not match with any of the names and the corresponding identity codes then you will be asked to repeat your name and identity code. This process continues until user says the

valid name and identity code.

The same process happens in Linux also. When you connect to the Linux system, you will be asked to enter your Login name at the Login prompt, you have to enter your Login name and you will be shown the Password prompt. You have to enter your password. To keep your password a secret, the Linux system will not display your password. The system compares these two items with the system files. If the match is not found, then you should enter these two items. Ofcourse, if the match is found you will be shown the following prompt;

[ilamathi@localhost ilamathi] $

Here you assume your login name is ilamathi.

You should be careful about the lower case and upper case letters because the system is case sensitive. The verification process does not allow any unauthorized person to access any of your directories or files. Normally, not necessarily, the name of the user is the Login name. The Home Directory is assigned to the user when he/she enters into the system for the first time, by the System Administrator (SA). You will learn about SA later.

Logout Process

If you come out of the system without closing your /home directory then the other people may tamper with your work. So it is mandatory to Logout of the Linux system. Entering **exit** or **logout** at the command prompt will end your current Linux session then the system displays the Login prompt on the screen for other users.

Changing the password

Suppose you have an uneasy feeling that someone knows your password. Then you have to spend sleepless nights. The Linux operating system has its own method of solving your problem. It allows you to change your password. A user can change his/her password with the **passwd** command. The steps followed by the user, ilamathi, to change her password are depicted below. The actual prompt is similar to the one shown here.

Example:

[ilamathi@localhost ilamathi]$ passwd (the user enters his/her password)
Changing password for ilamathi (current) password: (User enters the current password)
New password: (User enters the new passeord)
Retype new password: (User re-enters the new password)
passwd: all authentication tokens updated successfully
[ilamathi@localhost ilamathi]$ _

The passwd command asks for the old password. This command is essential to check up the authenticity of the user, otherwise the mischief mongers will play havoc on the work of the other people. Again the system demands that the user should make up his/her mind about the new password. That is why it asks the new password twice.

Example**:**

[ilamathi@localhost ilamathi]$ passwd

Changing password for ilamathi
(current) password:(User enters the current password)
New passwd:(User enters the new passeord)
Retype new password: (User misspells the new password)
Sorry, passwords do not match
New password:

If you commit a mistake in typing the current password (that is the password that you want to change) then you may not be an authenticated user. So, there is a suspicion that you are trying to change someone else's password. Therefore you will not be allowed to change the password. If two entries of the new password do not match then the system is in a dilemma. The system does not know which is to be accepted. So, your new password is outrightly rejected. Hence you should be careful about the old as well as the new passwords.

Some of the rules for changing the password:

The password should not be less than six characters in length.
When changing a password, the new password should differ from the old password by at least three positions
The password should be different from the user's Login name.
The new password cannot be the same as the old password. It is ridiculous to change the password with same password.
Since the SA can execute control over the entire system, SA can change the password of any user of the Linux system.

3.3   The Linux File System

How will you store information in a computer as long as you wish to

retain them? The information can be stored in a file. But, what is a file?

File is a collection of records. A record gives information about an entity. An entity may be a student or a railway passanger in the reserved compartment with a valid ticket. For example your mark sheet also is a record. A record consists of fields. Elementary fields cannot be further subdivided to give any meaning. Name, Rollno are examples for fields. The collection of the mark sheets of your classmates is an example of a file. Now consider the collection of mark sheets of all students of your school. This is a collection of files, which is called as a directory. The school has got many files such as mark sheet, T.C, Pay roll. They should be kept separately under suitable captions. The school has data about each student for T.C. For a class, these data will constitute a file. All relevant files of the same type will form another set of files called a directory as already stated. In this fashion a school may have several set of files. Mark sheets file should be separated from the data files for T.C and so on. So the school has to place mark sheet file separately from other collection of files. The collection of same type of files is placed in a directory. A single fixed disk can store thousands of files. Arranging the files in the above mentioned manner would make accession a particular file easier. All the files are stored on the disk under one main directory called the **root** directory. The files are arranged under a tree structure. If you stand **on** your head and watch a tree (without trunk), the root of the tree is at the top then comes braches, braches will give rise to other branches and ends up with leaves. The leaf represents the file, the branches represent directories or sub directories and the root as you have guessed is the **root** directory. The **root** directory has been further sub-divided into directories such as *bin, boot, home, usr, etc, lib, dev, tmp*. User directories are created under the home directory. The home directory is written as **/home.** This

shows that home directory is the child of **/**(called **root)** directory and **root** directory is the parent of **home** directory. There is no parent for the **root** directory. There is no child for the file.
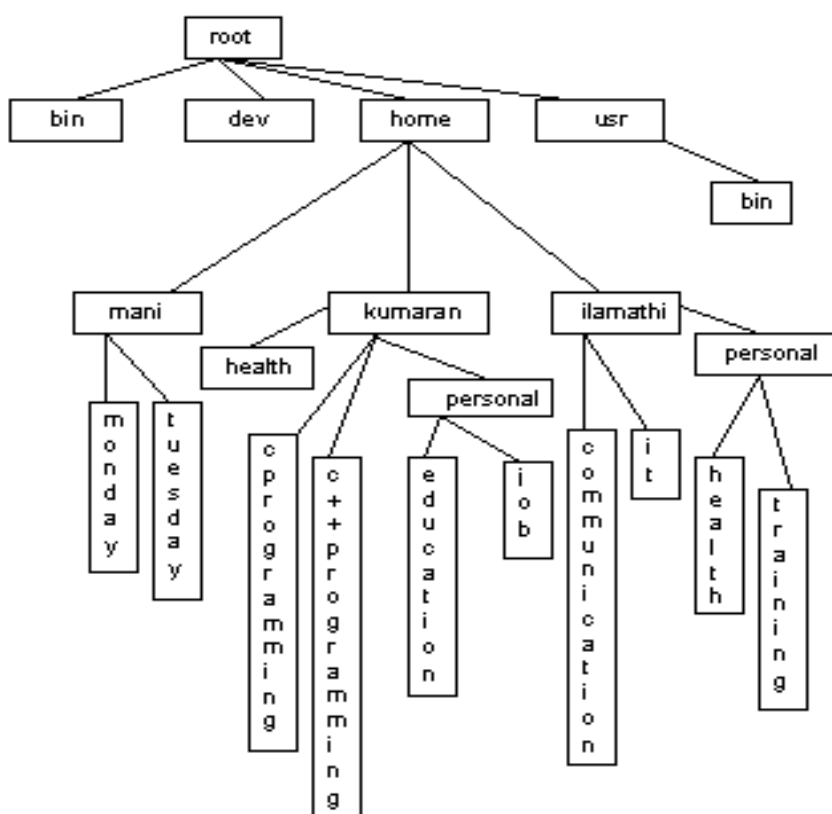
Fig 3.1 Linux directory structure

When a child is born, the child should be named. In a similar fashion, when you create a file, you should name it. You should follow the following rules to create a file.

The file name

1.   may contain characters, underscores, numbers, periods and commas.
2.   can be up to 256 characters.
3.   should not have a number as the first character.
4.   should not begin with a period. (Even though under certain special circumstances you may start a filename with a period, you will be in a better position if you do not make use of it. If a file starts with a dot, that file is called a dot file. The dot files are used normally by the system.)
5.   should not contain slash, question mark and asterisk.
6.   should not duplicate command names.

The filename may have a primary and a secondary name. The name before the period is called the primary (or proper) name. The name after the period is called the secondary name or extension. The extension is used to classify the files. For example consider files with extension c (.c files). These represents c files.

The name that you give, when you create a file or directory, is not its full name. The full name of a directory is its path name. The tree structure of the file system can be used to unambiguously identify and reference any directory or file. For
 example full name of the file C programming is **/home/kumaran/ cprogramming**

3.4   Types of Users

There are four types of users in the Linux system. They are

1. The System Administrator or the Root User

The System Administrator (SA) is primarily responsible for the smooth functioning of the system. The SA also creates /home directories for

the users and he/she does the service to groups of users for the system. He/she is the only one to use floppy disk and CD-ROM in the system and takes backups to prevent loss of data due to system breakdown. In Linux, he/she is also known as the root user or super user. The prompt for the root user is # while the prompt for others is $.

2. File Owner

The user who creates a file is said to be the owner of that file. The owner of a file can perform any operation on that file such as copying, deleting, and editing. He/she can execute a file after changing the file access permission by **chmod** command. You will see **chmod** command later.

3. Group Owner

A group of people who work on a single project should share their files for efficiency. The files are created in the group leader's /home directory. All the members of the group share their files. This group of people is called group users. A group of users is also given a name, just as a user is given a name.

4. Other Users

All the users of the system who are not members of a project group are referred to as Other Users, for the files of that group. Other Users are users who do not belong to that particular group.

3.5   Directory Commands
The general form of Commands

The first thing that you are expected to know is some fundamental concepts of commands. The general format of a command is**:**

Command option   argument

Not all commands need the "option" and "argument". Some needs option alone some needs argument alone and while others need both. For example **ls (**abbreviation for **lis**t but do not try to help the system by typing list   instead of **ls.** This helping action is greeted with an error message. The command **ls** saves you from typing two more characters, but it adds human memory load.**)** command works without option and argument, with argument alone, with option alone and with both.

Example :

Command   option   argument

ls		
ls	**-l**	
ls		***.c**
ls	**-l**	**file1**

You will meet the above shortly.

Command consists of a single word. The command generally starts with an alphabet. You do not have any control over the names of the commands. You should give the spelling and case as such.

Option starts with a minus (dash) sign followed by a single letter but you can combine two options. Case of the letter is very important. There should be no blank space between the minus sign and the letter. The minus sign distinguishes an option from a command and argument. The same option will not behave in the same fashion with different commands. The option is   influenced by the command preceding it. For example **–a** option with **ls**   command (ls –a) brings details about the files in the current directory including the hidden files. But the same option **-a** in [$mark **-ge**   81 **-a** $mark **–le** 100] behaves as **logical and** of C programming.

132

Argument usually refers the name of the file on which the command should work with. It should start with an alphabet. Some system files called dot files start with a **. (dot)**.

In the Fig. 3.1 even though there are two bins and two personal directories, they can be uniquely identified by the system because of the path name. The path names for the bin directories are /bin and /usr/bin. The first slash (/) always represents the root directory. Similarly the path names of the personal directories are /home/kumaran/personal and /home/ilamathi/personal. Even though the personal directories shared the common path upto /home, they take different routes later. This tree structure prevents the name collision. If you place all your files at one place, the same names cannot be used. Further it will take more time to search for a particular file. In Linux, you write a path name by listing each directory in the path separated by a forward slash. As already stated, a slash before the first directory in the path, represents the **root**. Path name also applies to files. The file is created within a directory by specifying a name. The system identifies the file with filename combined with the path of the directories from the **root** to the file's directory.

Path names are of two types. They are

1. the Absolute path name and
2. the Relative path name.

An absolute path name is the complete path name of a file or directory starting with **root** directory.

A relative path name begins with your working directory. It is the path of the file relative to your working directory.

Referring to the directory structure of the Fig. 3.1 , if your working directory is kumaran, the relative path name for the file 'job' is /personal/job. The absolute path name for the same file is /home/kumaran/personal/job.

How will you identify the current directory path? At times, you may not know where you are in the directory system. At such times **pwd** (**p**rint **w**orking **d**irectory or **p**ath of your **w**orking **d**irectory or the **p**resent **w**orking **d**irectory) is handy. You may assume that ilamathi has logged on the system and given **pwd** command in the $ prompt as follows:
[ilamathi@localhost ilamathi] $ **pwd**
/home/ilamathi
 [ilamathi@localhost ilamathi] $

will be displayed. The **/home/ilamathi** is the response of the system and gives the absolute pathname of home directory of ilamathi. Unless otherwise specified, the line next to the command is the output.

Changing the Current Directory

The **cd** (**c**hange **d**irectory) command changes the current directory to the specified directory. For example the current user ilamathi wants to switch over from her home directory to **/usr/bin.** She first finds out the current directory to ascertain her position and then she switches over to **/usr/ bin.** She executes the following commands at the command prompt.

[ilamathi@localhost ilamathi] $ **pwd**
/home/ilamathi
[ilamathi@localhost ilamathi] $ **cd /usr/bin**
[ilamathi@localhost bin] $ **pwd**
/usr/bin

If she wants to move to parent directory of the current directory then she has to use **cd ..** at the $ prompt.
cd command will not show the directory on the screen. To show the directory, **pwd** command should be given at the $ prompt.

Note: The double dots (**..)** denote the path of parent directory. The single dot(**.**) represents the directory itself. There should be a blank space between **cd** and **..** and there should be no blank space between the

two dots.

Example:
[ilamathi@localhost bin] $ **cd ..**
[ilamathi@localhost /usr] $ **pwd**

/usr
[ilamathi@localhost /usr] $ **cd ..**
[ilamathi@localhost /] $ **pwd/**

 Now kumaran has logged into the system and he is currently in the directory
/usr/bin. The user now wants to go to his home directory and he has to give simply the **cd** command. This will take him to his home directory.

Example:

[kumaran@localhost bin] $ **cd**
[kumaran@localhost kumaran] $ **pwd**
/home/kumaran

Note: The **cd** command without any path name always takes a user to his/her home directory.

kumaran is currently in the directory, **/usr/bin**. He decides to go to his **personal** directory. The easiest way in which this can be achieved is to use the combination of **tilde** (**~**) sign and /personal.
[kumaran@localhost bin] $ **~/personal**
[kumaran@localhost personal] $ **pwd**

/home/kumaran/personal
[kumaran@localhost personal] $ **cd ~**
[kumaran@localhost kumaran] $ **pwd**
/home/kumaran

Note: The tilde sign is a substitute for your home directory.

135

Assume kumaran is in his personal directory and he wants to go to his health directory. The **cd ..** command combined with **/**health will do the trick. He can give the following command at the $ prompt.

[kumaran@localhost personal] $ **cd .. /**health
[kumaran@localhost health] $ **pwd**
/home/kumaran/health

A directory under another dirctory is called the subdirectory of the latter directory.

Creating a Directory

The user kumaran wants to create an insurance directory under his health directory. He is already in the health directory. He should make use of **mkdir** (**m**ake **dir**ectory) command. This command creates the directory specified after the **mkdir** command**,** under the current directory. But the newly created directory will not become the current directory automatically. If you want move to the newly created directory you have to make use of **cd** command.

[kumaran@localhost health] $ **mkdir** insurance

A new directory under health is created.
You can specify the full path to make a directory with **mkdir** command.

Removing a Directory

When there is a provision to make a directory, there should be a provision to remove a directory also. Can you guess the command? Yes, you are right! It is **rmdir** (**rem**ove **dir**ectory)**.**

 Example:
[kumaran@localhost health] $ **rmdir** insurance

If you want remove a directory, the directory

1) should be empty and
2) should not be the current directory.

Kumaran did not store any thing in the insurance and the current directory is only health not insurance. So, it is possible to delete the insurance directory.

If you want to remove a directory, which is not empty, you can make use of **rm** command, which will be discussed shortly.

As in the case of **mkdir** command, you can use full path.

Listing the Contents of a Directory

Suppose you want to find out the name of the files and the subdirectories of a directory, **ls** is the only candidate to achieve this. For example the user Kumaran wants to know the names of the files and directories of /home/kumaran, he must give the following command.
[kumaran@localhost kumaran] $ **ls** /home/kumaran
healthcprogramming   c++programming   personal.

[kumaran@localhost kumaran] $ **ls - F**

health / cprogramming c++ programming personal /

The option **-F** adds / at the end of the directories and sub-directories.

Note: The simple **ls** also will give you the same result since kumaran is in his /home directory (without / at the end of the subdirectories).
If you want to have the information about the current directory, simple **ls** command fulfills your requirement. If you want to have more information about the files and sub-directories you should give **–l** option with **ls** command. The option **–l** will not list any hidden files but **–a** option with **ls** will list all the files and the sub-directories including the hidden files. You can combine the options **–a** and **–l** in any one of the following ways **–al**, **-la, –a -l** or **–l –a**.

137

The common options available with **ls** command are given in the following table.

Option	Function
-a	Lists all the files including hidden files.
-F	Shows the file type along with the name ('/'is added at the end of each directory to distinguish it from file).
-R	Lists Working Directory as well as all sub-directories..
-r	Displays files and sub-directories in the reverse order.
-s	Sorts by file size.
-A	Displays the files of almost all directories except the **.** and **..** directories

3.6   Other Commands

Manipulating the screen

The **clear** command clears the screen. The same thing can be achieved by **tput clear** command also.

The command **tput cup** 20 20 will position the cursor at row 20, column 20.

Extracting the Help

If you want to have help for some command say **ls** you have to use the command **man** (**man**ual). The **man** command works as Man Friday (a general servant or employee who does all kinds of jobs)

Example:

$ **man ls**
You can also specify the level of help you need from **man**, the level number should be specified in between **man** and the command for

138

which the help is sought. Now you can get help, online.

The **echo** Command

The **echo** command works as a combination of printf () and "\n" of C programming. If you want to display a message to the user you can make use of **echo** command.
Example:
$ **echo** " Please enter your name"
Please enter your name will be displayed on the screen and the cursor will be on the next line. If you want to have the cursor in the same line with the message, then you should use the **–n** option with the **echo** command.

Example:
$ **echo -n** "Please enter your name**"**
The double quotes ("") improve readability. You can also give the above statement as follows
 $ **echo -n** Please enter your name

Summary

You may enter into a Linux session by Logging on to the Linux system.
You may come out of a Linux session with the Logout or **exit** command.
You can change your password.
Linux uses a hierarchical file system to enable faster access to files.
There are different types of files in Linux, such as:
· Ordinary files · Directory files · Special files

The types of users for files are:
. Root user
· File owner
· Group owner
· Other users
The general format of a command is**:**

command   option   argument

Some of the commonly used as directory commands are:

- **pwd**   Prints the current working directory.
- **man**   gets help
- **mkfs**   formats a floppy.
- **cd**   changes the current working directory
- **mkdir**   creates. a new directory .
- **rmdir**   Removes an empty directory
- **ls-l**   displays the contents of a directory.
- **echo**   used for prompting.

Fill in the blanks

1. Ken Thompson is the creator of the _____ operating system.
2. Minix was created by _____.
3. The root user (SA) is empowered to change the _____ of any user.
4. The user can login into the system by entering his / her _____ name and ....
5. The user can change his / her password by entering the old _____ (current) Password with _____

.

6. Linux is a _____ user system.
7. The prompt for the root user is _____.
8. To find out the current directory the _____command is used.
9. There should be at least one _____ between cd and ..
10. _____ sign represents full path of your home directory.

State whether the following are True or False

1. Ken Thompson created Minix.
2. A small number of people from Hungary alone improved

140

Linux.

3. In simplest form Linux needs only 4MB of memory.
4. The version n.x.y is stable if x is an odd number
5. Login can be achieved by the user password alone.
6.. Logout can be achieved by entering both **logout** and **exit**.
7 . The user alone can change his/her password.
8 . In Linux, hierarchical structure is employed to make the search easier.
9. All users without any exception have $ prompt only.
10. One of the other terms for SA is super user.
11. To change from one directory to the other change directory command is used.

Answer the following

1. Who is the super user?
2. How will you change your current password?
3. What are the rules that you should follow when you change your password?
4. How will you know your working directory?
5. What are the privileges of the root user?
6.. How will you know a hidden file name?
7. How will you sort your files by size?
8. What are the essential conditions to remove a directory using **rmdir** command?
9. How will you execute a file in a floppy disk with the help o of **SA?**
10. How will you create a directory?
11. How will you display the files, directories and subdirectories and explain with options?
12. Explain the function of **man**.

3.7 File Commands

Displaying the contents of the file

The **cat** command lets the cat (contents of the file) out of the bag (file) but the **more** command does more. The **cat** command shows the contents of the   specified file normally on the screen. If the file is lengthy, it will run so quickly, what you see, in the end, is the last page. The command **more** is handy, in such situations. It will show one page at a time; if you want to move to the next page or the previous page you have to press **f** (**f**orward) or **b** (**b**ackward) keys respectively.Strictly speaking the **cat** and **more** are filters. They filter the data that pass through them.

There is one **tee** command, which does double the work of the **cat** command. The **tee** command is just like the **T** pipe. This pipe is made up of two tubes. The first part is a horizontal tube and the second part is a vertical tube. If water is allowed to flow through the **T** pipe, (placed in a horizontal plane) water flows horizontally and vertically simultaneously. In a similar manner the **tee** command takes the input from the standard input and displays the content on the screen (just like **cat** command) and stores the same in the file specified, after the **tee** command.

In Linux all files are arranged as a continuous stream of bytes. There is only one standard type of file in Linux,the byte-stream file.The input data stream is called as the **standard input** and the output stream is called as **standard output**. If you input the data, the data are converted into the data stream of continous set of bytes. This is called the **standard input.** Normally the standard input is connected to the key board. The **standard output** is also data stream of continous set of bytes.Normally the standard output is connected to the printer. You can redirect the standard input to the floppy disk etc. In a similar fashion standard output can also be redirected to a storing device such as CD, floppy.

You will be given examples on the above commands. Hereafter instead of giving the prompt as [ilamathi@localhost ilamathi] $, you will be given only the $ sign on command line.

Suppose you want to see the contents of the file1 on the screen, you should give the command as explained in the following example,

Suppose file1 contains the following text:
It is a fun.
You are encouraged to work with the Linux.

$ **cat** file1
The output is:
It is a fun.
You are encouraged to work with the Linux.

Suppose you want to see the contents of the file1 on the screen page by page, you should give the command as.

$ **more** file1
The output is:
It is a fun.
You are encouraged to work with the Linux.
From the above, you will not see any difference between **cat** and **more** commands. If the contents of the file1 exceeds one page, you can see the last page in **cat**, but you can see page-by-page in **more**

You will be given more examples for **cat, more, tee** after learning about **redirection** and **piping**.

In the absense of a proper rain-harvesting system, rain water collected on terraces of houses was usually let out on the streets. That is, the default connection of the terrace was the street. After the introduction of rainwater harvesting, people redirected the water collected, during the rain to the well or to the collection pits. Linux system can also redirect the output or the input to files other than screen or keyboard. (Linux considers the standard input, standard output, the screen and the

keyboard as files.) The redirection operator (**>**), "greater than**"** symbol achieves output redirection. The redirection operator (**<**), "less than**"** symbol achieves input redirection. The output redirection operator, redirects the contents of the left hand side file (that is the file name before the "**>"** symbol) to the file in the right hand side (that is the file name after the ">" symbol).

$ **cat** file1 displays the contents of file1 on the screen. Suppose you want to send (redirect) this output to file2 you have to give the following command

$ **cat** file1 **>** file2 .

$ **cat** file2 command displays the contents of file2 which is nothing but a copy of file1. The output of the above command goes to the standard output. Since there is no redirection the standard output is directed to the screen. But in the command $ **cat** file1 **>** file2, **cat** file1 sends the output to the standard output but the redirection operator (**>**) sends (redirects) output of the standard output to the file2. The redirection operator (**>**) prevents the output from going to the screen. If file2 does not exist, file2 will be created and the contents of file1 will be copied into file2. What will happen if the file already exists? The contents of file2 will be destroyed and the contents file1 will be copied. Actually there is something more than what the eyes meet. Even though **cat** file1 part appears before the **>** sign; only file2 part will be executed first. If file2 already exists it will be destroyed and then it will be constructed afresh. Then contents of the left hand side file namely file1 is sent to the standard output. The output of the standard output is taken as the input for file2. In short the contents of the file1 is copied into the file2. From the discussion you can conclude that commands like the following, will not work.

$ **cat** file1 **>** file1

When the above command is tried, since the file1 exists and the right hand side is executed first, file1 is destroyed. Then the left hand side is executed, now the file contains nothing. So, the command fails.

You can set the **noclobber** feature to prevent overwriting an existing file by the redirection operation. In this usage, overwriting of the existing

file will fail. But even this can be circumvented. To overcome the difficulty of overwriting the existing file append (**>>**) operator is used. The append (**>>**) operator adds the contents of the file, appearing left side of "**>>**" operator to the file appearing to the right side of the same operator, at the end of the existing material. For example the command $ **cat** file1 **>>** file2 appends the contents file1 to the contents of file2.

The Standard Input

Many Linux commands receive data from the **Standard Input**. The standard input is connected to either a device or to a file. By default the standard input is connected to the keyboard. The characters typed into the keyboard are taken to the standard input, which are then directed to the command.

The **cat** command without any argument takes the input from the standard input. You have to enter the data for **cat** command through keyboard, which is taken to the command through the standard input. Example
$ **cat**
                          .
This command expects data from the standard input   (Input from the keyboard)
This command expects data from the standard input   (output)

Now you have to enter the data from the keyboard   (Input from the keyboard)
Now you have to enter the data from the keyboard   (output)

Ctrl+D
$
The lines in normal letters are entered from the keyboard. At the end of the first line, the entered message is taken to the standard input from the buffer, which is directed to the **cat** command. Since there is no redirection operator, **cat** displays the message on the screen. The second line is the message sent by the **cat** command to the screen through standard output. The third line is the line entered by the user

145

and the fourth line is the response of the system. When you have completed your work, you have to inform the computer that you have finished your work. This is achieved by giving the command **Ctrl+D** in a separate line.

Note: Data can be compared to the water in a dam. Buffer is like a dam where water is collected before sending for irrigation (the standard output) when there is copious supply of water (data). It stores and sends water (data) in more orderly manner when there is a request for water or when the dam is full. **Ctrl+D** character is the end-of–file character for Linux file.

You can combine the **cat** command with output redirection operator.

Example:
$ **cat >** file3
The typed in material will be redirected
to the **cat** command through the std input   (Input from the keyboard).
and this message is sent to the file after ">"
symbol.
Ctrl+D
$ **cat** file3
The typed in material will be redirected
to the cat command through the std input
and this message is sent to the file after ">"
symbol. $

Input Redirection operator (<)

The data is normally sent to the standard input through the keyboard. You can make the standard input to receive data from files also. This is made possible by the redirection input operator. In order to make the **cat** command to get data from the file3 (not from keyboard) you have to give the following command at the prompt.
$ **cat <** file3
The typed in material will be redirected

146

**to the cat command through the std input
and this message is sent to the file after "＞"
symbol.**

Since there is no ＞ symbol the message is sent to the screen.
The redirection operator sends the contents of file3 into the standard
input. Then the **cat** command reads the standard input and displays
the contents of file3 on the screen. If the standard input is to be redirected
to receive its data from file3, and the standard output is to be redirected
to place its data in file4, you have to give the following command.
$ **cat ＜** file3 **＞** file4

Pipes

The redirection operator is completely helpless, if you want to send the
output of one command to another command. Redirection operator
works only on files. Pipe is handy in this situation.

Here you should understand clearly, the difference between files and
commands. Consider an example. Let there be a godown where you
store wood. You employ a carpenter to make chairs out of the wood
and a painter to paint the chairs. Now wood is converted into chairs
and the chairs are sent by the conveyor belt to the painter's place.
Here the chairs are taken from carpenter to painter. The conveyor belt
in this instant is a pipe. That is the goods are sent through a pipe (the
conveyor belt) from a worker (command) to another worker (command).
This is a piping operation.

After painting the chairs, they are sent back to the godown by a lorry.
This time lorry is used to carry the chairs from a worker to the godown,
to be stored there. This is redirection. File is like a strange warehouse
with the related items when its contents are sent, it will not become
empty, and it will get another copy. File is a storage medium (like
warehouse in the above example) to store the data whereas command
is a program (Just like a worker in the example) to execute a set of
instructions. You can save data into or retrieve data from a file. A

147

command may read from or save into a file, but a command itself cannot store the data. Redirection simply places output in a file, but pipes send output to another command. Suppose you want to have the contents of file to be printed. The **cat** command gets contents of a file and sends it to the standard output. The output of **cat** is piped to **lpr (** **l**ine **pr**int). The **lpr** command takes the standard output as input and sends it to the printer. Here you have two commands. As you have seen, redirection operator will not serve the purpose; you have to seek the help of pipes. The pipe receives the data from the command, placed before the pipe and sends the data as input to the command placed after the pipe. The piping symbol is the vertical bar "|". The above requirement can be met by the following command.

$ **cat** file3 **| lpr**

The contents of file3 are sent to the **l**ine **pr**inter connected with the Linux system currently.

You know already how to send the material through the keyboard to the screen. The **cat** command without any file name will do the work. What will you do to send the output to be printed on the line printer, instead of the screen? Ah! You   already know the answer! The combination of simple **cat** command and the **lpr**   command will solve your problem. Bear in mind that these two are commands, so you should combine them by | .Did you guess the answer? Yes, you are right. The answer is

$ **cat | lpr**
The printer should be made ready.
Interesting things are going to be printed.
Ctrl+D
The above message (excluding **Ctrl+D**) will be printed.
There arises another situation in which you are expected to print a file (say file3) along with line number, on the printer. This can be solved by the following   combination.
$ **cat –n** file3 **| lpr**
The **cat** command with the option **–n** sends the contents of the file3 to the   standard output after numbering each line. The option **–n** in the

148

presence of **cat** command numbers the lines of the contents of file3 and forces the **cat** command to send the numbered contents to the default printer. Now you are going to meet your old friend **more**. Already you have been told, if the file is lengthy and if you use the **cat** command, you can see only the last page. To overcome this difficulty the command **more** is used as follows

$ **cat** file3 **| more**
Suppose you need to display the contents of the file3 along with the line number, the above command can be modified as follows.
$ **cat –n** file3 **| more** .

You can have more than one file in the above command as follows.
$ **cat –n** file1 file2 file3 **| more**.
Note: Comma should not be used between file names; only blank space/ spaces should separate them.

There is another useful command, known as **sort** command, which sorts each line of the given file alphabetically and sends the sorted version to the standard output. You can send the sorted output to **more**, **cat –n**, **lpr** or to any of the suitable combinations of these.

Examples
$ **sort** file3 **| more**
$ **sort** file3 **| cat -** | **more**
$ **sort** file3 **| cat -n | lpr**
You have some acquaintance with the **tee** command. It plays dual role.

The **tee** command copies the standard output to a file. It takes as its argument the name of the new file to which the standard output is copied. It seems when the standard output sees the **tee** command; it will split into two copies. Normally, one of them is redirected to the file appearing after the **tee** command and the other goes to the screen. The following example not only copies file5 to file6 but also displays the contents of file5 on the screen.

$ **cat** file5 | **tee** file6

The sorted contents of the file can be copied into another file and also can be displayed on the screen.

Example:

$ **sort** file5 | **tee** sfile5

Here sfile5 gets the sorted version of the contents of file5 and sorted version is displayed on the screen.

The contents of file5 are not at all affected by the sort command. Only a copy of the contents of file5 is sorted. Suppose file5 contains the following text.
Chandran
Ashok
Malar

You can combine **sort, cat, tee** and **lpr** as follows:

Example:

$ **sort** file5 | **tee** sfile5 | **lpr**
The output is :
Ashok
Chandran
Malar
The above message is shown in the screen and also copied into sfile.
In addition to it, you can get a printed copy of the same.

$ **sort** file5 | **cat** **-n** | **tee** sfile5 | **lpr**
The output is :
1 Ashok
2 Chandran
3 Malar

The above message is shown in the screen and also copied into sfile. In addition to it, you can get a printed copy of the same.

You can interchange **tee** and **cat -n** commands. The result will change accordingly.

Copying file

While you deal with files, you will need frequently to copy the contents of one file into another. Even though you have been introduced to several sophisticated commands by means of which copying of files is made possible, you can do the same in a straightforward manner by the command **cp** (**cop**y).
Syntax
$ **cp** [options] <source file/s> <destination directory/file>
The word/words appearing within brackets will be taken as optional. You can include or exclude word /words appearing in the bracket. Here you can either include "options" or exclude them. The word/words appearing within angle   brackets ("< >") should be given compulsorily and actual names should be   substituted while the actual command is issued.
Now you want to copy the contents of the file1 into file6. The command is
$ **cp** file1 file6
The file1 is the source file (that is data emanate from file1) and file6 is the destination file (that is data go to file6). The above command will not affect the contents of the file1. If file6 already exists its contents will be overwritten by the contents of file1. The **cp** command copies the contents of source file after creating destination file. If the destination file already exists then the existing file is destroyed then a new file with same name is created. So, you should be vigilant about loosing contents of the destination file (if it exists already) in the copying process. You should add option **–i** in the above command for getting a warning from the system before overwriting, so you can stop the copying process.

151

Example:

$ **cp -i** file1 file2
overwrite file2 ? **n** $

If files are not in the current directory, then the full path should be given. If you want to establish a link between file1 and file2, you should replace –**i** by –**l**.

 You can also copy a directory recursively using **cp** command with the -**r** option.

Example**:**

$ **cp -r** alpha alpha1
This command copies all the files and sub-directories of the alpha directory to the alpha1 directory recursively. Here you do not have the danger of loosing the already existing data. If directory alpha1 exists already, all the contents are put inside the directory. If alpha1 does not exist it will be created and all the files and the sub-directories are stored. This alpha1 is created under the current working directory. You need not be afraid of the term recursion.

In Mathematics n! = n* (n-1)! and 0!=1 is given the definition for factorial (n). Factorial (1)=1*factorial (0)=1. Similarly factorial (2) =2*factorial (1)=2. In order to find factorial (n), we have to find factorial (n-1) then it should be multiplied
by n. This is an example for recursion.
There are two more options -**s** and -**v**. The option -**s** creates a symbolic link and the option -**v** (stands for verbose) explains in detail, what is being done.

Removing Files

To delete files or directories the **rm** command is used. This is superior to **rmdir**.

152

Example:

$ **rm** file1 file2

Now file1 and file2 are removed from your current directory. If the file/s is /are not in the current directory then the complete path name has to be given. You have already seen the command **rmdir,** is not of any help if the directory is not empty. But **rm** can be employed in such conditions with the **–r** or **–R** (for   recursion) option to remove the directory.

Example:
$ **rm -r** alpha1

The above command removes alpha1 directory along with its sub-directories. You can use **–i -v** with the usual meanings. There is one more option **–f**, you will be in a better position if you do not make use of it.

Wildcard entries and filename arguments

 If you have partial information about the names of files, the Linux provides special characters *, ?, [ ] (comma is not included in the list) which will help you to find out the exact name(s). if you want to list out the files which start with ch or end with .c, the special character * will be helpful to you.

Example
$ **ls**
main.c fact.c swap.c char1 char2.ex doc1 doc2

$ **ls** ch*
char1   char2.ex
$ **ls** *.c
main.c fact.c swap.c

The special character * stands for any number of characters
$ **rm** *
This command is a very dangerous command, which will wipe out all the files.
The question mark,?, matches only a single incomplete character in filenames.

$ **ls** char?
      char1

Note: the question mark, (?), fixes the number of characters. The name char? fixes the length as 5. While the first four characters are fixed, the last one may be any character including numbers.
Note: char2.ex will not be displayed, since the length of char2.ex is greater than 5 characters.

The bracket [ ] gives you a set of characters to search the file with them. Suppose you want list the files that start with doc and end with either 1 or 2, you should give the following command
$ **ls** doc[12]
doc1 doc2

Here the characters are 1 and 2, it will not be treated as 12. You can also set a range.

Example: doc[1-5] doc[a-g]
Here system may search for doc1, doc2, doc3, doc4, doc5.
Similarly system may search for doca, docb, docc, docd, doce, docf, docg.

Edit Text and Commands.
Before pressing the Enter key you can edit the command line with the help of left arrow (or **ctrl + B**) , right arrow (or **crtl + F**). Back space (**ctrl + H**) and Delete key are used to erase the character as usual. **Ctrl + U** deletes the entire line. You can enter more than one command in the same line but you should separate them with a semicolon (;).

You can also enter only one command in several lines by typing a backslash in each line.

Moving and Renaming the Files.

The **mv** (move) command is used for
1.   to move a file or directory from one location to another.
2.   to change the name of a file or a directory.

Note**:** Moving a file from one location to another is different from copying a file in that no file is created while moving a file.
Syntax: **mv**   [options] <source>   <destination>

Example:
$ **mv**   temp   temporary

Here the temp directory is renamed into a temporary directory. A file can be moved as shown below:

Example:
$ **mv** file1   /home/ilamathi/personal/file1

You can use **–i, -v** and **–f** options along with this type of commands.

Viewing the System Date and Time

You can view the system date and time by giving the command **date** after $ prompt.

Example**:**
$ **date**
Wed July 07 11:41:12 EST 2004

There are several options that can be used to format the date and time before displaying them. The options are specified within double-quotes, and within the quotes, they must begin with a +symbol.

155

The day, the month, the year, the date, the time in hours, in minutes and in seconds can be referred to, as shown below.

Option   Function

%d   Day of the month(in digits)

%m   Month of the year (in digits)

%y   Year(last two digits)

%D   Date as mm/dd/yy

%H   Hour(00 to 23)

%M   Minutes(00to 59)

%S   Seconds(00 to 59)

%T   Time as HH:MM:SS

%a   Abbreviated weekday(sun to sat)

%h   Abbreviated month(jan to dec)

%r   Time in the AM/PM notation

Options of the date command

Note: You should be very careful about %m and %M, the first one represents month of the year in number, the latter represents minutes in numbers. Similarly, you should be careful about %h and %H .The %h stands for abbreviated month but %H stands for hour You will see an example using **date** and **%D** when you see the grave accent.

Example:
 $ **date "+%m"**
7
$ **date "+%D"**
07/07/04
$ **date "+%T"**
11:43:14

Note: With the help of the above options, the SA can change any part of the **date** command.

3.8   File Systems: **mount** and **umount**

 All the files in your Linux system are connected into one overall directory tree; the files may reside on various storage devices such as hard disk drives, floppies and CD-ROMs. The Linux files on a particular storage device are organized into a file system. Your Linux directory tree may include several file systems, each on different storage devices. The files themselves are organized into one perfect tree of directories beginning from the **root**. Although the root may be located in a file system on your hard drive partition, there will be a path name to files located on the file system for your CD-ROM and floppy.

A floppy disk with Linux files will have its own tree of directories. This tree is a sub-tree, detached from the main tree. Your school is just like a sub-tree. In order to function smoothly, it should be attached with the Director of School Education. But, your school can take some action without consulting the Director of School Education, the floppy or even CD-ROM completely dependent upon the **root** directory. If you want to access the contents of the files, in the file system, you should start from the **root** directory. For that you have to connect the sub-tree to the main tree. Until it is attached, you will not be able to access the files on your floppy disk. This applies to all the storage mediums unless they are connected already. Even the file system on your hard disk partition has to be mounted with a mount command. But the system takes care of this activity.

Establishing the connection between a file system on a storage device and your main directory tree is called mounting the device. This is done with the **mount** command. You can then change to that directory and access those files. The main drawback is that the root user can alone do the mounting operation. Even though this appears as a handicap, it protects the integrity of the system. You may observe that the command line prompt is changed into # from $.

Naturally the **mount** command should take two arguments. One of the arguments is the storage device such as floppy disk, through which Linux accesses the file system. The other one is the directory in the file structure to which
the new file system is attached. Suppose "destination" is the directory on your main directory tree where you want the files on the storage device to be attached. The "**device** " is a special device file that connects your system to the hardware device. The syntax for the mount command is as follows:
\# **mount** device destination

Device files are located in the /**dev** directories. They usually have abbreviated names, ending with the number of device. For example, **fd0** (the last character of **fd0** is zero (not the letter o)) may reference the first floppy drive attached to your system. Similarly fd1 may reference the second floppy drive (if any) attached to your system. On Linux systems operating on PCs, the hard disk partitions have a prefix of **hd** followed by an alphabetic character that labels the hard drive and then a number for the partition. For example, **hda2** references the second partition on the first hard drive. The letter **a** stands for the first hard drive, the number 2 refers the second partition.

For a file system to be accessible, it must be mounted. Floppy disks and   CD-ROMs, however, have to be explicitly mounted. The following example mounts a floppy disk in the first floppy drive device (**fd0**) to the /destination directory.
\# **mount** **/dev/fd0** /destination

The mounted file systems should be unmounted either before you shut down your system or before you want to replace a mounted file system with another. Your main file system is automatically unmounted for you, as already stated. Assume you have mounted a floppy disk and now you want to take it out and put in a new one. First, you must unmount that floppy disk. Next you remove the floppy and you can put in and mount the new one. You unmount a file system with the **umount** command (Note the spelling there is no n between u and m). The

command **umount** can take as its argument either a device name or the directory where it was mounted. Here is the syntax:
umount device (or destination)

The following examples unmount the floppy disk mounted to the / destination directory:
 # **umount /dev/fd0**
umount /destination

There is one important constraint on the unmount command. You can never unmount a file system that you are currently working in. You will never cut the bottom of the branch of a tree while sitting on the same branch.

Mounting and Formatting Floppy Disks

If you want to read a book you should first have the book. In a similar manner if you want to access the contents of a file on a floppy disk, first of all you should **mount** it. As already stated **/dev/fd0** references your floppy drive. You can **mount** any directory of your choice. However, your OpenLinux installation already created a convenient directory, to use your floppy disks which is **/mnt/floppy** directory. The following command **mount**s the floppy in the system.

Note : **mnt** stands for **mount.**
mount /dev/fd0 /mnt/floppy

The system tries to read the files in your floppy disk. If you change a floppy by another one without unmounting the first floppy you will get an error message. If you want to replace a floppy disk by another one, you have to unmount the floppy in **/dev/fd0** and then explicitly mount the new floppy as follows:

umount /dev/fd0
          or
umount /mnt/floppy

159

and
mount /mnt/floppy

Note: The **mkfs (m**ake **f**ormatting**s)** command formats a floppy.

Mounting CD-ROMs

If you want to mount a CD-ROM disk, you are not expected to specify the device name. The OpenLinux system has the directory **/mnt/cdrom** for CD-ROM file systems. The following command **mount**s a CD-ROM.

mount /mnt/cdrom
If you want to change a CD-ROM disk by another one, first you have to unmount the existing CD-ROM disk and then mount the new CD-ROM as follows:

umount /mnt/cdrom # You can interchange the CD-ROM.
mount /mnt/cdrom
If you want to mount a CD-ROM to another directory, you have to include the device name in the mount command. The following example mounts the disc in your CD-ROM drive to the /destination directory. The particular device name for the CD-ROM in this example is **/dev/hdc.**
mount /dev/hdc /destination.

Summary

Some common file-handling commands are:
. **cat**   -Displays the contents of files
. **more**   -Displays the contents of specified file page by page
            and you can move forward or backward by using **f** and
            b respectively.
. **tee**   -Displays the contents of the file on the screen and copies
            into the specified file
. **pipe**.   -Takes data from one command to another command.

. **>** operator -Takes data to the file.

- **<** operator -Takes data from the file to the command.
- **cp** - Makes copies of files
- **rm** - Removes a file or directory.
- **mv** - Moves or renames files and directories.
- **mount** - Establishes the connection between a file system on a storage device and your main directory tree.
- **mount** -Command should take two arguments.
- **umount** -Unmounts a file system.
- **fd0** -References the first floppy drive attached to your system.

Some other commonly used commands are:

- **date** - Used to view and change the current system date and time
- **tput clear**- Clears the contents of the screen
- **tput cup** - Used to position the cursor on a specified row and column
- **man** - Displays help on any Linux command

Fill in the blanks

1. Files can be copied directly by _____ command.
2. If you want to get help for a particular command, the _____ command will provide you help.
3. The _____ command shows the contents of a big file page by page.
4. The _____ option in **cp** command warns you from overwriting the destination file.
5. $ **cat** file1 _____ file2 appends the contents file1 into file2.
6. _____ feature can be set to prevent overwriting an existing file by the redirection operation.
7. End of file is given by _____.
8. Mounting a device means _____ the connection between a file system on a storage device and your main directory tree.

161

9. Although the root may be located in a file system on your hard drive partition, there will be a _____ _____ to the files located on the file system for your CD-ROM.
10. The/a _____ _____can only do the mounting operation.
11. All the files in your Linux system are connected into one overall _____ tree.
12. A floppy disk with Linux files will have its own _____ of directories

State whether the following are True or False

1. The **ls** command displays the contents of the files.
2. The **mv** command moves a file into another location.
3. **tput cup** 20 20 is equivalent to **gotoxy** (20,20) of C programming.
4. **echo – n** " This is nice ", command will print, This is nice and 0the cursor is taken to the next line.
5. **gets** (myname) of C programming is equivalent to read myname of
   shell script.
6. The **cat** command displays the contents of a file page by page.
7. The $ **cat** file1 > file1 overwrites itself.
8. The $ **cat** file1 > file2 works only when both file1 and file2 exist.
9. The **lpr** command takes the standard output as input and sends it to the screen.
10. The **mount** command should take two arguments.
11. The command **umount** can take as its argument either a device name or the directory where it was mounted.

Answer the following

1. What is the difference between the commands **rm-r** and **rmdir**?
2. How will you display your name like My name is x ?
3. How will you delete a directory along with its sub directories?

162

4. What does **cat** command do? Write and discuss all the variations of**cat** command.
5. Distinguish between pipes and redirection.
6. Distinguish between **mv** and **cp** commands.
7. How will you copy contents file1 into file2 in different ways?
8. How can you copy a directory along with all files in the directory?

3.9   VI EDITOR

Editors are mainly used for creating, deleting, and editing the files. There are several editors in Linux and some of them are highly sophisticated but all systems have two standard editors, they are Ed and Vi editors. Ed allows the user for one line editing only, so it is not of much use. Vi editor allows the user to edit text of one screen at a time. So, Vi editor is still widely used. When Vi editor was introduced in Unix, it provided lot of facilities, which the other editors could not even dreamt of, at that time.

Editors use the keyboard for two entirely different purposes. They are

(1)   to specify editing commands and
(2)   to receive character input.

Common PC editors divide the two functions among the keys of keyboard; alphabetic character keys are used to input whereas the functional keys and control keys are used to edit commands. Such PC editors can rely on the extended keyboards and the number of keys is ever increasing.

But Unix and hence Linux are frugal in many respects. Any keyboard can be used in the Linux system. Editors in Linux were designed to assume a minimum number of keys with alphabetic characters; some control characters as well as ESC and ENTER keys. How can Vi Editor manage its affair with minimum number of keys?

You behave as a student in your school but you also behave as a son/ daughter at home. You act in a formal way when you are at school but you act in an informal way when you are at home. The same person plays a dual role (actually more number of roles). In a similar manner Vi editor makes the keyboard to play a dual role. Vi editor has got two modes. They are

(1)   the command mode and
(2)   the input mode.

In command mode all the keys on the keyboard become editing commands. In the input mode, the keyboard behaves as a normal typewriter with the exception

When you change the mode, the functionality of the keyboard also changes. For example when in the command mode, the key **x,** just like the **delete** key of your ordinary keyboard, erases the character where the cursor is on. In the input mode the same key **x** simply adds **x** to the file. The ESC key is the only exception. While you are in the input mode, if you press ESC key, you will be taken to the command mode but if you are in the command mode itself, there is a beep sound. This activity can be of much help to a novice user of the Vi editor. If you have doubt about the mode in which you are working, press ESC key. If you hear a beep sound, you can conclude that you are in the command mode and if you do not hear the beep sound, you can presume that you are in the command mode at present, but previously (before presing ESC key) you were in the input mode. So ESC key eliminates lot of confusions.

Even though Vi command mode handles many editing operations effectively, it cannot perform actions such as file saving. The line editing commands handle these actions. When you are in the command mode (the colon) **:** takes you to the line-editing mode. After completing the necessary actions if you press ENTER key that will take you again to the command mode. In addition to the modes that you already know, you have been introduced to the line mode also.

Creating, Saving, Editing And Quitting a File in Vi

If you want to edit an existing file or to create a new file with name student, you have to give the following command.
$ **vi** student

If the file already exists, you will be shown the contents of the file page by page on the screen. If the file does not exist then you will see an empty screen and a column of tildes at the left hand side. Whether the file exists or not, you will be in the command mode only. Tildes inform that part of the screen is not in the file. If you want to input data, you should change to the input mode. By pressing **a**, **i** or **o** you will be taken to the input mode, after entering the data, pressing the ESC key will take you to the command mode. Then you press upper case **ZZ** (without space) (You hold down the SHIFT key and press **Z** twice when caps lock is not in effect). This action saves your file and then control exits the Vi editor, to return to the Linux shell. When you are in the command mode, you should save and exit from the file. But, while you input the file, you have to save it frequently. You have to press ESC to change to the command mode and then you have to press **:** (colon) to go to the line editing mode. Then press **w**. This sequence of actions will save the file, and you will return to the command mode. The command **:w** with a file name saves the file with the given name. It works as a "save as" command in the other word processors.

You can create unnamed file as follows
$ **vi** (Creates a file without a file name.)

When you create a file without a file name, there is no actual file. The entered data enters the buffers but you can input, edit on the matter that you have created on the buffer. Now you cannot save the file with **ZZ** command. The **ZZ** command saves the file when the file actually exists. **ZZ** will not save the file but **:w some-name** will save the file. In this case the contents of the buffer are copied into some–name file. Clash of file names will not be tolerated. If you give a file name, which already exists, the name that you give will be rejected. You have to retry with another name. If you want to quit Vi editor, if you give **:q** command it will take you to the shell but if you have made one or more

165

changes then this command will not work. In this case **:q!** will take you out of Vi editor without saving the changes.

Cursor movement

You can use the arrow keys for moving into the text but if you want to strictly follow the character keys alone h, j, k, l will solve your problem

h	=	left arrow
l	=	right arrow
j	=	down arrow
k	=	up arrow

The keys h, j, k, l lie in the middle row of the keyboard and they are closer to your right hand. Among these four keys h is the left most key and l is the right most key, that is why, h is used as substitute for ß (left arrow) l is used as substitute for à (right arrow).
You can also use Enter for h and spacebar for l.

Each text line begins at the left most column of the screen and ends where you press the Enter key. The space between the end of the line and the end of the screen is called the dead space and will not be stored in the file. You can move with h and l keys within the line. If the cursor is at the end of the line and if you want to move to the end of next line, use j key and if you want to move to the end of the previous line press k. You can move through the text, a whole screen at a time, using **ctrl+F** and **ctrl+B** key combinations. **ctrl+F** moves one screen (F) forward and **ctrl+B** moves one screen (B) backward at a time.

Line number G

Vi sequentially numbers each line of text using that line number; you can go to any line by entering the line number followed by (upper case) G. If you want to move to the end of a file, you enter G without giving line number.

If you want to set the word wrap margin you can do so by
 :set wm=col (col should be replaced by a suitable number)

When you press the **a** key, Vi editor places you into the input mode, after the character where the cursor is currently on. The **i** key places you into the input mode before the character where the cursor is on. The (lower case) o key opens a new line below where the cursor is on and places you into the input mode at the beginning of that new line.

Deletion

x key, in the command mode deletes a single character. As already stated, it acts as the **delete** key of your ordinary keyboard. Repeated use of x will delete the desired number of characters.

The **dd** command removes the entire line that the cursor is presently on. A number immediately followed by **x** or **dd** removes that much number of characters or lines from the file.

 Suppose you want to delete 5 characters from the position of the cursor, you have to give the command as 5**x** when you are in the command mode. Suppose you want to delete the line where the cursor is on, simply give the command **dd** in the command mode. The line is erased. If you give 5**dd**, 5 lines starting from the line where the cursor is on, are erased.

Undo

The **u** command will undo the last modification.
Break a line

To break a line, you have to enter into the input mode and press the Enter key. To join two lines press (upper case) J key.

167

Moving

Suppose you want to move a certain part of the text, you have to delete those lines by using **ndd** (Where n is the number of lines to be moved). This statement is somewhat equivalent to cut facility of WINDOWS XP PROFESSIONAL.The deleted lines will be placed in the buffer. Here buffer is used as clipboard of WINDOWS XP PROFESSIONAL Next you move the cursor where you want to move the text, press **p**, then the editor inserts the deleted lines after the line, on which the cursor is on. This is equivalent to the paste facility of WINDOWS XP PROFESSIONAL.

Copying

You can copy a line by **yy** command. If you want to copy n lines then nyy copies n lines starting from the where the cursor is on into the buffer. Then you have to paste the text in the buffer to the desired detination. Move the cursor till the desired detination and press p key. Copying is succesfully done. You already know the difference between moving and copying from Windows XP. As you did for moving, you can copy with **nyy**. But here the text is not deleted.

Searching

You can search any pattern within the text. Suppose you want to find out the occurrence of a particular word or any pattern, you have to use either **/** or **?.**. The (/) allows you to search the pattern, forward in the text. The **?** allows you to search the pattern backward in the text. When you press **/** key a line opens at the bottom, the character **/** appears in the first column. The cursor is immediately placed after it, you should key in the pattern and press the ENTER key. The search will start functioning from the position of the cursor, where the cursor had been before the **/** key was pressed and the search continues upto the end of file. The search goes on in the forward direction.
The **?** mark does the same thing as that of **/**, but in a reverse direction. The search does its function from the position of the cursor, where the

cursor had been before the **?** key was pressed and the search continues upto the beginning of the file. The search goes on in the backward direction.

Changing

The **cc** command allows you to change the entire contents of a line. First it erases the line and changes to the input mode. So, you can enter the fresh line and then press ESC key. It is the combination of **dd** and **o**. The **r** (**r**eplacement) command allows you to change a single character where the cursor is currently on. Unlike the other commands it will not take you to the input mode. After typing in the replacement character, you remain in the command mode only. The **R** (**R**eplacement) command allows you to overwrite text. It is similar to the overwrite command of the other text editors but it allows you to change the text because you are in the input mode and you have to change that mode by entering ESC key. It differs from the other text editors in this aspect.

What you do to a line with a **cc** command, you can do to a word by simply replacing the last character namely c by w. That is, the command **cw** allows you to change a word. The command **dw** deletes a word.

Now let us apply this newly acquired knowledge to the following text.

Having computer knowledge is an invaluable asset, it will do a world of good to young people seeking jobs.

1   **Reliability:** Linux is a highly reliable system. Linux servers are not shut down for years together. Normally operating failures are unknown to Linux systems. It does not mean that you need not be vigilant. Do not forget the computer adage. If something can go wrong, it will.

2   **Backward Compatibility:** Linux has excellent support for older

hardware. It can run on different types of processors including the older ones. It can run the commands of its ealier version successfully.

3. **Simple Upgrade and Installation:** The installation procedure of most Linux versions is menu driven and easy.

4. **Suitable to any machine:** Suitable Linux version can run on any machine available now. This allows low investment for the hardware. The users, who have low configuration machines, prefer to use Linux OS compared to other OSs that require higher configurations.

You should make the third point as the first point. Bring the cursor to the beginning of the number 3. If you are not in command mode, press ESC key. Now issue the command 2dd. These two lines are deleted from the screen and stored into the buffer. Now go to the second line. For going to the second line you should issue the command 2G. Now press (lower case)p. You want that point 3 to appear in a seperate line.Press (lower case) a key to change into input mode. Then press ENTER key. Point 3 appears in the third line and then press ESC key to return back to the command mode. But you have to alter the point numbers. Place the cursor under number 3 and press (lower case) r key. The number 3 is deleted, but the cursor waits for the replacement. Enter the number 1. You will not be taken into the input mode. Again bring the cursor to the number 1 and change it into 2. Similarely change the number 2 into 3. Now you want to enter the incomplete statment "I want to stress the point that " at the end of the text. Press G key, you will come to the end of text and again press o key, In the ensuing blank line, enter "I want to stress the point that " (without the "). Go to the beginning. Issue the command 2yy. Now those two lines are copied into the buffer. Again issue command G that will place the cursor after the space after the letter t then press p key. Then those two lines are copied after the word "that". now you have to change H of having into h. Place the cursor under P then press r key. The H is deleted and the cursor is waiting for your command. Enter the h.

Having computer knowledge is an invaluable asset it will do a world of good to young people seeking jobs.

1   Simple Upgrade and Installation: The installation procedure of most Linux versions is menu driven and easy.

2   Reliability: Linux is a highly reliable system. Linux servers are not shut down for years together. Normally operating failures are unknown to Linux systems. It does not mean that you need not be vigilant. Do not forget the computer adage. If something can go wrong, it will.

3   Backward Compatibility: Linux has excellent support for older hardware. It can run on different types of processors including the older ones. It can run the commands of its ealier version successfully.

4   Suitable to any machine: Suitable Linux version can run on any machine available now. This allows low investment for the hardware. The users, who have low configuration machines, prefer to use Linux OS compared to other OSs that require higher configurations.

3.10 Shell Script

A shell script is a text file that contains Linux commands. You can create a file by using any of the standard editors such as Vi editor (which you are going to study at the end of the chapter). Suppose you want to repeatedly execute a set of Linux commands, in the same order then you will seek the help of shell script. Entering the commands in the command line sequentially is not only a frustrating job, but also an exacting one. Either you may commit mistakes in spelling or you may change the order of the commands. In either case, you will run into difficulties. A shell script is handy in these circumstances because you have to enter the command only once. Shell scripts allow input/output operations and manipulation of variables.

Executing a Shell Script

When you Logon to the Linux system, you get a copy of the shell to work with. This shell is known as the Login shell. Your default shell is BASH shell. The BASH shell has the capabilities of the programming languages. You can create complex shell programs with its capabilities. A shell program combines Linux commands to solve the given problems. The Linux shell provides many of the tools found in C language. You can create variables and assign them values. You can also create variables in a script file, which can be assigned values interactively by the users. By giving **sh** command in the command prompt, a new shell is created. This new shell is known as the sub-shell or the child shell of the current shell, which can be used to execute a shell script. The shell script is passed to the child shell for execution. This arrangement makes the Login shell impervious to the whims and fancies of the user. If any undesirable event   happens, only the child shell is affected and that may be deleted immediately without causing any damage to the Login shell.

You should create the shell script carefully. When you create a file, you will have the **read** and **write** privileges but you will not be granted **execute** permission automatically. Even with these limited capabilities you can run the shell program with either of the following commands in the command prompt.
$ **sh** file_name
$ **.** file_name

If you want to run a shell script directly at the $ prompt, you can change the File Access Permission (FAP) of the specified shell script by granting the **execute** permission. This can be achieved by **chmod** command. Suppose you want to run the edufile directly from the $ prompt the following commands should be given.
$ **chmod u+x** edufile
 $ edufile
 +x command in tandem with **chmod** gives the **execute** permission to any user. The **u+x** command gives the owner of the file the **execute**

172

permission. When you execute the above shell script, the current shell creates a new shell and executes the script in the newly created shell.

3.11 Variables

Variables are placeholders to store values. All Linux variables are treated as character strings. This may seem that you cannot do any mathematical operations with those variables. However, this limitation can be overcome by **expr** and **let** commands.

Creating Variables

As already stated, the BASH shell is your default shell; you have to do your work in BASH shell only, unless you desire to change to some other shell. The variable created within a shell is called a shell variable. Variables can be created as and when the user wants to create them by simple assignment of values. A variable can be created without a value being assigned to it by leaving the right-hand side including the assignment operator.

The variable name in shell script may consist of alphabetic characters, the underscore and a number.
It can not include the exclamation mark (!), the ampersand (&) or the blank space.
The number should not be the first character.
It should not be of unreasonable length.
Command names should not be used as variable names.
Valid script variable names: file1, bookshell, book_shell, a+b, rs-paise
Invalid script variable names: a + b, a+ b, a!b, ab&, a=b.

 The syntax for creating a variable is given below:
<variable_name>=<value>
Note: When declaring a variable, there must be no space on either side of the assignment operator (=). It is like the assignment statement of C programming. If you leave blank space before and after the "="

173

operator it is like "==" (equality) operator of C programming.

If the value being assigned contains any delimiters (such as embedded spaces), then it should be enclosed within either single or double quotes. From the above statement it may seem that single quote and double quotes can be used interchangeably. But, there is a subtle difference, which you will see later.

Example:
name= "Ezhil kumaran"

You can also write the above command as

Name='Ezhil Kumaran'

If the value does not have spaces, the quotes are optional.
Example:

name=Ilamathi
name='Ilamathi'
name= "Ilamathi"
All the above are equivalent.

Consider the following assignment
number=12

In the above assignment, the variable number, though in the form of the number, it is not a numeric value. It is a character string. Therefore, the variable number contains the character '1' and '2' not the number 12 = 1100 in binary form. So, you cannot do fundamental operations of algebra, that is, you cannot add, subtract, multiply or divide.

Referencing Variables

The $ symbol is used to refer the contents of a variable. The $ sign extracts the contents of the variable following it. Consider the example

var1=${var2}. The variables var1 and var2 refer to the memory locations. ${var2} command extracts the value found in that location. The copy of the obtained value from that location is stored in var1. The braces are optional. But, if you want to concatenate the contents of one variable with another value, the braces are essential. For example the variable **father** contains the value John and if you want to add son to John and store the result in **son1** variable, you should give the following command

$ son1=${father}son

then the contents son1 will be Johnson.

What will happen if you omit the braces? The result is obvious. Without braces the command will be

$ son1=$fatherson

The first $ is the prompt and the second $ is the reference operator.

Since there is no blank space in between father and son, fatherson will be taken as a variable, and if such variable does not exist, you will get one type of error. Unfortunately if that variable exits then that value will be assigned without any warning .The result is an unpleasant surprise (If you leave a blank space in between father and son the variable will be rejected).

Reading a Value into a Variable

If you want to get the name from the user, you should enter
"please enter name"
Then you should make arrangements to store the entered name into the memory.
The above can be written in Linux as follows.
echo " Please enter your name"
read name

The **echo** command simply prints the string on the screen. This serves as a prompt for the user. The **read** command, on execution, waits for the user to enter a value for the variable. When the user presses the

<Enter> key, after entering the value, the remaining part of the shell script, if any, is executed.

The response of the user is caught into the variable name (called) **name**. The read command can be used at the shell prompt, but is usually used in shell scripts.

Note: As already stated, the double quotes ("") improve the readability.

3.12   Expressions

The **expr** and **let** Commands

For any individual, a day hardly ends up without doing any arithmetic calculation. Can you imagine a world without arithmetical calculations? The answer is an emphatic "no". Most shells do not support numeric variables. All variables are treated as character strings. However, to program in the shell, it is imperative that you will be able to mathematically manipulate variables. This is possible by the use of the **expr** and **let** commands. The **expr** command is used to evaluate arithmetic expressions. The output of this command is sent to the standard   output (screen).

Example:
$ **expr** 21 + 51
will display 72 on the screen. Note that there must be a space on either side of the operator (+). Variables can be used in the **expr** command such as
$ num1=7
$ num2=3
$ **expr** $num1 + $num2
Since output is sent to the screen, the value 10 is displayed on the screen.   Remember $ when it is not used as the user prompt, is used to refer the value of the variable.

Therefore $num1 is replaced by character '7' and $num2 is replaced by   character '3'. The command **expr** then converts these characters into numbers and the addition is done afterwards.

176

The **expr** command supports +, -, *, and /. But you should be careful when using the * operator. Since * is used for wildcard character, it should be distinguished for multiplication operation. If you write *, then this will be treated as multiplicative operator.

$ **expr** 1 / 2   and will display 0 and not 0.5. Please note the blank space before and after the / sign. What will happen if you give the following command?

$ **expr** 0.5 / 2

What you get is an error message. Since the decimal point will be treated as **dot,** 0.5 will not even be treated as a number.

The command **let** lets you do arithmetic calculation and compare two values. The syntax for **let** is

$ **let** <value1><operator><value2>

Here the operator stands for either the arithmetic or the relational operator. The command **let** is an improvement over **expr.** The command **let** evaluates any variable and converts its value into an arithmetic variable. This capability is utilized in shell script to manage control structures. While **expr** needs blank space/spaces   before and after the operator, the command **let** demands no blank space/spaces either before or after the operator. If you want to leave blank space/spaces then you can do so by enclosing the entire expression within quotes. If you do not assign the result of an operation of the **let** command, the result is displayed on the screen.

Example:

$ **let** pr=5*10

echo "The product is $pr"

The product is 50

Note: The operator * (multiplication) should not be entered as *.

Note : Unlike **expr , let** should have variable name in the left hand side of assignment operator.

If you want to leave blank space/spaces before and after the operator, you should enclose the entire operation within quotes.
Example:
 $ **let** " pr = 5 * 10 "
echo "The product is $pr"
The product is 50
Suppose you assign, the result of an operation to a variable, the result will not be displayed on the screen, whereas the result is assigned to the variable. If you want to see the result on the screen you should use the **echo** command.

Example**:**
$ **let** " sum = 2 + 4 "
$ **echo** "The sum is $sum"
The sum is 6

The following assignments are also possible in script programming by using the **let** command as follows.

let a=0
let a=a+1
Note: Alas, **let** also fails with decimal numbers such as 2.3, 0.5 etc..
You were told that there is subtle difference between single quotes and double quotes. You will now see that difference. The variable **name** contains Ilamathi. Suppose you give the following commands

echo "The given name is $name"
echo 'The given name is $name'

what you see on the screen are

The given name is Ilamathi
The given name is $name
Double quotes actually references the variable. $name is replaced by its contents namely by Ilamathi. But single quote, simply reproduces whatever is found within it. That is, $name is reproduced as such.

3.13   Command Substitution

Suppose you want to access the current date of the system within a message, $**date** will not be of any help (reference can extract values only from variables not from commands). If you want to extract data from a command you should place the command within backward quotes (or grave accent). In this condition, you should place date command within single backward quotes (or grave accent). The grave accent key is found normally ahead of the number 1(or "!"). The backward quote is found under ~ (tilde) sign. If you want to display today's date, you should give the following statement.

Example**:**
echo "Today's date is `date +%D` "
When message displayed `date +%D` is replaced by system date. You better note that there is a blank space in between date and + sign. Otherwise the above command will not work. (The +%d symbol extracts the date part alone in the format mm/dd/yy). That is, the shell first replaces the enclosed command of the output, and then executes the entire command. Command substitution can also be used to store the output of a command in a variable.

Example **:**
      cfiles =`**ls** *.c | **wc** –l`
The variable, cfiles, will now contain a count of the number of files in the current directory whose names end with .c.
The output of **expr**, as stated earlier, goes to the standard output. If you require this output to be stored in a variable instead, you can use command substitution. For example:
$ varl=5
$ varl=`**expr** $varl + 20`
would assign 25 to var1.

The command "**expr**" is in single backward quotes. The shell first replaces the enclosed command of the output, and then executes the rest.

You can make use of commands such as **if, for, while** available in other higher level languages in **Advance Shell Script**.

3.14   Features of Linux

Reliability: Linux is a highly reliable system. Linux servers are not shut down for years together. Normally operating failures are unknown to Linux systems. It does not mean that you need not be vigilant. Do not forget the computer adage. If something can go wrong, it will.

Backward Compatibility: Linux has excellent support for older hardware. It can run on different types of processors including the older ones. It can run the commands of its ealier version successfully.

Simple Upgrade and Installation: The installation procedure of most Linux versions is menu driven and easy.

Suitable to any machine: Suitable Linux version can run on any machine available now. This allows low investment for the hardware. The users, who have low configuration machines, prefer to use Linux OS compared to other OSs that require higher configurations.

GUI Interface: The graphical interface for Linux is the KDE, GNOME. It is divided into two sub systems consisting of a server and a client. The KDE, GNOME provides nearly all the comforts of the Windows 98 system.

Multiple Distributors: There are multiple distributors for Linux. Each one   provides one's own added facilities. This results in the buyers market. Some distributors of Linux are Red Hat, Caldera, Mandrake, Debian, and Slackware**.**

No Virus Attack: Virus is the most dreaded word in the Computer industry. Virus actually decelerated the spread of the internet. Linux is said to be free of any virus attack. It is rumoured only now that a kind of virus attacks Linux system also.

Security Features: Internet mischief mongers play havoc on other people's work. Linux provides excellent security features. This is the reason why many Internet Service Providers (ISPs) switch over to Linux systems.

Can Support a High User Load: Linux can support a large number of users working simultaneously.

Development Libraries: Linux offers an excellent platform for many development languages like C++ and Perl.

Summary

- ♦   (n**yy** or n**Y**) **p** Commands to copy n lines.
- ♦   Shell scripts can be created in Linux using any text editor
- ♦   The **expr** command is used to evaluate arithmetic expressions.

♦ The **let** command is superior to **expr** command.

♦ The Vi editor can be invoked by the **vi** command

♦ The Vi editor works in two modes, the input mode and the

♦ command mode. The <ESC> key is used to determine in which mode the user is currently in.

♦ In addition to the above modes, you have been introduced to the line mode also.

♦ Command substitution is used to have more than one command execute as a single command

Fill in the blanks

1. $ **vi** filename. The file is saved by entering two upper case _____

2. **:w** means _____ and _____ the file.

3. When you press ESC, if you hear a beep sound, you are in _____ mode.

4. The command n**dd** and _____ moves a specified text to the desired destination.

5. The _____ command does arithmetical calculations more efficiently.

CHAPTER 4

PROBLEM SOLVING TECHNIQUES AND C PROGRAMMING

4.1. Problem Solving Techniques

The computing we do is independent of the computer language we use. It does not depend on the computer also. The choice of the computer and the language mostly depend on their suitability for the given situation. While developing a program, we think only in the natural languages like Tamil and English. Only at the last moment the design is converted into a program in a high level language.

In the computer languages every statement must be written precisely, including commas and semicolons. One has to be very careful while writing these lines. In the natural languages the sentences may be long. Sometimes they may be vague. All the natural languages have this property. So, to understand things clearly without any ambiguity, we write it in an intermediary language. This will be easy to write and understand, and also without any ambiguity. These intermediate languages are in between the natural languages and the computer languages. We shall study two such intermediate languages, namely, the flow chart and the pseudo code, which are widely used.

First let us consider the flow chart. Since the flows of computational paths are depicted as a picture, it is called a flow chart. Let us start with an example. Suppose we have to find the sum and also the maximum of two numbers. To achieve this, first the two numbers have to be received and kept in two places, under two names. Then the sum of them is to be found and printed. Then depending on which one is bigger, a number is to be printed. The flow chart for this is given in flow chart 4.1. In the flow chart, each shape has a particular meaning. They are given in flow chart 4.2.

Flow Chart 4.1

Start, End

Input, Output

Calculations

Decision

Direction of control flow

Connector
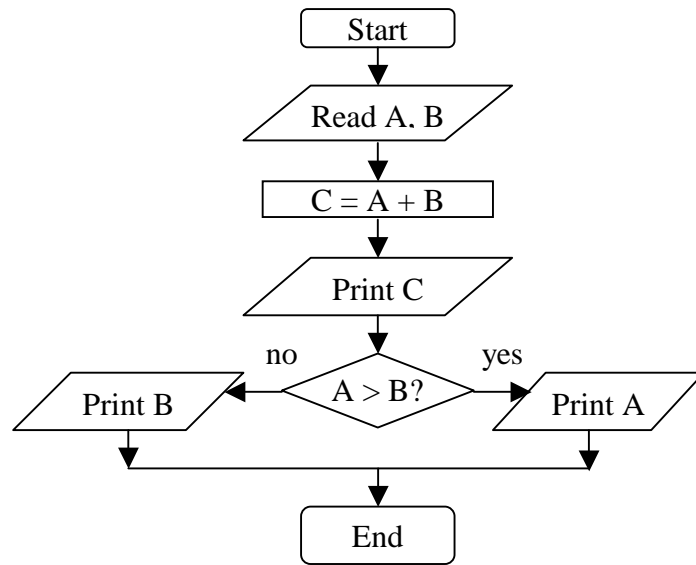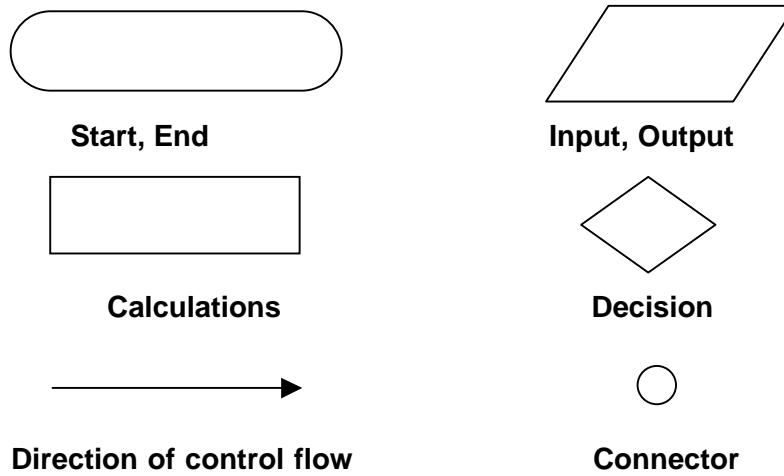
Flow chart 4.2

In the flow chart mentioned above, there is a special meaning in writing   C = A + B. Though we use the familiar equal to sign, it is not used in the sense as in an equation. This statement means — Add the current values available for the names A and B, which are on the RHS, and put the sum as the new value of the name C, which is in the LHS. For example, under this explanation, A = A + 1 is a valid statement. The value of A is taken, incremented by one and the new value is stored as A. That is, the value of A gets incremented by 1. Note that we can write A = A + B, but not A + B = A. On the LHS there must be only a name of a place for storing.

We can write all the computations we can do with the computers as flow charts. If the problem is small, the flow chart is also small. What about big problems? Real life problems are always very big. Only the class room problems are small, as they are meant to teach some particular concepts within limited time. For big problems, the flow chart will also be big. But our paper sizes are limited. We may need many pages for one flow chart. But how to go from one page to another? This is solved by using small circles, called connectors. In this circle, we put some symbol. All connectors having the same symbol represent the same point, wherever they are, whether they are in the same page or on different pages. Flow chart 4.3 gives an example.
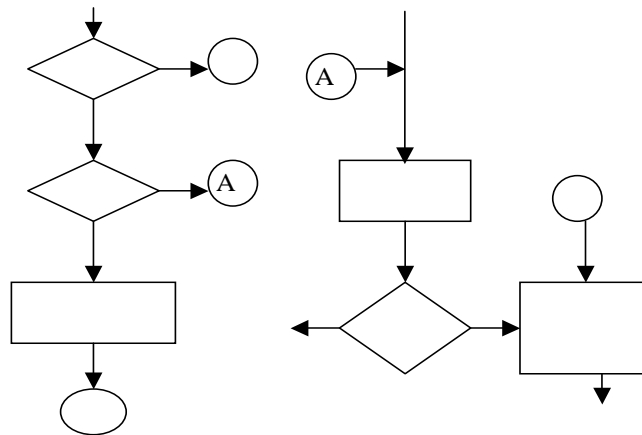
Flow Chart 4.3
185

The advantages of the flow charts are:

- They are precise. They represent our thoughts exactly.
- It is easy to understand small flow charts.

The disadvantage is that real life flow charts can occupy many pages, and hence very difficult to understand. So no one uses flow charts in such situations.

Consider the small flow charts given for the following problems. See whether they will solve the problems. Do not memorize them. Try to understand them. Note how much we have to think before writing a program.

Flow chart 4.4 estimates the volume of a box using its length, breadth and height.
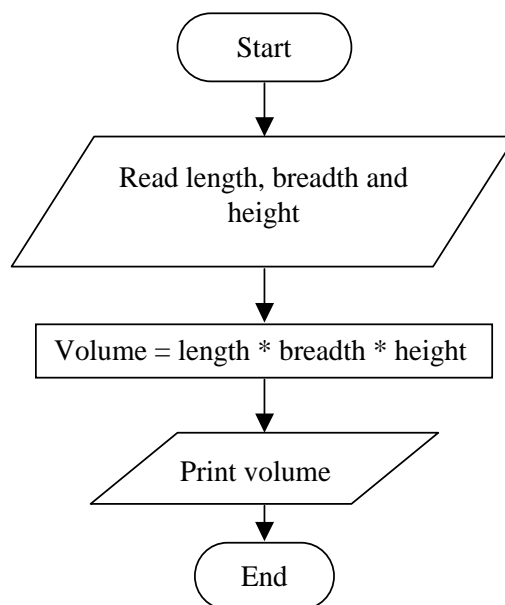
Flow Chart 4.4
186

Flow chart 4.5 reads a number between 0 and 3 and writes it in words.

Flow chart 4.5

187
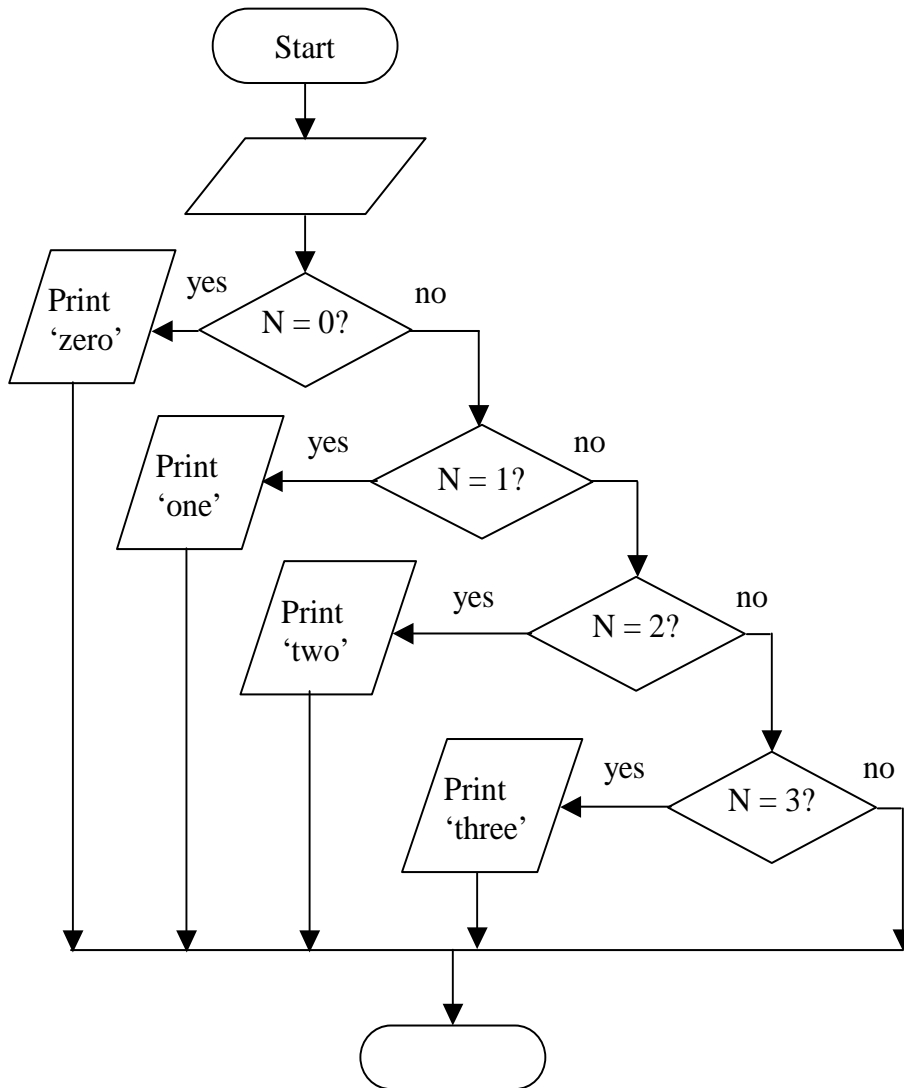
Flow chart 4.6 finds the minimum of 3 given numbers.
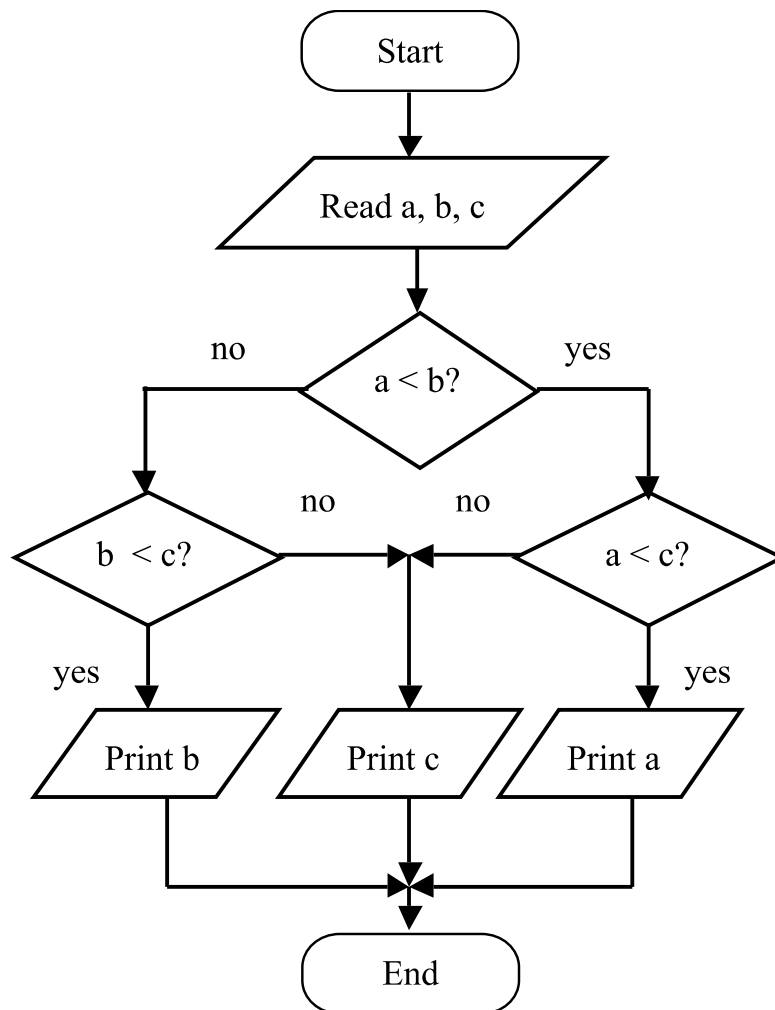
Flow chart 4.6

Flow chart 4.7 provides a method to solve the quadratic equation
$ax^2 + bx + c = 0$.

Start

Read a, b, c

a = 0?

yes → Print 'not a quadratic equation' → End

no → d =b²-4ac

d < 0?

yes → Print 'imaginary roots' → End

no → d = 0?

yes → r = -b/a → Print 'equal roots', r, r → End

no → e=√d
r1=(-b+e)/2a
r2=(-b-e)/2a → Print 'different roots', r1, r2 → End

Flow chart 4.7

189
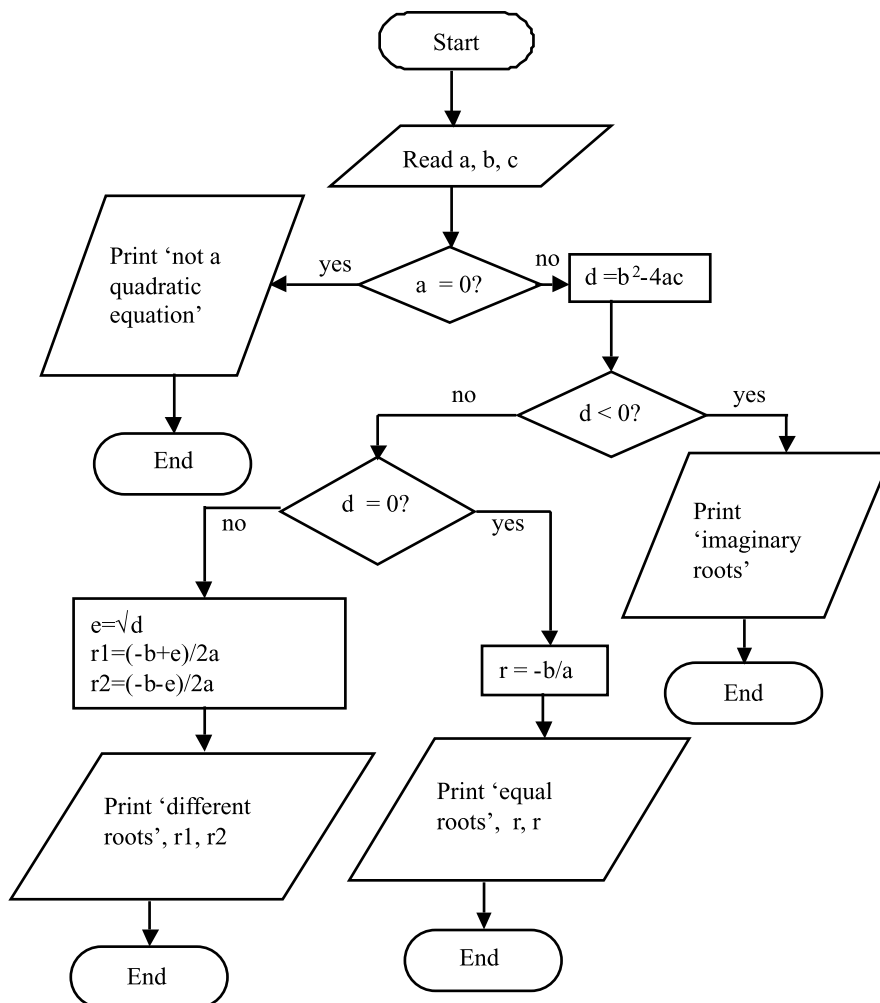
Flow chart 4.8 reads 100 numbers and prints their sum.
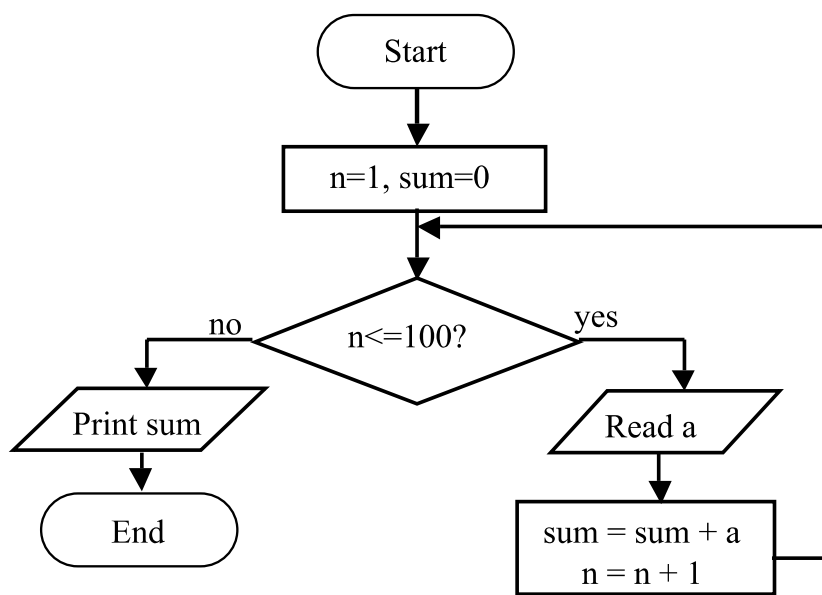
Flow chart 4.8

Flow chart 4.9 determines whether a given integer is a prime number or not a prime a number.
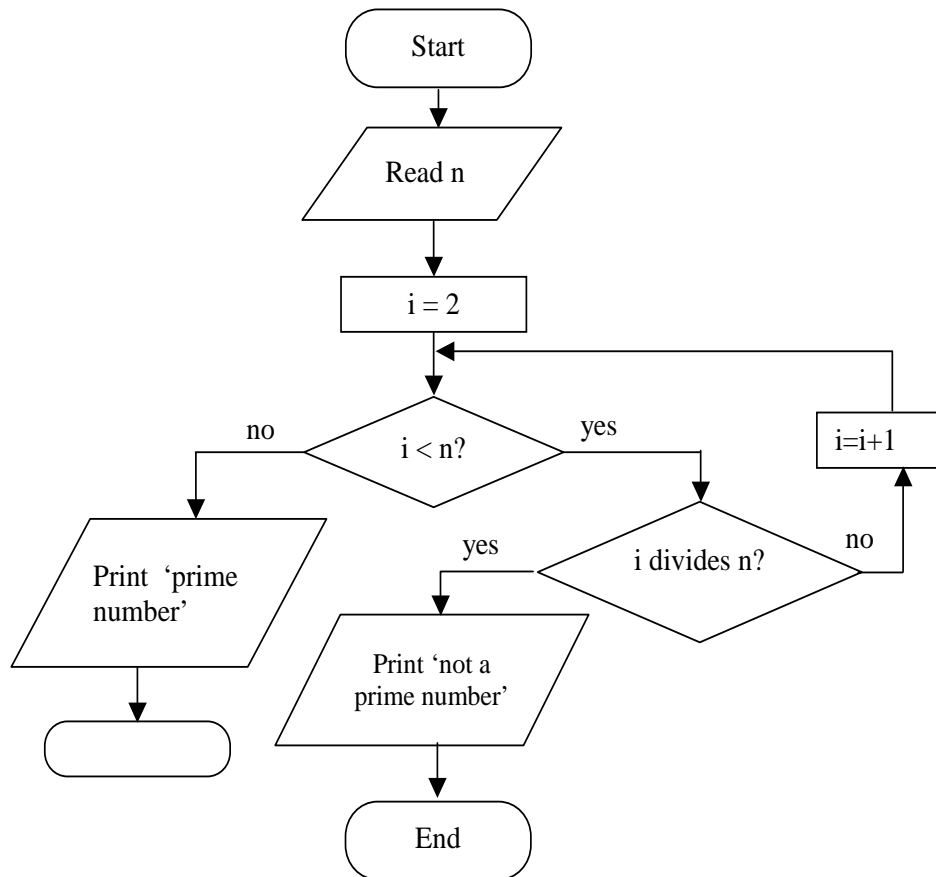
Flow chart 4.9

Flow chart 4.10 finds the smallest integer n such that, 1+ 2+ 3+...+n is equal to or just greater than 1000.
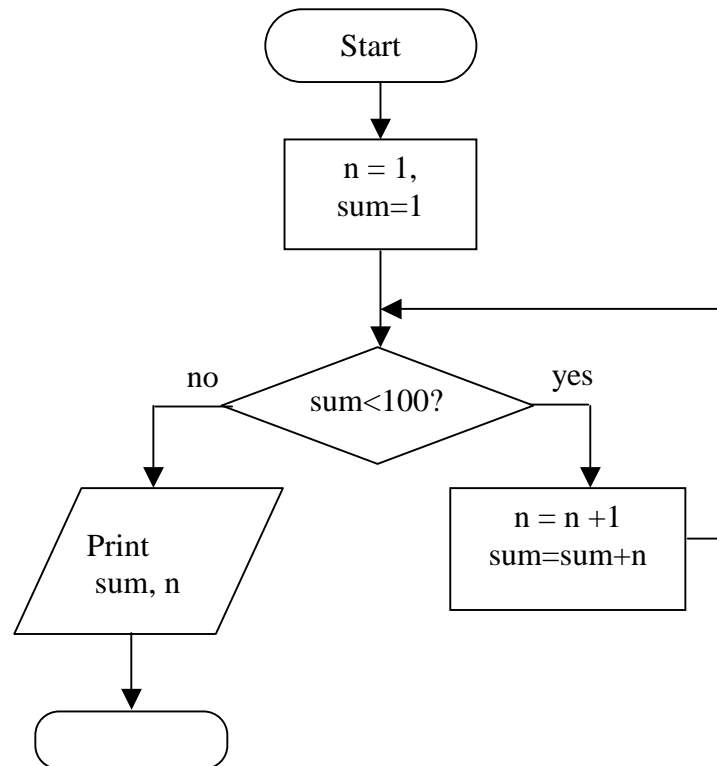
Flow chart 4.10

4.1.1 Fundamental Conditional and Control Structures

In a computer pseudo code, we have to instruct the computer regarding each and every step. Only if we can decide all these things by ourselves, we can write a computer pseudo code. The computer will simply do what we say. And exactly as we say. It won't do anything on its own. We can even say that the computer is the fifth disciple of Paramaartha guru. Do you wonder why?

We shall see one episode in a sequence of such episodes. Paramaartha guru's disciples are named (in Tamil) as Matti, Madayan, Moodan and Muttaal, all different forms of the word 'fool'. They buy an old horse for their guru. The guru sits on the horse and the disciples come by walking. The guru's turban hits a branch of a tree and falls down. After some time the guru asks for the turban. The disciples say that it fallen down. When he asks why they had not brought it, they say that he had not told them so. The guru says that they should take and bring anything that falls down.

One disciple goes back and brings the turban. The guru is annoyed to see the horse dung in his turban. When the guru asks why he had put the dung in the turban, the disciple answers "You only asked us to bring whatever has fallen down. How do we know which one to take and which one to leave? Better give us a list of things to take, so that there won't be any confusion." The guru dictates a long list and then they proceed.

After some time, the old horse trips and falls down. Then one disciple starts reading from the list. Things like turban, dhoti, towel etc. are taken one by one. The guru lies there only with the loincloth. He asks them to take him and put on the horse. For this comes the prompt reply " You are not in the list, guru." When his pleadings fall on deaf ears, the guru asks tem to revise the list, and include his name in the list. Then he asks them to check the list again. Now he was helped to get up.

193

A computer pseudo code is like that list. The computer is like one of these disciples. It is not much different. It is our responsibility to instruct the computer properly and elaborately. We should get trained in thinking in such elaborate manner.

In all these, they are only three techniques, which occur again and again. All the computing is done using only these techniques. Understanding them is just as necessary as the fisherman learning to swim.

Sequencing

Usually the calculations are done one after another, in a sequence. This is one of the fundamental control structures.

4.1.1.1.   Branching

Two-way branching

Ask a question. Get the answer as 'Yes' or 'No'. Depending on the answer, branch to one of the two available paths. This is depicted by a diagonal shaped box. You can see this box in many flow charts. Also many times in the same flow chart. Branching is also a fundamental control structure.

For example, if A > B, then print A, otherwise print B.

This is called the "If ...Then ...Else" structure. If no action is to be taken in one path, then we can use the "If...Then" structure. In this, if the answer is 'No', then it means that the execution goes to the next statement without doing anything.

For example, If you find anyone there then say hello.

Multi-way branching

For some questions there may not be just a Yes or No answer. For example -" What is the age of this boy?" the answer can be one of many integers. Depending on the answer, we may have to make different set of computations, by going through different paths. This is called multi-way branching. This can be depicted as in the flow chart 4.1.1
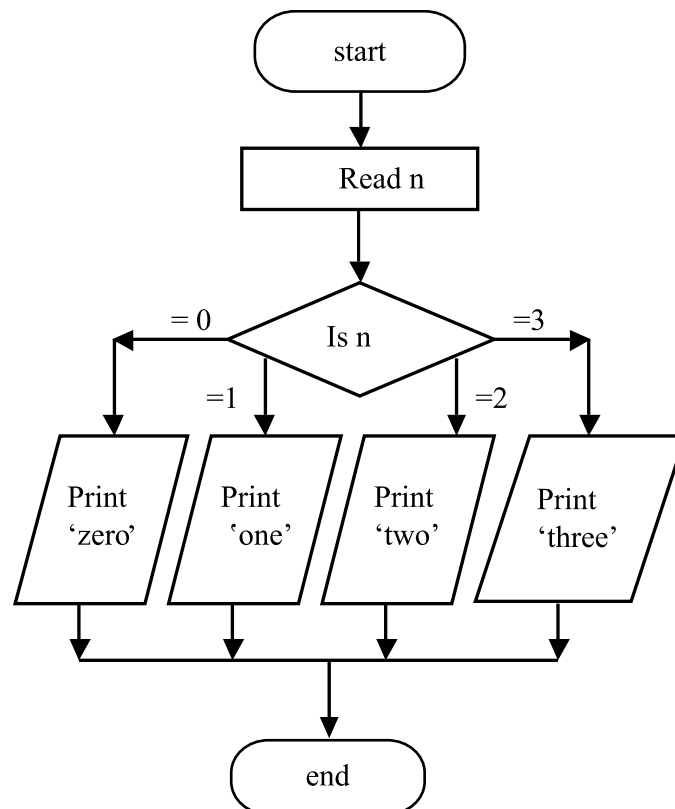
Flow Chart 4.11

For example, if n is 0 then print 'zero'

                1 then print 'one'

                2 then print 'two'

                3 then print 'three'

4.1.2 Iteration

Definite Iteration

The third fundamental technique is iteration. That is, repeating a set of actions again and again. Of course, we do not repeat the same actions for the same data, as this will be just a waste of time. The action will be the same, but the data will change every time. For example, reading 100 numbers, or, finding the interest payable for 1000 customers. In both these examples, we know how many times we are going to repeat the same action. Hence the name definite iteration. In this method we have to keep track of the count of the number of times the actions are performed. For this we use a variable called the index variable or control variable.

There are 4 basic steps involved in using an index variable.

- The index variable should be given an integer as the initial value to start with.
- The current value in the index variable v should be compared with the final value to decide whether more iteration is required.
- If the answer is Yes, then

  - Do the required actions once.
  - Then increment the index v by 1.
  - Go to step 2 and do the checking again.

- If the answer is No, then

  - The iterations are over.
  - Go to the next action in the sequence.

196

The flow chart 4.12 explains about definite iteration. In this case, the iteration is shown by the presence of a loop formed by the directed lines.
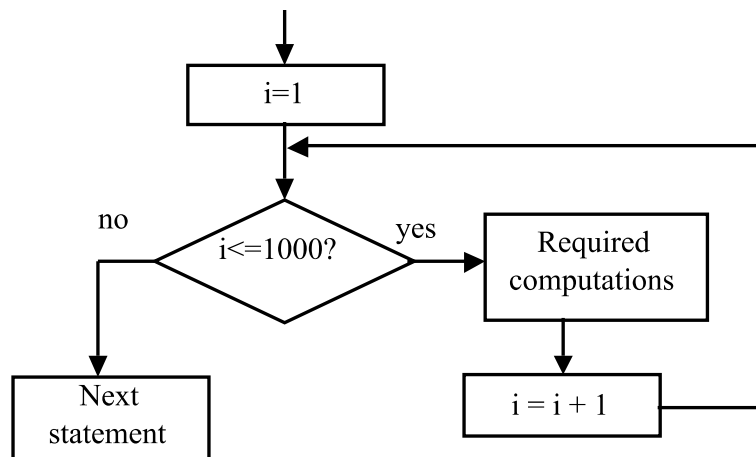
Flow chart 4.12

In some situations, we may not know exactly how many times the iteration is to be performed, as a number, in the beginning. For example, suppose we have to find the smallest number n such that 1 + 2 + 3 + ... + n gives at least 100. Suppose we add numbers one by one and test whether 100 has been reached. We will stop when 100 has been reached. Here we do not know when we are going to stop as a number. A count is not going to work here. Only a condition is to be checked for this. Such iteration is called an indefinite iteration.

Using sequencing, branching and iteration, all the computation can be carried out.

4.1.3 Pseudo Code

Instead of using flow chart, pseudo code can be used to represent a procedure for doing something. Pseudo code is in-between

197

English and the high-level computer languages. In English the sentences may be long and may not be precise. In the computer languages the syntax has to be followed meticulously. If these two irritants are removed then we have the pseudo code.

There are only a few basic sentence types in this. They are also small in their size. But they have the power to specify any procedure. We have one more felicity. We can use the usual brackets in the usual sense of combining many things together. By indentation also we can club statements together. It is easy to understand things written in pseudo code.

The flow chart fundamental control structures for branching and iteration correspond to the following pseudo code.

- If .... then .... else ....
- If .... then ....
- For ..... to .... do ......
- While .... do .....

A few examples for these are as follows.

- **If** a > b **then** print a **else** print b
- **If** a < 10 **then** b = c + d
- **For** i = 1 **to** 20 **do**
       n = n + i
- **While** sum < 100 **do**
      sum = sum + i
       i = i + 1

Note that in the **while .. do** example, the two lines with indentation are to be clubbed together and treated as one unit for execution. The corresponding flow charts for the above examples are given below:

198
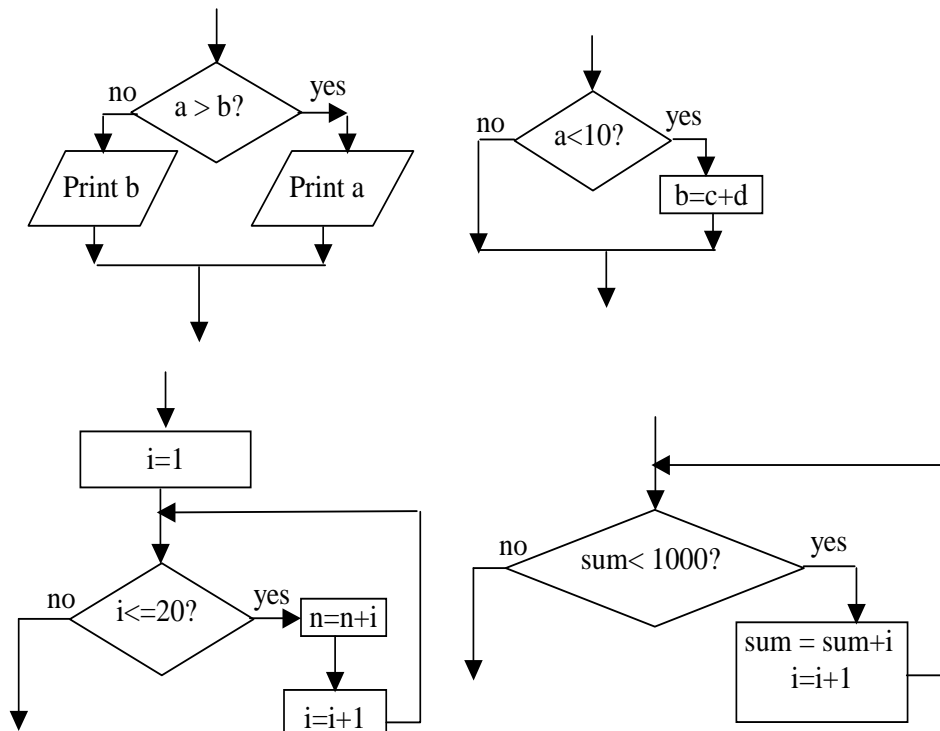
Flow Chart 4.13

Note that all these control structures have only one entry point and only one exit point. That is after executing the things mentioned in these statements, the control is transferred to the next statement. That is, after this statement computer takes up the next statement for execution. This is one of the strong points of these control structures. This makes the understanding and debugging (finding and removing errors) easier.

The pseudo codes corresponding to the problem we have worked out in an earlier section are given below. Compare the flow charts and the pseudo codes, and ascertain their equivalence.

- Finding the volume.
  ```
  start
  read length, breadth and height.
  volume = length x breadth x height
  print volume
  end
  ```

Only sequence is used in the above example.

- Write in words.
  ```
  start
  read n
  if n is
          0 then write 'zero'
          1 then write 'one'
          2 then write 'two'
          3 then write 'three'
  end
  ```

- Finding the minimum of 3 numbers.
  ```
  start
  read a, b, c
  if a < b then
          (if a < c then
                  print a
          else
                  print c
          )
  else
           (if b < c then
                  print b
           else
                  print c )
  end
  ```

- To solve quadratic equation.

```
start
read a, b, c
if a = 0 then
        (
         write 'this is not a quadratic equation'
         exit
        )
else
      find d = b² - 4ac
      if d < 0 then
              write 'imaginary roots'
      else
              if d = 0 then
                      r = -b/a
                      write ' equal roots'
                      write r, r
              else
                      r1 = (-b + d)/ 2a
                      r2 = (-b - d)/2a
                      write 'unequal roots'
                      write r1,  r2
end
```

- Sum of 100 numbers

```
start
sum = 0
n = 1
while n <= 100 then do
            read a
            sum = sum + a
            n =  n + 1
print sum
end
```

- Prime number

```
         start
         read n
   for i = 1 to n-1 do
    if i divides n then
                   (write 'not a prime'
                    exit program
                   )
write 'prime number'
end
```

- Sum giving 1000 or more

```
       start
 i = 1
 sum = 1
 while sum < 1000 do
          i = i + 1
           sum = sum  + i
 print i and sum
 end
```

There are a few points to be noted.

- Within one 'if then else' statement, there is another 'if then else' statement. To show this clearly indentation is used.

- Only the inner statement is written with extra indentation. All the statements in a sequence have the same indentation.

- Just as we use brackets in Mathematics, here also we use brackets for bunching.

- Since the procedures written in the pseudo code look similar to a program, it is very easy to convert it into a high-level language computer program.

4.1.4 Walkthrough

As a first step in writing a program, a flow chart or pseudo code is created, to represent the solution method. This comes under the designing a solution for the problem. From this, the program is written with the specific syntax rules of a particular language. Before writing the program, one must be sure of the correctness of the method used. Hence it is necessary to check the correctness of the flow charts and pseudo codes. First we shall see what an algorithm (solution method) is. Then we shall see a method, called walkthrough, of checking the pseudo code or flow chart.

An algorithm is a procedure with the following properties.

- There should be a finite number of steps.
- Each step is executable without any ambiguity.
- Each step is executable within a finite amount of time, using a finite amount of memory space.
- The entire program should be executed within a finite amount of time.

Suppose a pseudo code or flow chart for an algorithm is given. A method of checking the way in which a computer will work using this is called a walkthrough. Let us consider an example of a flow chart. Assume that you are the computer. You have to start at the box named start. From this point you have to move along the path indicated by the arrows. If you enter a rectangular box do the necessary calculations.

If it is an input box, assume some specific values for the particular variables. Each time assume a different set of values so that you will take different paths. For small problems, it is not difficult to exhaust all possible paths from start to end. This is one step in assuring that the method is correct.

If it is an output box, write down the current values of the variables mentioned.

4.1.5 Creating a Program

Writing a small program is easy. A flow chart or pseudo code can be drawn for this first. Then from this the program can be written easily. But this method does not work in the case of real life programs, which are big. A systematic approach is very much essential in this case.

In a program we have to provide many details. But, when we start a big venture, if we start thinking about finer details, then we will not be able to concentrate on the real job. We may never be able to finish the job.

This is the right way to do any big job. We know the strength of unity. We have read many stories to illustrate this. For example, we cannot break a bundle of small sticks. But when they are unbundled, each stick can be broken very easily. This divide and conquer approach is used in many places. This method is useful in creating a big program also.

To create a program, the problem should be divided into many smaller problems. We should know the method of putting together the results of these sub problems to get the result for the bigger problem. This is one step in creating a program. This step is repeatedly used, until the problem becomes small enough to write a flow chart or pseudo code.

The interesting thing is that the logic for combining the sub problems also can be written using the same pseudo code we are familiar with. A flow chart can also be used for this. So using many pseudo codes or flowcharts, the entire program methodology can be written. Converting these into a computer program is a simpler job now.

This method of approach is called the 'Top Down Approach'. In each step we concentrate on a single thing, find how to get the solution by combining the results of smaller problems. In the case of computer programs, when this method was used first, more importance was given to the procedures, that is, the method of executing things, and not for the data. It is called 'Structured Programming'. When dividing a problem into smaller parts, if both the procedure and the data are taken into account, then we have what is called the 'Object Oriented Approach'. It is found that this approach is best suited for developing programs. Computer languages like C++ and Java help in writing programs developed using the object oriented approach. We will not be going into details of this approach here.

We shall illustrate the use of top down approach, using an example. You should use this method to develop any program.

Problem: Write a program to arrange 100 numbers in the ascending order.
The method is:
- Read 100 numbers and put them as a1, a2, ...
- Find the smallest of these.
- Interchange this number with the first number.
- Repeat these steps, 98 times (can you say why?), starting from the second position, third position etc.
- Output the 100 numbers.

Let us write this in pseudo code in a more systematic way.

Read 100 numbers and put them in an array, as a(1), a(2) etc.

Do for i = 1 to 99

     Find the smallest of a(i) to a(100)
     Let the smallest number be at the jth position
     Interchange a(i) and a(j)

Output a(1), a(2) ... a(100)

Here we have told that we can get the answer, if we can do a few things like,

- Inputting and outputting the 100 numbers.
- Finding the smallest of a given set of numbers
- Interchanging two numbers

We have not bothered to talk about the details of achieving these. That is left to the next stage of processing. In the second stage we take these smaller problems one by one and try to solve them.

Reading 100 numbers:
For count = 1 to 100 do
    Read a(count)

This method can be used for outputting also.

To find the smallest of the numbers a(1) to a(100):

    Let i = 1
    Let position = 1
    Let min = a(i)

    For n = i+1 to 100 do
        If a(n) < min then
            min = a(n)
            position = n

    Let j=position
    Interchanging a(i) and a(j):

    Let temp = a(i)
    a(i) = a(j)
    a(j) = temp

    Here, it will be wrong if we write

    a(i) = a(j)
    a(j) = a(i)

That is because, after executing the first statement, the old value of a(i) is lost.

4.2 Introduction to C Programming

The C programming language is a popular and widely used programming language for creating computer programs. Dennis Ritchie at AT & T Bell Laboratories developed the C language three decades ago. Though it was designed originally as a language to be used with UNIX operating system, the C language is a general-purpose language. It is an efficient, flexible and portable language. Portability refers to the case in which a unit of software written on one computer may be moved to another computer without any or little modification. C has a wide diversity of operators and commands. C can be effectively utilized for development of system software like, operating systems, compilers, text processors, and database management systems. C is well suited for developing applications programs too.

The C language is composed of the following basic types of elements:

constants, identifiers, operators, punctuation, and keywords

These elements are collectively known as **tokens**. A token is a source program text that the compiler does not break down into component elements.

Consider the statement: **number = number + 1;**

The tokens are,

    number   - identifier (variable)
       =   - operator
       +   - operator
       1   - constant
       ;   - punctuation

The above statement has a collection of tokens such as identifier, constant, operator etc. The words like **if, while, for** etc., are termed as keywords in C programming language and their usage is restricted to specific purposes. The keywords are discussed in a later section of this chapter.

4.2.1 Constants

A **constant** is of numeric or non-numeric type. It can be a number, a character or a character string that can be used as a value in a program. As the name implies, the value of a constant cannot be modified. A constant is immutable. Numeric data is primarily made up of numbers and can include decimal points. Non-numeric data may be composed of numbers, letters, blanks and any special characters supported by the system. In other words, non-numeric data consists of alphanumeric characters. A non-numeric data can be called as a literal. Constants are characterized by having a value and type.
Numeric constants are of three types:

- integer constant
- floating-point constant
- character constant

The fundamental data types of C language are discussed later in this chapter.

4.2.1.1 Integer Constant

An integer constant is a decimal number (base 10) that represents an integral value (the whole number). It comprises of the digits 0 to 9. If an integer constant begins with the letters 0x or 0X, it is a hexadecimal (base 16) constant. If it begins with 0 then it is an octal (base 8) constant. Otherwise it is assumed to be decimal.

> 23, 36 and 948 are decimal constants
> 0x1C, 0XAB, and 0x23 are hexadecimal constants
> 071, 023, and 035 are octal constants

Integer constants are positive unless they are preceded by a minus sign and hence −18 is also a valid integer constant. Special characters are not allowed in an integer constant. The constant 2,345 is an invalid integer constant because it contains the special character.

4.2.1.2 Floating - point Constant

A floating-point constant is a signed real number. It includes integer portion, a decimal point, fractional portion and an exponent. While representing a floating point constant, either the digits before the decimal point (the integer portion) or the digits after the decimal point (the fractional portion) can be omitted but not both. The decimal point can be omitted if an exponent is included. An exponent is represented in powers of 10 in decimal system.

Example:

58.64 is a valid floating-point (real) constant. It can be represented in exponent form as follows:

$5.864E1 => 5.864 \times 10^1 => 58.64$
$5864E-2 => 5864 \times 10^{-2} => 58.64$
$0.5864e2 => 0.5864 \times 10^2 => 58.64$

The letter E or e is used to represent the floating-point constant in exponent form.

4.2.1.3 Character Constant

A character is a letter, numeral or special symbol, which can be handled by the computer system. These available symbols define the system's character set. Enclosing a single character from the system's character set within single quotation marks forms a character constant. The characters used in C language are grouped into three classes.

- Alphabetic characters   a, b, c, …., z, A, B, C, ….., Z
- Numeric characters   0 through 9
- Special characters   + - * / % # = , . ' " ( ) [ ] :

Example:

'1', 'a', '+', and '-' are the valid character constants.

Since two single quotes are used to represent the character constant, how do we represent the single quote itself as a character constant? It is not possible to represent a single quote character enclosed between two single quotes, that is, the representation ''' is invalid.

An escape sequence may be used to represent a single quote as a character constant. Character combinations consisting of a backslash **** followed by a letter are called escape sequences.

'\'' is a valid single quote character constant.

Similarly some nonprintable characters are represented using escape sequence characters.

Examples:

'\a'	Bell (beep)
'\b'	Backspace
'\f'	Form feed
'\r'	Carriage return
'\n'	New line
'\0'	null character

To represent the backslash itself as a character, two backslashes are used (**'\\'**).

4.2.1.4 String Literal

A string literal or a string constant is a sequence of characters from the system's character set, enclosed in double quotes. By default, the null character '\0' is assumed as the last character in a string literal. To have a double quote itself as a character in the string constant, an escape sequence '**\'**' is used.

210

"hello" is a valid string literal. The actual number of characters in this string literal is 6 including the null character at the last. The null character is invisible here. Six bytes are required to store this string in memory. However, the physical length of this string is 5 characters.

4.2.2 Identifiers

Identifiers are the names that are to be given to the variables, functions, data types and labels in a program. The name of a variable can consist of alphabets (letters) and numbers. Other characters are not allowed in the name of a variable. An underscore character can be used as a valid character in the variable name. The variable name starts with an alphabet and its length may vary from one character to 32 characters. The first character in a variable's name should be an alphabet, ie., a number is not allowed as a first character in the variable name. The valid variable names are:

> x
> length
> x_value
> y_value
> a123

Keywords (which have special meaning in C) cannot be used as identifiers. The following variable names are invalid and the reasons are stated.

123	- The first character is a number
1abc	- The first character is a number
x value	- A blank character is used
x&y	- A character other than alphabet and number is used & is not a valid character in a variable name.
for	- It is a keyword

4.2.3 Fundamental Data Types

Data is one of the most commonly used terms in programming. Data can be defined as the raw information input to the computer. Data is differentiated into various types in C. There are three numeric data types that comprise the fundamental or primary data types. The fundamental data types are: **int, float and char**. They are also called as pre-defined **primitive types** associated with the C compiler. The C compiler knows about these types and the operations that can be performed on the values of these types. The following are the declaration statements of variables of different types:

> **int x;**
> **float f;**
> **char ch;**

We know already that an integer is a whole number, a float is a real number and a character is represented within single quotes.

Example:

> **1   is an integer (whole number)**
> **1.0   is a real number (floating point number)**
> **'1'   is a character constant**
> **"1"   is a string literal**

The data type associated with a string literal is **char *** (character pointer) and the explanation is provided in the Arrays section of this chapter.

An integer requires 2 bytes of memory to store its value, a float requires 4 bytes of memory and a character requires 1 byte of memory.

4.2.3.1   Derived Types

Long, double, unsigned, arrays and pointers are the derived types from the fundamental primitive types. A long integer can be declared as follows:

long int i;

or

long i;

To store a long integer value, four bytes of memory are required. To store the real (float) values more precisely, the type **double** is used. It occupies 8 bytes in the memory.

Unsigned numbers are represented as follows:

unsigned int i;

Unsigned int occupies 2 bytes as normal integers but all the bits are used to store the value. In a regular integer, the most significant bit (the left most bit) is a sign bit.

4.2.3.2 Pointer Variables

The variables in C are classified into **ordinary variables and pointer variables.** An ordinary variable can take values of its associated type.

Example:

int x;

Here, **x** is an ordinary variable of type integer and assumes a whole number as its value.

x = 10;

A pointer variable is declared as follows:

int *y;

The above declaration is a pointer variable declaration. Here, **y** is a pointer variable whose type is an **integer pointer (int *)**.

A pointer variable assumes only address as its value. Each variable takes some locations in the main memory according to its type. Every location in the main memory is addressable.

In the above examples, **x** is an ordinary variable and **y** is a pointer variable. **x** is an ordinary integer and **y** is a pointer to an integer. Hence, we can store the address of **x** in **y**.

y=&x;

There are only two operators associated with pointers - an **address of (&)** operator and an **indirection (*)** operator. Both operators are unary operators. The unary operators have been discussed in the next section.

Since **x** is an integer, it occupies two bytes in the memory. Pictorially it can be represented with its address as follows:

948   949

```
┌──────┬──────┐
│      │      │
└──────┴──────┘
```
X
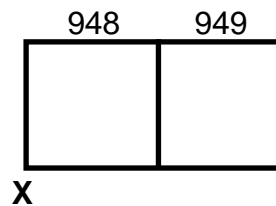
The address of the variable **x** is 948 as shown above. Every byte is addressable in main memory. To obtain the address of the variable, we have to use the "address of" operator (**&**). So in the statement **y = &x;** the address of **x** is stored into the pointer variable **y**. Since **y** is a pointer variable, it can assume only an address of a variable. We can say that **y** points to **x**. We can represent this as follows:

948   949

```
┌──────┬──────┐
│      │      │
└──────┴──────┘
```
Y   X

214
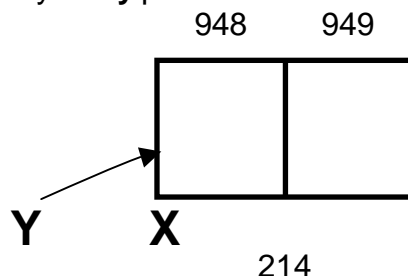
Assume, the value of **x** is 10. To retrieve the value of **x** through the pointer variable **y** (since **y** already points to **x**), we can use the "indirection" operator (*). That is, ***y** will give the value contained in the location pointed by **y**. If the value of x is 10 and the pointer variable y is pointing to x then the value of the expression ***y** is 10.

Remember in the above example,

- **y** represents the address of the variable **x**   (**&x**)
- ***y** represents the value of the variable **x**   (**x**)

Both **address of** and **indirection** operators are used as unary operators here.

4.2.4 Operators

C has rich set of operators. Operators are what make things happen. An operator is defined as a symbol that specifies an operation to be performed. Operators inform the computer what tasks it has to perform as well as the order in which to perform them. The order in which operations are performed is called the **order of precedence**. It is also called **hierarchy**. The direction in which operations are carried out is called **associativity**. There are three types of operators in C.

- Unary operators
- Binary operators
- Ternary operator

4.2.4.1 Unary Operators

Order of precedence is high for the unary operators and they have only one operand. The order of evaluation (associativity) is from right to left. Table 4.1 lists the unary operators and their functions.

Table 4.1 Unary Operators

Symbol	Type of operation	Associativity
++ — * & !	Increment Decrement Indirection Address of Negation (logical NOT)	Right to Left

The increment or decrement operator is used to increase or to decrease the current value of a variable by 1. Generally the unary operators appear before the operand. For example, to find the address of integer variable x, we can use the expression **& x**. In case of increment and decrement operators, they may appear before or after the operand.

Hence there are two forms:
Postfix increment or decrement
Prefix increment or decrement

The unary operators ++ and — are called **prefix** increment or decrement operators when they appear before the operand. The unary operators ++ and — are called **postfix** increment or decrement operators when they appear after the operand. Their usages are explained in the other sections of this chapter.

4.2.4.2   Binary Operators

Arithmetic Operators

The arithmetic operators like **+, -, *, /,** and **%** are binary operators as they have two operands. Table 4.2 shows the list of arithmetic operators. All the arithmetic operators observe left to right associativity**.**

Table 4.2   Arithmetic Operators

Symbol	Type of operation	Associativity
+ - * / %	Addition Subtraction Multiplication Division Modulus	Left to right

The arithmetic operators can be used with integers and float values. The modulus operator works with integers only and it gives the remainder value after division. The integer division truncates the result.

Example:

5 / 2 = 2   (the fractional part is truncated)

5 % 2 = 1   (the remainder after division)

With respect to integers, the division operator returns the quotient of the division and the modulus operator returns the remainder after division. The division, multiplication and modulus operators have higher precedence than addition and subtraction operators.

Note: The symbol * is a binary operator used for multiplication. The same symbol * is used as a unary operator (indirection operator) along with pointer variable as we had seen earlier.   All depends upon the context in which the symbol * is used.

4.2.4.3   Relational Operators

The relational or Boolean operators are binary operators as they always have two operands but as an exception, the Boolean operator **!** (Which is called as a negation operator) is a unary operator, that is, it has only one operand. Table 4.3 shows the list of relational operators. All the relational operators observe left to right associativity, that is, they are evaluated from left to right.

Table 4.3 Relational Operators

Symbol	Type of operation	Associativity
== \< \> \<= \>= !=	Equal to (equality) Less than Greater than Less than or equal to Greater than or equal to Not equal to (inequality)	Left to Right

The relational operators are used to compare two values (items) and the result will be either true or false. The logical operators "**&&**" and "**||**" are used to connect two or more relational expressions. The "**&&**" operator is the logical AND operator and it returns a value of true if both of its operands evaluate to true. A value of false is returned if one or both of its operands evaluate to false. The logical OR operator "**||**" returns a value of true when one or both of its operands evaluates to true and a value of false when both of its operands evaluate to false. The relational operators have lower precedence than the arithmetic operators. The logical **&&** and **||** operators have still lower precedence than the relational operators.

218

Example:

The expression (10 < 15) && (14 < 23) is always true. The expression is evaluated from left to right. Two relational expressions are combined using logical && operator. These two expressions are evaluated before the logical **&&** operator evaluates the entire expression. Consider another example:

The expression (10 < 15) || (14 < 23) is also true. The logical operator || combines two relational expressions. As per the rule of logical OR, the entire expression is true if either one of the relational expressions is true. Since the first relational expression is true in this example, the second relational expression will not be evaluated. This concept is called Short Circuit Evaluation. In case of logical AND (&&) operator, if the first operand evaluates to false then the entire expression becomes false and hence, the second relational expression will not be evaluated.

4.2.4.4 Assignment Operators

The assignment operator (**=**) assigns the value of the right-hand operand to the left-hand operand. C has arithmetic-assignment operators too. They are **+=, - =, *=, /= and %=.**

Consider the following statement:

i = i + 1;

Here, the old value of the variable i is incremented by 1 and the incremented value is stored as the new value of i. The above statement can be represented in the following ways.

Using the arithematic - assignment operator**,**
i += 1;

In the above statement, the operator **+=** is used to increment the old value of **i** by one and the new value is stored in the variable **i** itself. Similarly, the statement **i *= 2;** represents that the old value of **i** is multiplied by 2 and the new value is stored in **i** itself.

Of all the operators, the assignment operators have the least precedence

4.2.4.5 The order of evaluation

We have discussed the precedence of operators in the previous section. Using the precedence of operators, we will see how to evaluate an expression. The bracketed expression should be evaluated first.

Consider the following expression

5 * 2 + 8 + (3 – 2) * 5

It will be evaluated as follows:

5 * 2 + 8 + 1 * 5   (bracketed expression is evaluated first)

10 + 8 + 5   (multiplication has higher precedence over addition)

23   (the value of the expression is 23)

4.2.4.6 Ternary Operator

C has one ternary operator, which is a conditional operator. The symbol used for this operator is **?:** and it has three operands. The syntax of the conditional operator is as follows:

conditional expression? expression 1 : expression 2;

If the conditional expression is true, expression 1 is evaluated. If the conditional expression is false, expression 2 is evaluated.

Example :

The following example show the uses of conditional operator:

j = i < 0 ? - i : i;

This example assigns the absolute value of i to j. If i is less than 0, -i is assigned to j. If i is greater than or equal to 0, i is assigned to j.

4.2.5 Punctuation and Keywords

C's punctuation symbols and their uses are listed as follows:

[ ] - used to represent array index (square brackets)

{ } - used to cover the body of the function (curly braces)

( ) - used to represent a function, to group items and to group expressions   (simple parentheses)

< > - used to enclose a header file in a preprocessor statement

(angular brackets)

" " - used to represent string literals (double quotes)

' ' - used to represent a character constant (single quotes)

/* */ - used to represent a comment

;   - used as a statement terminator

,   - used to separate items

blank and white space – used to improve readability of the program

The following are the list of important keywords. They cannot be used as identifiers for the variables in a program.

221

**auto   break   case   char   continue   default   do
else   if   float   for   int   return   static
switch   while**

A keyword must be specified precisely as given in the list.

For example, **auto** is a keyword, whereas, **Auto** or **AUTO** are not keywords.

4.3 A Sample C Program

A program is defined as a set of instructions to be executed sequentially to obtain the desired result. A function is a program, which is being used to carry out some small task.   C programs can be very small.   C programs are made up of functions. A program cannot be written without a function. A function may be pre-defined or user-defined. We will see the first program in C. The C program code is given below:

```
#include <stdio.h>
main()
{
   printf("Hello World");
}
```

A Sample C program

This program, upon execution, will display the message **Hello World** on the screen. There must be a function defined as ***main()***. The main() function is a user-defined one. The user has to define the main() function to provide necessary code. When a C program runs, the control is transferred to this function. This is called the program's entry point. The program has two functions, one is the user-defined main() function which is the entry point and the other one is printf() function which is pre-defined and used to display the results on the stan-

222

dard output (screen or monitor). Simple parentheses () are used to represent a function. Here main() is the **calling function** and printf() is the **called function**. The "Hello World" program with necessary explanations is given below.

```
#include <stdio.h>    /* <=  preprocessor statement */
main()         /* <=  function header statement */
/* function definition starts */   <=  comment statement
 {

        printf("Hello World");

 }
```

(with annotations: **function call statement**, **function body**)
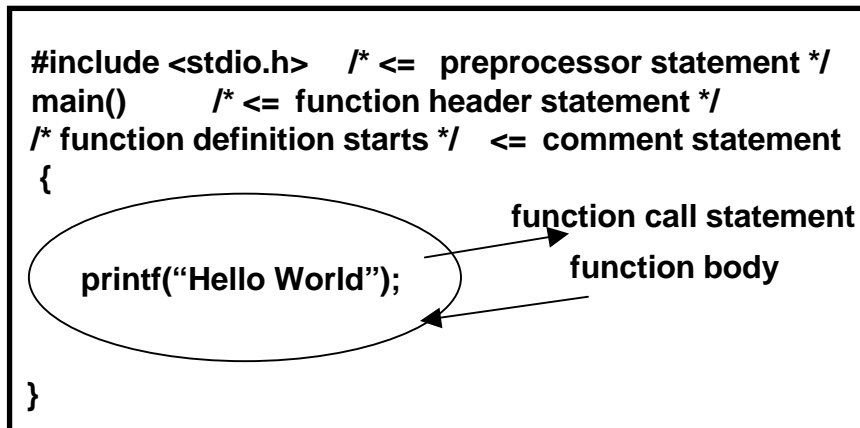
The first line in the program **#include <stdio.h>** is a preprocessor statement. #include is a preprocessor directive. The preprocessor is a software program that will expand the source code while the program is compiled. The #include <stdio.h> statement includes the contents of the stdio.h file (standard input and output header file) globally, that is, prior to the main() function. The contents of the stdio.h file are the function declaration statements of the pre-defined output and input functions like printf() and scanf() etc.

The declarations of the functions printf() and scanf() are as follows:

> **int printf(char *, …);**
> **int scanf(char *, …);**

The ellipses (**…**) represent that the above two functions can take variable number of parameters. But the first parameter is always a string. A parameter is a data or information passed on to the called function. Zero or more number of such parameters are passed to any function. They are given one after another within the brackets, which come after the name of the function. In the program shown above, the

223

printf() assumes only one parameter which is a string to be displayed on the monitor. The first parameter type (string type) is char * which will be read as "character pointer" and will be discussed later.

The C compiler is able to recognize the pre-defined functions only because of their declarations in the appropriate header files. The following program illustrates the use of header files. The program while it is being executed clears the contents of the screen before displaying **hello** on the monitor.

```
#include <stdio.h>
#include <conio.h>
main()
{
clrscr();
printf("hello");

}
```

Illustration of header files

The function clrscr() is a pre-defined one whose prototype (declaration) is available in conio.h file and hence it has been included. If the statement **#include <conio.h>** is not included, the C compiler expects the definition of clrscr() function from the programmer and if the definition is not provided, it reports an error.

4.3.1 Statements

Each and every line of a C program can be considered as a statement. There are generally four types of statements. They are:

- Preprocessor statement
- Function header statement
- Declaration statement
- Executable statement

As you know already, the preprocessor statement is used to expand the source code by including the function declaration statements from the specified header files. The function header statement is used as a first line in the definition of a function. The declaration statements are further classified into variable declaration statements and function declaration statements.

```
#include <stdio.h>      =>  Preprocessor Statement
main()                  =>  Function header Statement
{
    int a,b,c;              => Variable declaration statement
    int add(int,int);       => Function declaration statement
    a = 10;                 => Executable statement
}
```

The declaration and definition of user-defined functions have been discussed in the **Functions** section. There are many forms of executable statements. An assignment statement is a fundamental one, which is used to assign a value to a variable.

4.3.1.1 Assignment Statement

An assignment statement is defined as:

Variable = Expression;

A semicolon terminates the assignment statement. An expression is of many types, which will be discussed later. An expression produces one result as it is being evaluated, i.e., it has always been reduced to a single value. The value of the expression is assigned to the left hand side variable. The '=' sign is the assignment operator which is used to assign a value to a variable. Everything on the right hand side of an assignment operator is considered as an expression.

225

All statements in a function are enclosed between curly braces as we have already seen. The **printf("hello");** statement is a function call statement. When it is being invoked, the message **hello** is displayed on the screen and the function returns an integer value that represents the number of characters displayed successfully on the screen. From the declaration of the printf() function, we can understand that it returns an integer. For example, consider the following program segment:

int n;   /* variable declaration statement*/

n = printf("hello");   /*assignment statement*/

Here, the variable n is being declared as an integer. While the assignment statement is being executed, the right hand side expression is evaluated first and then the value of the expression is assigned to the left hand side variable. In this example, we have a **function call expression** on the right hand side of the assignment operator. The printf() function is invoked to display the message **hello** on the screen. The variable **n** is being assigned with a value 5 that is the value of the right hand side expression as the printf() returns an integer that represents the number of characters successfully displayed on the screen.

4.3.1.2 Increment and Decrement Statements

To increment the old value of a variable by 1 and to store the new value into the same variable can be achieved by using the statement:

i = i + 1;

The same can be achieved by using an increment statement, which is,

i++;   /* postfix form */
or
++i;   /* prefix form */

226

If the increment operator is used in stand-alone statements as shown above, there is no difference in the postfix and prefix representations. In both cases, the value of **i** is incremented by 1. To decrease the old value of **i** by 1, a decrement operator is used instead, ie., **i—; or —i;** These operators may have different effects in other situations, which will be discussed later.

4.3.1.3 Expression

An expression occurs usually on the right hand side of an assignment statement. It has a value when it is evaluated. There are many forms of expressions and some of them are shown below:

      int a,b,c;   **variable declaration statement**

      a = 10;

On the right hand side, a constant value is used and hence it is a **constant expression** whose value is 10.

      b = a;

A **variable expression** is used here whose value is 10. The right hand side of this assignment statement has been associated with two values: a variable's value and an expression value. In this case both values are same. But always remember that the **expression value** is assigned to the left hand side variable.

The expressions can be named based on the operators used. The other expressions are:

                         right hand side

     c = a+b;   **arithmetic expression**

     c = a > b;   **relational expression**

     f = d = e;   **assignment expression**

227

In case of a relational expression, a true or false value is assigned to the left hand side variable and hence 1 or 0 is assigned to the variable c. The '=' symbol is used as an assignment operator and hence assignment expression is allowed in C. Cascading of assignment is permitted. The value of the assignment expression (the value assigned to **d** is the value of the assignment expression) is assigned to the variable **f**.

Consider the statement,

f = d = 10;

The value of the assignment expression is 10 which is the value assigned to d. The value of the assignment expression is then assigned to f.

4.3.1.4 Postfix and Prefix increment Expressions

Consider the following code segment:

int x, i;
i = 10;
x = i++; /*postfix increment expression on the right side*/
printf("%d %d\n", x, i);

The unary ++ operator has the highest precedence and the assignment operator has the least precedence. In case of postfix form, first the value of the variable is used as the value of the expression and then the value of the variable is incremented by 1. Hence on the right hand side, there are two different values, one is the value of the expression and another is the value of the variable. In the above assignment statement, the value of the expression, which is 10, will be assigned to the left-hand side variable **x**. The value of the variable **i** becomes 11. We have to consider always the value of the expression while assigning a value to a variable on the left hand side in an assignment statement. Hence the output will be

10 11

Consider the following code segment:

```
int x, i;
i = 10;
x = ++i;      /*prefix increment expression on the right side*/
printf("%d %d\n", x, i);
```

In case of prefix form, first the value of the variable is incremented by 1 and the incremented value is used as the value of the expression. The value of the expression, which is 11, will be assigned to the left-hand side variable **x**. The output will be

11 11

Consider the following example with relational expression:

```
nt x, z;
x = 100;
z = (x == x++);      /* relational expression with one of the
                     operands as postfix increment expression*/
printf("%d %d\n", z, x);
```

What will be the output?

The relational expression used in the above assignment statement will yield a value of true or false, i.e., 1 or 0. Therefore, the value of the variable z may be 1 or 0. Because of the highest precedence of ++ operator, the right side operand of the relational expression is evaluated first. Since it is a postfix expression, the expression value will be 100 and then the variable x gets incremented and its value becomes 101. Now the left operand (which is a variable expression) of the relational expression has the value 101. Hence, the values to be compared are 101 and 100 and they are not equal, the value 0 is assigned to z. The output will be:

0   101

Remember:

In the assignment statement, we have to give importance to the value of the expression and not to the value of the variable.

4.3.1.5 Input and Output Statements

As we have seen already, the function printf() is used to display the results on the standard output (screen).   We have seen the use of printf() to display the string on the monitor. Actually, the first parameter of the printf() function is a string which is used to control the output and hence it can be called as "**control string**". This parameter is used to format the output for display and hence we can also call it as a "**formatting string**".

Example:

To print a value of an integer:

> **int n;   /* variable n is declared as integer*/**
> **n = 10;**
> **printf("%d", n);**

In the above example, '%d' is used as a formatting character within the control string of printf() function to display the value of an integer.   The control string of printf() function can take three types of characters.

- Ordinary characters
- Formatting characters
- Escape sequence characters

Ordinary characters within the control string are displayed as such. In the case of **printf("hello");** statement, the control string has ordinary

characters only. They are displayed as such on the screen. Table 4.4 lists the formatting characters used to display the values of various types.

Table 4.4 Formatting Specifications

Formatting character	Data type
%d	int
%f	float
%c	char
%s	char [ ]
%ld	long int
%lf	long float or double

As seen already, the escape sequence characters are represented by a backslash followed by another character. They are in fact single character constants only. They are stored and manipulated as a single character. Escape sequences allow partial control over the format of the output. The frequently used escape sequences in the control string of printf() function are:

'\n'   -   new line character
'\t'   -   tab character
'\b'   -   backspace character

Consider the following printf() function call statement:

int i = 15;

printf("the value of i = %d \n", i);

The output displayed on the screen is:

the value of i = 15

All the three types of characters namely, ordinary characters, formatting character and escape sequence character are used in the control string of above printf() function. The ordinary characters are displayed as such. Because of the formatting character %d, the value of the next unprocessed integer has been displayed. The output is displayed in one line and because of the new line character '\n', the cursor comes to the next line and the subsequent outputs are displayed there.

The statement **printf("one\ntwo\nthree\n");** will print

one
two
three

Each word is displayed in a separate line because of the new line character, '\n'.

Consider the code segment

int x;
float y;
x = 10;
y = 10.5;
printf("%d %f", x, y);

The output is:

10 10.500000

The floating-point values are displayed with respect to six decimal places of accuracy by default. If we need two decimal places of accuracy, then the formatting specification **%0.2f** has to be used. Here the decimal fraction is restricted to two decimal places of accuracy.

Hence the statement, **printf("%0.2f", y);** will display the output as 10.50. To control the total width while displaying the output, the width can be included along with the formatting specifications. i.e.,

printf("%10d %10.2f", x, y);
will print
bbbbbbbb**10**bbbbbb**10.50**   /* b represents a blank space */

Input from keyboard

To read a value from the keyboard (standard input), the function **scanf()** is used. The prototype of scanf() is similar to the prototype of printf(). It can take variable number of parameters.

The code segment to read a value for an integer variable x is given below:

int x;
 scanf("%d", &x);

While the scanf() function is being executed, the system waits for the user's input. The user has to provide data through keyboard. The data will be placed in the location of x only after "Enter" key is pressed in the keyboard. The second parameter of the above scanf() function is **&x**, which represents the address of x. **&** is the **address of** operator and when it is being used with a variable, it provides the address of that variable.

4.3.2 User-defined Functions

When a specific task is to be repeated numerous times or at different places in a program**,** it is convenient to transfer the code associated with that task into a function. The user or programmer can write functions to define specific tasks that may be used at many points in a program. Hence, a function is also a kind of program, which

233

contains a set of code to be executed sequentially to obtain the desired result. The function which calls another function is termed as **calling function** and the other is termed as **called function**. The called function may or may not have parameters.

Functions are invoked by a function call. The function call specifies the function name and provides necessary information as parameters that the called function needs in order to perform its specific task. This is a good practice in C programming to declare functions as variables were being declared. A function declaration may be called as a **function prototype** or a **function model**. The function prototype has four components.

- Name of the function
- Return value type
- Number of parameters
- Type of each parameter

For example, consider a user-defined function that will add two numbers and return the result to its caller. The function prototype looks like,

int add(int, int);

In the above function declaration statement, **add** is the name of the function, which takes two parameters each of type integer respectively and returns an integer. The function is defined as follows:

```
int add(int a, int b)   /* function header  statement */
{
    return (a+b);
}
```

A User-defined function

The function prototype i.e. function declaration statement is terminated by semicolon and the function header statement is not terminated with semicolon. The function header statement is the first statement while the function is being defined.

Defining a function means to write a set of instructions (code) within curly braces { }. The code written within the curly braces is called as function body or a block. All variables declared in function definitions are local variables. They are known only in the function in which they are defined. A function's parameters are also local variables. The parameters provide the means for communicating information between the calling function and called function.

The complete program that invokes the **add** function is given below:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a, b,c;
    int add(int, int);
    a = 12;
    b = 11;
    c = add(a,b);        /*  a and b are actual parameters */
     printf("%d\n", c);
}
int add(int x, int y)     /*  x and y are formal parameters */
{
     return(x+y);

}
```

Actual Parameters are the parameters defined in the calling function and they have the actual values to be passed to the called function.

Formal Parameters are the parameters defined in the called function and they receive the values of the actual parameters when the function is invoked.

In the above program, when the assignment statement **c = add(a, b);** is being executed, the program control is transferred to the **add** function. The parameters **a** and **b** are called as actual parameters since they have the actual values to be passed to the function when the **add** function is invoked. These values are received by the formal parameters **x** and **y** of the called function. When the **add** function is called, the values of the actual parameters are copied to the formal parameters on one to one correspondence basis and this mechanism is called as "**call by value**". The **add** function returns the result to calling function. As the statement **c = add(a+b);** has a function call expression in its right hand side, the value of the expression is the return value of the **add** function.

When the function execution is over, the program control is returned to the calling function to the place from where it is transferred in the case of a function call expression. In case of a function call statement, the program control is returned to the next statement in the calling function. It is also important to note that the local variables' values of the called function are lost and the local variables themselves have been destroyed when the function execution is completed. Local variables are the variables those are declared, defined and used within a function.

In the above example, the **add** function is invoked using call by value. When the parameters are passed call by value, a copy of the parameter's value is made and passed to the called function. Changes to the copy in the called function do not affect the original variable's value in the calling function. If the called function knows the address of the local variable of the calling function then the called function can modify the local variable's value of the calling function. This can be achieved by the "**Call by address**" concept

236

Example:

Consider the following program segment:

```
#include <stdio.h>
main()
{
    int i;
    void change(int *);
    i = 20;
    change(&i);
    printf("%d\n", i);
}
void change(int *x)
{
    *x = 23;
}
```

The program output should be 23, that is, the value of the local variable of the calling function is changed. The formal parameter of the **change** function is an integer pointer **x** which receives the address of the local variable **i** of the calling function, thus **x** points to **i**. The assignment statement in the called function, ***x = 23;** assigns a new value in the location pointed by **x**. Since **x** points to **i**, the value of **i** is modified.

As seen already, the indirection operator ***** is used to retrieve the value contained in a location which is referred by a pointer. Consider the following code segment:

```
int i = 32;
int *p;
int m;
p = &i;
m = *p;
```

237

The ***p** which is in the right side of the assignment statement retrieves the value from the location pointed by **p**. It is nothing but the value of the variable **i**. Using ***p** on the right side, it is possible only to retrieve the value and not possible to modify the value contained in the location. In the **change** function, the indirection operator is used along with the pointer variable **x** on the left side of the assignment statement. If the indirection operator is used in the left side, it gives the location value, i.e. the address and not the contents. Hence, in the statement ***x = 23;** the value 23 is stored in the location pointed by **x**.

If the indirection operator is used in the right side, it gives the data value, that is, it is used to retrieve the value and not to modify that value. It is read-only value. If the indirection operator is used in the left side, it provides the location value, where the contents can be modified. Yet another important concept of a function is explained using the following code segment, which behaves differently that you may not be aware.

```
int i = 10;
printf("%d %d", i, i++);
```

If you think that the expected output is 10 10, or may be 10 11, you are wrong. There is a hidden concept in this program. **In C language, when a function is invoked, its parameters are stored onto a stack from right to left.** A stack is a last-in - first-out (LIFO) structure. Consider a stack of plates or tumblers. In a dining room, the plates or tumblers are placed one above the other. The last plate or tumbler placed (which is on the top) will be taken out first and then the next. A stack is pictorially represented as follows:

238

```
┌─────────┐
│ %d %d   │ ──────▶  top of the stack
├─────────┤
│         │
│   11    │
│         │
├─────────┤
│         │
│   10    │ ──────▶  the first elemen
│         │
└─────────┘
  stack              pushed onto the stack
```
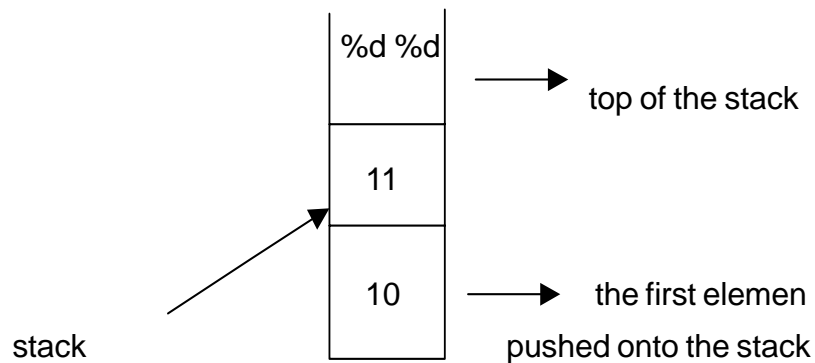
When the above printf() function is called, the first value pushed onto the stack is the value of the expression i++. Since it is a postfix expression, the value of the expression is 10 and it is pushed onto the stack. Then the value of the variable i gets incremented by 1 because of the increment operator. The value of i becomes 11.   The next parameter (from right to left) to be pushed onto the stack is the value of the variable i. Hence, the value 11 is pushed onto the stack and then the last parameter, that is, the control string "%d %d" will be pushed onto the stack. When there are no more parameters to be pushed onto the stack, the printf() function starts to pop out the elements from the top of the stack.   The first element to be popped out is "%d %d". The function analyses the control string, since the first occurrence is %d, it will pop out the next element from the stack (value 11) and then displays it. For the next %d, the function pops out the next element from the stack (value 10) and displays it. The stack becomes empty. You can see the output

1 1 1 0

You can understand now, how the C function works.

4.4 Storage Classes

So far, we have seen that a variable has the following attributes:

- Name
- Type
- Value

239

Storage class is another attribute that is associated with the variable. C provides four storage classes:

- auto
- static
- register
- extern

A variable's storage class is used to determine its scope and lifetime. **auto** variables are actually local variables. They are created when the function in which they are declared is entered, and they are destroyed when the function is exited. The variables declared within a function are local to that function. Their scope and lifetime are limited within that function, which means, all the local variables are destroyed when the function execution is over. We cannot access the values of the local variables outside the function, i.e., the scope of the local variables is within the function in which they have been declared. Consider the following program:

```
#include <stdio.h>
main()
{
  add();
  add();
}
add()
{
    int i = 0;
    i = i + 1;
    printf("%d\n", i);

}
```

The output of the above program will be
        1
        1

The variable **i** is a local variable in the **add** function. Each time the **add** function is called, the variable **i** gets recreated and initialized to 0 and hence the result will be always **1**. If the variable has been declared as a **static** variable, its value will be retained even after the function execution is over. Change the **add** function as follows:

```
add()
{
    static int i = 0;
    i = i + 1;
    printf("d\n", i);
}
```

The variable **i** has been declared as a static variable. It is local to the **add** function. Its scope is still within the **add** function and its value cannot be accessed by any other function. The static variables are created only once during the first call of the function. The main advantage of static variables is that their values are retained even after execution of the function. So, each time, the function **add** is called, the value of the variable **i** gets incremented and the output will be

1
2

The above program is rewritten by declaring **i** as a global variable. Global variables are declared before the main() function. They can be accessed and modified by all the functions in the program.

```
#include <stdio.h>
int i = 0;
main()
{
    add();
    add();
}
add()
{

  i = i + 1;
  printf(“%d\n”,i);

}
```

The output of the above program will be:

1
2

In the above program, **i** is a global variable and it can be accessed and modified in all functions. The static variables retain the characteristics of local variables. The life time of the static variable as well as the global variable ends only when the entire program execution is over. The scope of the static variable is to the function in which it has been declared. The scope of the global variable is to all the functions in the program. Except these differences, the global variables declared outside any function are static by default. But a static variable is not a global variable.

The **register** variables behave like **auto** variables. If a variable is declared with **register** storage class, its value is placed in one of the computer's high-speed hardware registers. If the compiler does not find sufficient registers to use, it may ignore **register** declarations. The register variables are used to speed up operations, by reducing memory access time.

Global variables are accessed in the functions of a program stored in one file. If the global variables have to be accessed by the functions in a file other than the one in which they are declared, the **extern** storage class can be used. For example, if we define global integer variable **count** in one file, and refer to it in a second file, the second file must contain the declaration **extern int count;** prior to the variable's use in that file. The **extern** variables have global scope and the lifetime is throughout the execution of the program.

4.5 Conditional Statements

4.5.1 if statement

In C, the conditional statements rely on the idea of Boolean expressions. The **if** statement controls conditional branching. The Boolean expression or in other words a relational expression will yield a **true** or **false** value. In C, a value other than 0 is true and 0 is considered as false. The body of an if statement is executed if the value of the expression is true i.e., non-zero. There are two forms of **if** statement

if(relational expression)
        statement;

if(relational expression)
     statement1;
else
     statement2;

In the first form, if relational expression is true (non-zero), the statement is executed. If the expression is false, the statement is ignored. In the second form, which uses **else**, statement2 is executed if the expression is false. With both forms, control then passes from the **if** statement to the next statement in the program.

Here is a C program demonstrating a simple **if** statement:

```c
#include <stdio.h>
main()
{
        int x;
        printf("Enter an integer: ");
        scanf("%d", &x);
        if (x > 0)
           printf("The value is positive\n");
}
```

This program accepts a number from the user. It then tests the number using a conditional expression of the if statement to see if it is greater than 0. If it is, the program prints a message. Otherwise, the program is silent i.e., there will be no output. The **(x > 0)** portion of the program is the Boolean expression. C evaluates this expression to decide whether or not to print the message. If the Boolean expression evaluates to **true**, then C executes the single line immediately following the **if** statement (or a block of lines within braces immediately following the if statement). If the Boolean expression is **false**, then C skips the line or block of lines immediately following the **if** statement. Consider another example:

```c
#include <stdio.h>
main()
{  int x;
   scanf("%d", &x);
   if (x < 0)
     printf("The value is negative\n");
   else if (x == 0)
    printf("The value is zero\n");
   else
     printf("The value is positive\n");

}
```

244

In this example, the **if-else-if** construct is used which we can call as a nested if-else structure. The nested if-else structure is used to perform some operations based on choices. Consider a simple arithmetic problem in which add, subtract, multiply and divide operations are to be carried out depending on the choice. The options may be displayed. The program can be written in two ways:

Using simple if

```
#include <stdio.h>
main()
{   int a,b,c,choice;
    scanf("%d%d", &a,&b);         /*b is not zero */
    printf("1. addition\n");          /* option 1 */
    printf("2. subtraction\n");      /* option 2 */
    printf("3. multiplication\n");  /* option 3 */
    printf("4. division\n");          /* option 4 */
     scanf("%d", &choice);
    if(choice == 1)
      c = a + b;
    if(choice == 2)
       c = a - b;
    if(choice == 3)
      c = a * b;
    if(choice == 4)
      c = a / b;
    printf("the result = %d\n", c);

 }
```

The above program estimates the value based on the choice and prints the result. The number of comparisons made in the program is 4 irrespective of any choice. It performs comparisons unnecessarily. When (choice == 1) is true, it is not necessary to evaluate the other conditions and may be skipped. The use of nested if-else construct will solve this problem.

Using nested if-else construct

```c
#include <stdio.h>
main()
{        int a,b,c;
   int choice;
   printf("Enter two integers: ");
    scanf("%d%d", &a,&b);        /* b is not zero */
   printf("1. addition\n");           /* option 1 */
   printf("2. subtraction\n");      /* option 2 */
   printf("3. multiplication\n");   /* option 3 */
   printf("4. division\n");          /* option 4 */
    printf("Enter your choice: ");
    scanf("%d", &choice);
   if(choice == 1)
     c = a + b;
   else
    if(choice == 2)

         c = a - b;
     else
      if(choice == 3)
        c = a * b;
      else
       if(choice == 4)  /*comparison is optional */
         c = a / b;

   printf("the result = %d\n", c);
         }
```

The above program uses nested if-else construct. If the first condition is true, that is, (choice == 1) is true, the statement c=a+b; is executed and the statement following the **else** is skipped. Therefore, if the first condition is true, only one comparison is made and all the other comparisons are skipped. Only when the first condition fails, the program continues to compare the second condition and it goes on similarly. This program works faster than the previous program. But if choice is 4, both programs have to do four comparisons.

4.5.2 Switch - case statement

The switch – case statement is the modular replacement of the cumbersome nested if-else structure. The switch and case statements help to control complex conditional and branching operations. The switch statement transfers control to a statement within its body. The syntax of the switch – case statement is as follows:

```
switch (conditional expression)
{
    case constant-expression 1:
                        ……………..
                        break;
    case  constant-expression 2:
                        ……………..
                        break;
        .
        .
    default:
                        ……………..
}
```

The type of switch (conditional expression) and the case constant-expression must be integer type. Control passes to the statement whose **case** constant-expression matches the value of switch(conditional expression). The switch statement can include any number of case statements, but no two case statements within the same switch statement can have the same constant-expression. Execution

of the statement body begins at the selected case statement and proceeds until the end of the body or until a **break** statement transfers the control out of the body. The break statement is used to end processing of a particular case statement within the switch statement. Program continues to the next case, executing the statements until a break or the end of the statement is reached. The **default** statement is executed if no case is equal to the value of switch(conditional expression). If the default statement is omitted, and no case match is found, none of the statements in the switch body are executed. The default statement is an optional statement and it need not come at the end; it can appear anywhere in the body of the switch statement. The program to solve the simple arithmetic problem using nested if-else structure is rewritten using switch – case statement as follows:

```c
#include <stdio.h>
main()
     {  int a,b,c;
        int choice;
        printf("Enter two integers: ");
        scanf("%d%d", &a,&b);          /*b is not zero */
        printf("1. addition\n");          /* option 1 */
        printf("2. subtraction\n");      /* option 2 */
        printf("3. multiplication\n");  /* option 3 */
        printf("4. division\n");          /* option 4 */
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch(choice)
        {
          case 1:
             c = a + b;
             printf("%d", c);
             break;
          case 2:
             c = a - b;
             printf("%d", c);
             break;
```

```
        case 3:
           c = a * b;
           printf("%d", c);
           break;
        case 4:
           c = a / b;
           printf("%d", c);
           break;
        default:
            printf("the choice is out of range\n");

}
```

The above program is modular and has easy readability than the program written using nested if-else structure. Consider another example that displays whether the given character is a vowel or consonant.

```
char ch;
ch = 'a';
switch(ch)
{
      case 'a':
      case 'e':
      case 'i':
      case 'o':
      case 'u':  printf("the given character is
                    vowel");
                    break;
       default:  printf("the given character is
                      consonant");

}
```

In the above example, as the value of the character ch is 'a', the first case is satisfied and since there is no break statement, the program continues to the next case, executing the statements until a break or the end of the statement is reached. Hence, if the value of the character ch is 'a' or 'e' or 'i' or 'o' or 'u', the statements belong to the case 'u' have been executed, otherwise, the statement belongs to the default case is executed.

4.6 Control Statements

The body of an **if** statement will be executed only once when the condition is satisfied, i.e., the relational expression contained within the **if** is true. In some situations, it may be necessary to repeat a set of statements until certain specified conditions are met. To achieve this, control or looping statements are required. A loop is a part of a program that comes back and repeats itself as many times as necessary. In C programming, there are three control statements namely, while, for and do while.

4.6.1 while statement

The **while** statement is used to execute the set of statements repeatedly till the condition specified remains true. In the while statement, the condition is tested at the entry level. The number of times the loop gets executed is controlled by a control variable. The following program prints the first ten natural numbers each in one line. Here, a loop is required to execute the printf() function for 10 times.

```
#include <stdio.h>
main()
{
   int i;
   i = 1;                  /* Initialization */
   while(i <= 10)          /*condition */
   {
       printf("%d\n", i);  /*processing statement */
       i = i + 1;          /*updating */
   }
}
```

250

In the above program, the variable **i** is used as a control variable that will control the execution of the while loop. The control variable is properly initialized just before the while statement. Since the above program is to print the first 10 natural numbers and the first number to be printed is 1, the control variable is initialized to 1. The loop has to be executed for 10 times and hence the condition **i <= 10** is used. The body of the while loop is executed if the specified condition is true. The control variable is incremented by 1 every time the loop executes so that the loop will be repeated exactly for 10 times. If the test condition fails, the control is transferred out of the loop. On exit, the program continues with the statement immediately after the body of the loop. If you look into the program once again, the control variable is tested immediately after the initialization of it and it is incremented just before the end of the while loop (just before the closing curly brace). The syntax of the while statement is as follows:

Initialization of the control variable
while(condition)
{
    ……..; ⟶
                  processing statements
    ……..; ⟶

    updating the control variable;
}

The control variable is tested against a condition in the while statement and it should be properly updated within the while loop for proper termination of the loop. If the updating line is missing, the value of the control variable will be always 1 and the loop never ends.

If we omit the initialization, condition and the updating statements from the above program, it contains only one processing statement, which is **printf("%d\n", i);** that will serve the purpose of this program.
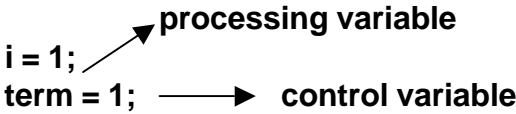
Let us see another example. Write a program to generate the following sequence:

1, 2, 4, 7, 11, 16 ………………… 15 terms

The second number in the sequence is obtained by adding 1 to the previous number, the third number is obtained by adding 2 to the previous number, the fourth number is obtained by adding 3 to the previous number and so on. The loop is to be executed for 15 times. The program is as follows:

```
#include <stdio.h>
main()
{
   int term;
   int i;
                   processing variable
   i = 1;
   term = 1;        control variable
   while(term <=15)
   {
      printf("%d\n", i);
      i = i + term;
      term = term + 1;
   }
}
```

In the above program, the variable **term** is used to control the execution of the loop and the variable **i** is used as a processing variable. The logic of the program depends on the statement **i = i + term;** where the next number in the sequence is obtained by adding the value of **term** to the previous number and the **term** is incremented by 1 every time the loop executes. The success of the program that uses a while loop depends on the proper use of control and process variables. The control variable is to be initialized, tested in the condition and then updated in the loop. The process variable is used in the logic.

A while statement can be nested within another while statement. The inner while statement executes faster than the outer while loop, that is, the inner while loop terminates its execution before the termination of the outer while loop. Each while loop must have proper initialization of its control variable, proper condition to control the number of times the loop gets repeated and proper updating of the control variable.

Example:

Write a program to print 1 once, 2 twice in the next line, 3 thrice in the next line and continue up to 10, that is, 10 written ten times. This problem requires two loops; the outer loop controls the number of the times the inner loop has to be repeated. Each time the inner loop is executed, the number of times it gets repeated depends on the outer loop control variable's value. The program is as follows:

```
#include <stdio.h>
main()
{       int i, j;
        i = 1;
        while(i<=10)
        {
         j = 1;
         while(j <= i)
         {
              printf("%d ", i);
              j++;
         }
         printf("\n");
         i++;
        }
}
```

The output is as follows:

```
1
2 2
3 3 3
………
10 10 10 10 10 10 10 10 10 10
```

The inner loop of the above program is executed for i number of times where i is the control variable of the outer loop.

4.6.2   for statement

The **for** loop in C is simply a shorthand way of expressing a while statement. For example, suppose you have the following code in C:

```
x=1;
while (x<=10)
{
   printf("%d\n", x);
   x++;
}
```

You can convert this into a **for** loop as follows:

```
for(x=1; x<=10; x++)
{
   printf("%d\n", x);
}
```

Note that the while loop contains an initialization step (x = 1), a test step (x <= 10), and an increment step (x++). The **for** loop lets you put all three parts into one line. In the **for** loop also, the condition is tested at the entry level. The body of the above **for** loop executes 10 times. Here, the control variable is initialized first and then it is tested. If the test condition is true, the body of the loop is executed; otherwise

the loop is terminated and the execution continues with the statement that immediately follows the loop. After the body of the loop has been executed, the control is transferred to the **for** statement, where the control variable is updated and then retested. The loop continues till the condition remains true.

The syntax of the **for** loop is as follows:

for(initialization; condition; updation)
{
      body of the loop;
}

A **for** loop can also be nested. The problem of printing 1 once and 2 twice can be written using nested **for** loops as follows:

```
#include <stdio.h>
main()
{
        int i;
        int j;

        for(i=1;i<=10;i++)
        {

        for(j=0;j<=i;j++)  ─────┐
        printf("%d",i);         ├──▶  Inner Loop

         printf("\n");
        }
}
```

The inner for loop has only one statement and the outer for loop has two statements. If the body of the loop contains only one statement, there is no need for curly braces to enclose the body of the loop.

The problem which is solved using **while** loop can also be solved using a **for** loop. It is a good programming practice to use **for** loop where the number of times of repetition is known precisely. The **for** loop is a definite repetition loop where we know in advance exactly how many times the loop will be executed. The **while** loop is preferred when the number of repetitions is not known before the loop begins executing.

Example:

```
#include <stdio.h>
main()
{
   char ch;
    int count = 0;
    ch = getchar();
    while(ch != '\n')      /*condition*/

    {
          count++;
          ch = getchar();
    }

printf("the number of characters entered: %d\n", count);

}
```

In the above example, the number of times the loop will be executed is not known until the user presses the Enter key in the keyboard. Once the variable ch obtains its value as the new line character (Enter key pressed), the test condition fails and the control is transferred to the next statement immediately after the body of the while loop. For this problem, the while loop is preferred since the number of characters to be entered by the user is not known precisely. In the above program, the function getchar() is used to read a character at a time from the keyboard and it is a pre-defined function.

256

4.6.3 do – while statement

In the while loop, the condition is tested at the entry level. If the condition fails at very first time, the body of the loop will not be executed at all. In case of do – while statement, the condition is tested at the exit level and hence the body of the loop is executed at least once whether the condition is true or false. At the end of the do – while loop, the condition is tested and if it is true, the loop gets executed once again. This process continues as long as the test condition is true. When the test condition becomes false, the loop is terminated and the control is transferred to the statement immediately after the **do – while** statement.

Example:

```
x = 14;
do
{
     y = x + 2;
     x—;
} while (x > 0);
```

In this do-while statement, the two statements y = x + 2; and x—; are executed regardless of the initial value of x. Then x > 0 is evaluated. If x is greater than 0, the body of the loop is executed again and the condition x > 0 is reevaluated. The body of the loop is executed repeatedly as long as x remains greater than 0. Execution of the do-while statement terminates when x becomes 0 or negative. The body of the loop is executed at least once.

4.7 Arrays

An array is a collection of homogeneous elements i.e. elements of similar data type. Use of arrays makes programming easier in many cases. For example, you might want to find the maximum among 3 integers for which 3 independent variables may be used and can be solved using simple **if** statements as shown below:

```
#include <stdio.h>
main ()
{       int a, b, c;
        int max;
        printf("Enter the 3integers:");
        scanf("%d %d %d",&a,&b,&c);
        max = a;
         if(b > max)
            max = b;
        if(c > max)
            max = c;
        printf ("%d is the maximum ",max);

}
```

If the above program is extended to find the maximum among 100 integers then the program might use 100 independent variables and 100 **if** statements. This makes the program more complex and lengthy and moreover it is not the correct method of programming. An easier way is to declare an array of 100 integers:
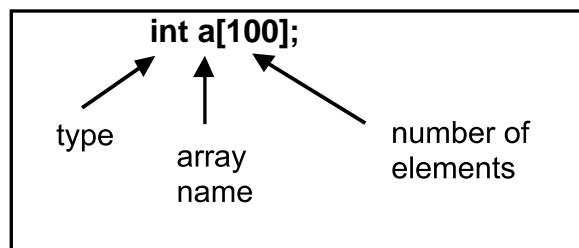
The 100 different integers inside the array are accessed by an appropriate index. It reduces the complexity of using 100 variables for storing 100 different values. An array declaration specifies the **name of an array** and the **type** of its elements. A constant expression should be used within the square brackets that specify the **number of elements** in the array and its value must be greater than zero. The storage associated with an array is the storage required for all of its elements. The elements of an array are stored in contiguous and increasing memory location from first to last element.

In C, arrays can be single dimensional or multi-dimensional. The array elements can be accessed using indices. A single dimensional array of 10 integers is declared as follows:

int a[10];   /* => array declaration statement */

When the program starts to execute, the complier allocates 20 bytes in the main memory to store the 10 elements of this array, since each integer requires 2 bytes of memory. These 10 elements are stored in contiguous memory locations. An array index starts from zero. If there are n elements in an array, they can be accessed using the indices, which start from zero to n-1. The value for the array elements can be assigned as follows:

```
a[0]=10;
a[1]=20;
.
a[9]=100;
```

One of the nice things about array indexing is that you can use a loop to manipulate the index. For example, the following code initializes the values in the array sequentially:

```
int i;
int a[10];
for(i=0;i<10;i++)
     scanf("%d", &a[i]);
```

The second parameter &a[i] in the above scanf() statement indicates the address of the ith element where i=0 to 9.The scanf() function gets the value and places it in the appropriate location (address).

An array can be initialized as follows:

int a[3]={10,15,20};

The array elements are stored in the memory as given below:

948	950	952	⟶ address
10	15	20	⟶ Elements

a[0]   a[1]   a[2] ⟶ accessing array elements
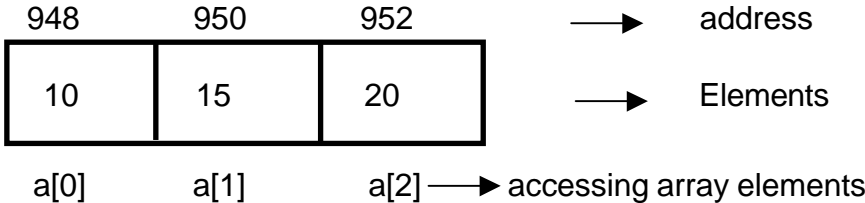
Each integer occupies 2 bytes of memory. The compiler allocates 6 consecutive bytes, which is called as a **block of memory** and has a starting address of 948 as shown in the above figure. The addresses of the elements a[0], a[1] and a[2] are 948, 950 and 952 respectively.

The address of the first element is represented as **&a[0]** and it is 948 in our example. The value contained in this location is 10 and it can be retrieved using the expression a[0]. Hence the value of a[0] is 10 and its address is 948. The compiler after allocating memory for this array stores the starting address in the array name itself. Hence,

both **a** and **&a[0]** represent the starting address. We can say that the array's name **a** points to the array. It means that **a** points to the starting address. In other words, it points to the first element of the array.

```
      948         950      952
   ┌─────────┬─────────┬─────────┐
   │   10    │   15    │   20    │
   └─────────┴─────────┴─────────┘
 a     a[0]      a[1]      a[2]
```
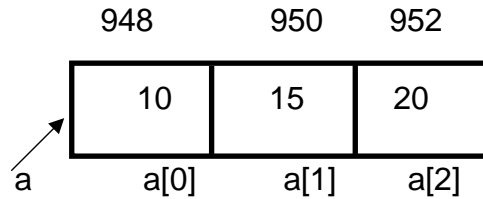
As we have seen already, the indirection operator (unary *****) is used to retrieve the value contained in a memory location and hence ***a** will yield the value 10. Hence the following expressions

a[0],

 *a

 and ***(&a[0])**

will provide the same value 10.

So, we can say,

a[0] **<=>** *a **<=>** *(&a[0])

Here the symbol **<=>** is used to represent "all are one and the same" and it is not a C language operator.

From the above discussions you can understand that the starting address or the base address of an array is stored in the array's name itself. Since the address is stored in the array name it becomes a **pointer**. Hence the array name itself points to the first element of an array. The array's name always points to the starting address of the array and the base address of an array cannot be modified. It means that we cannot make the array's name to point to the second element in the array. We cannot modify the address stored in the array's name and hence it is a pointer constant.

261

The arrays and pointers are closely related to each other. Consider the pointer variable **x.**

int *x;

Consider an array of 3 integers as already explained,

int a[3] = {10, 15, 20};

Here, **x** is a pointer variable which can assume an address of another integer and **a** is a constant pointer to an integer, i.e., to the first element of the array. The base address of the array can be assigned to the pointer variable **x**.

x = a;

We made **x** to point to the starting address of the array. Since **x** is a pointer variable, we can make it to point to any other element in the array during execution of the program. Pointer arithmetic is an excellent feature in C programming. The following operations can be carried out using pointers.

- An integer can be added to or subtracted from a pointer
- Two pointers can be subtracted

Consider the following illustration:

948	950	952
10	15	20
a[0]	a[1]	a[2]

x ⟶
a
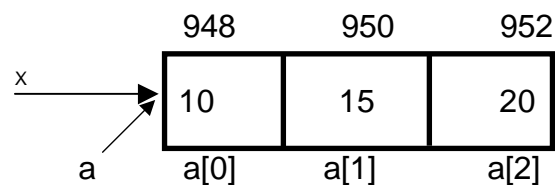
Both **x** and **a** point to the first element of the array because of the statement **x=a;**. Here, both **x** and **a** point to the address 948. Adding 0 to both will not change the address values of **x** and **a**. We can use the indirection operator ***** to retrieve the value contained in that location. The expressions *(a+0) and *(x+0) both yield the value 10. The expression *(a+0) is the pointer notation to access the first element value in the array. The first element of the array is usually obtained using array notation (square bracket notation) **a[0]**. Consider the following identities:

$$x + 0 \ <=> \ a + 0$$

$$*(x + 0) \ <=> \ *(a + 0)$$

Since *(a+0) and a[0] are one and the same, we can write,

$$*(x + 0) \ <=> \ x[0]$$

Here, the pointer variable x is used with a subscript (index) 0 to access the first element of the array pointed by it. So, a pointer can be indexed. This is possible only when the pointer points to a block of continuous memory locations. The expression **x+0** is the address of the first element and ***(x+0)** is the value contained in that address. Adding 1 to the array name **a** or to the pointer variable **x** will give the address of the second element in the array.

$$x + 1 \ <=> a + 1$$

An integer is added to an address. We are performing pointer arithmetic and not an ordinary arithmetic. Remember this important rule – **while we are performing pointer arithmetic, the arithmetic is scaled by the size of what type of element it points to.** Here **x** points to an integer and the size of the integer is 2 bytes and hence the scaling factor is 2. Therefore,

$$x + 1 \ <=> \ 948 + 1 * 2 \ = \ 950$$

$$a + 1 \ <=> \ 948 + 1 * 2 \ = \ 950 \ <=> \ \&a[1]$$

....
....

a + i < = > &a[i] < = > x + i < = > &x[i]

The expression **a+i** is the address of the **ith** element in the array and ***(a+i)** is the value of the **ith** element.

The above discussions will make you to understand the following:

- Arrays and pointers are closely related and
- A pointer can be simulated as if it were declared as an array

Example: Write a program to read values for an array of 10 integers.

```
#include <stdio.h>
main()
{
   int i;
   int a[10];
   for(i=0;i<10;i++)
     scanf("%d", a+i);

}
```

Note that, **a+i** is used in place of **&a[i]** in the scanf() function, because we have already proved that they are one and the same. When i = 0, the expression a+i will provide the address of the first element and when i = 1, it will provide the address of the second element and so on.

So far we have seen integer arrays and how to handle them using pointers. Handling character arrays is our next focus. An array of 24 characters is declared as follows:

char name[24];

Here, name is an array of 24 characters. Twenty-four consecutive bytes have been allocated and the starting address is stored onto the array name **name**. A **string** can be defined as a collection of characters terminated by a null character ('\0'). So, if it is required to handle a string, an array of characters is needed. To read a string from the keyboard, the following statement is used:

scanf("%s", name);

The formatting specification character %s is used here to read a string. '%s' will treat the blank space within the string as a delimiting character and hence it cannot be used to read a string which contains blank spaces. When the above statement is being executed, it waits for the user's input. All the characters entered by the user will be stored from the starting address specified by **name** and a null character '\0' will be automatically appended at the end. To print the string, we can use

printf("%s", name);

The formatting specification character '%s' will retrieve characters from the starting address and displays them until it receives a null character.

Since string is a collection of characters (an array) its type will be **char ***. The data type associated with the string constant is **char *** and hence its value should be an address and not the contents of the string. The value of a string constant is the starting address where the string has been stored in the memory.

Consider the code segment,

if("rama" == "rama")
 printf("equal");
else
  printf("not equal");

Here, the strings are compared as any other constants compared with the equality operator. The C compiler accepts this syntax and the code segment produces the output **not equal**. Even though the identical strings are used for comparison in the above code segment, strings are stored in two different locations. We know already that the value of a string is its starting address and not the string itself. Actually the locations are compared and they are not equal. Therefore the code segment produces the output **not equal.**

We will see a program to count the number of characters in a specified string. The **string.h** file provides declarations of many string handling functions. The function strlen() is used to find the length of the string. The strlen() is a pre-defined function whose prototype is available in string.h file. The prototype of the strlen() function is:

int strlen(char *);

The strlen() function accepts string (a pointer to a character) as a parameter and returns its length as an integer.

```
#include <stdio.h>
#include <string.h>
main()
{       char name[24];
        int len;
         printf("enter a string: ");
        scanf("%s", name);
   len = strlen(name);
        printf("%d\n", len);

}
```

The above program reads a string from the key-board, finds its length and displays the result.

When an array has to be passed as a parameter to a function, pointers can be effectively utilized. We will write our own function to find the length of the string. In the following code, the function lenstr() is a user-defined one.

<u>Version 1:</u>

```
int lenstr(char *s)
{
        int count = 0;
        while(s[count] != '\0')
         count++;
        return(count);

}
```

If the string "rama" is passed as a parameter to the lenstr() function, it is received by the formal parameter which is a character pointer and hence **s** points to the string "rama". As the loop starts its execution with count = 0, **s[0]** represents the first character in the string, that is **'r'** and it is not equal to null character. Since the condition is satisfied, the body of the loop is executed. The count gets incremented. The loop gets executed repeatedly until s[count] becomes '\0'. The function returns the value of the count, which represents the number of characters in the string.

Version 2:

```
int lenstr(char *s)

{      int count = 0;
       while(*s != '\0')

       {
           count++;
           s++;
       }

       return(count);
}
```

When the control enters into the loop for the first time, ***s** has the value 'r', the first character in the string "rama". In the body of the while loop, the pointer s gets incremented so that it points to next character in the string. The process continues until ***s** becomes '\0'.

Version 3:

```
int lenstr(char *s)
{
           char *start, *end;
           start = end = s;
           while(*end)
             end++;
           return(end  – start);
       }
```

All the character pointers **start**, **end** and **s** are made to point to the string "rama", assuming that when the function lenstr() is invoked, the string "rama" is passed as a parameter.

end 948 949 950 951 952

s

| r | a | m | a | \0 |
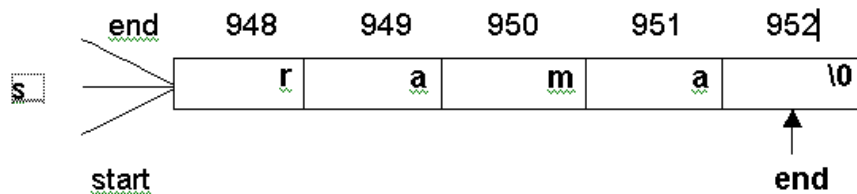
start   end

The start, end and s are pointing to the address 948, which is the address of the first character. Use while loop to traverse the end pointer until it points to '\0'. When the loop completes its execution, the pointer **end** points to the address 952. Examine the test condition in the while statement. In C, a value other than 0 is treated as true. First time, ***end** has the value **'r'** which is other than 0 and hence the condition is satisfied. When ***end** has the value '\0' which is 0, the condition becomes false. When the loop terminates, **end** points to 952 and **start** points to 948. The expression **end – start (**i.e., 952 – 948) has the value 4 and it is returned as the number of characters in the specified string. When one pointer is subtracted from another, the result is the number of bytes in between them. Pointer arithmetic feature alone enables the handling of arrays using pointers.

4.7.1 Multidimensional Arrays

A multidimensional array has been considered as an array of arrays in C language. A two dimensional array has the following declaration:

int a[3][3];

The above declaration represents a 3 X 3 matrix. There are 9 elements and the compiler allocates 18 consecutive bytes to store the elements of the matrix. The first dimension represents the number of rows and the second dimension represents the number of columns. To read values to this matrix (two dimensional case), we need two

269

indices, one is row index and another is column index. The array index starts from 0 in C language. We can access the first element using a[0][0]. Consider the following code segment.

```
int a[3[[3];
int i;
int j;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
scanf("%d", &a[i][j]);
```

The above segment reads values for the matrix row wise. For i = 0 in the outer **for** loop, the inner loop executes for j = 0, 1, 2 (three times) and hence it reads the first row of three integers. It continues for i = 1 and i = 2. When you are giving input, each integer is separated by one or more blank spaces.

There are 3 rows and each row represents an array of 3 integers. It is in the array of arrays form. The first dimension has 3 elements and each element is an array of 3 integers because of the second dimension. Consider the following matrix:

        1   2   3
        4   5   6
        7   8   9

It can be represented in an array of arrays form as follows:

    a[0] → [1   2   3]

    a[1] → [4   5   6]

    a[2] → [7   8   9]

270

The first dimension is an array of 3 elements where each element is an integer array and hence each element is an integer pointer. Hence the declaration **int a[3][3]** can be interpreted as follows. The first dimension is an array of 3 integer pointers and the second dimension is an array of 3 integers. **a[0]** points to the first row of three integers and it is an integer pointer, whereas, **a[0][0]** is the first element in the matrix, that is, the first element of the first row. **a[1]** points to the second row of three integers. The compiler treats the two-dimensional array in this way only. But we are simply assuming that the first dimension represents rows and the second dimension represents columns.

We had seen how to read values for a matrix of size 3 X 3. Next we will write a code segment to print a matrix in row wise.

```
for(i = 0; i < 3; i++)
{
for(j = 0; j < 3; j++)
   printf("%d ", a[i][j]);
printf("\n");
}
```

A matrix addition program which adds two matrices of size 3 X 3 and stores the results in another matrix is given below:

```c
#include <stdio.h>
#include <conio.h>
main()
{
        int a[3][3], b[3][3], c[3][3];
        int i, j;
    /* read values for the input matrix a */
        for(i=0;i<3;i++)
          for(j=0;j<3;j++)
            scanf("%d", &a[i][j]);


    /* read values for the input matrix b */
        for(i=0;i<3;i++)
          for(j=0;j<3;j++)
            scanf("%d", &b[i][j]);
    /* initialize the output matrix c with all elements 0 */
        for(i=0;i<3;i++)
          for(j=0;j<3;j++)
            c[i][j] = 0;
    /* add matrix a and b and store the result in matrix c */
        for(i=0;i<3;i++)
          for(j=0;j<3;j++)
            c[i][j] = a[i][j] + b[i][j];
    /* print the resultant matrix c */
        for(i=0;i<3;i++)
        {
        for(j=0;j<3;j++)
          printf("%d  ", c[i][j]);
        printf("\n");
        }
}
```

Comments are included in the above program in order to understand every part of the program. The following program illustrates about independent handling of rows in a matrix. The program is to find the maximum value in each row of a given matrix of size 3 X 3.

```c
#include <stdio.h>
#include <conio.h>
main()
{       int a[3][3], max[3];
        int maximum(int *);  /* declaration of user-defined function */
        int i, j;
        clrscr();           /* to clear the contents of the screen */
        /* read values for the input matrix a */
        for(i=0;i<3;i++)
           for(j=0;j<3;j++)
              scanf("%d", &a[i][j]);
/* find the maximum in each row and store in the array max */
        for(i = 0; i < 3; i++)
            max[i] = maximum(a[i]);
        /*  print the max array */
        for(i =0; i < 3; i++)
            printf("The maximum value of row %d = %d\n", i+1,
                   max[i] );
}
int maximum(int *x)
{           int mvalue;
            mvalue = x[0];
            for(i = 1; i < 3; i++)
              if(x[i] > mvalue)
                 mvalue = x[i];
            return(mvalue);
}
```

In the above program, a user-defined function has been written to find the maximum value in an array of 3 integer elements. The name of the function is **maximum**. It takes one parameter of **int *** type and returns an integer value, which is the maximum value. This function is called for 3 times from the main() function within a for loop. Each time the **maximum** function is called, a row of three integers has been passed as a parameter, i.e., **a[i],** where **i = 0, 1, 2**. The starting address of each row is received by the formal parameter **x** which is an integer pointer in the called function. As we have seen already, the pointer **x** can be indexed (as it points to an array of 3 elements) to access the elements of the array. The maximum value of each row is stored correspondingly in the array called **max**. Hence, each row of elements can be handled independently by using its starting address.

4.8.   Structures

Structures are derived data types in C language. They are constructed using variables of other types. Structures are used to create user-defined types. Structures are commonly used to define records to be stored in files. A file is a collection of records. A record is a collection of fields of information.

A student record may contain the following fields:
Roll number, Name and Age
The values of these fields may be
1001   Anand   18

The above record contains information about the student "Anand" whose roll number is 1001 and his age is 18.

An array is a collection of elements of same data type. A structure is a collection of elements of different data types. An array is a homogeneous collection whereas a structure is a heterogeneous collection of elements.

The fields of the student record shown in the above example are of different data types.

Roll number: an integer field
Name: an array of characters
age: an integer field

Consider the following structure definition:

```
struct student
{
   int rollno;
   char name[24];
   int age;
};
```

The **struct** is a keyword, which is used to define a structure. The identifier **student** is a structure tag or a tag name. The above definition just tells us about the record structure, that is, how many fields it contains and their respective data types. Definition of a structure will give only the template of the record. It is nothing but the skeleton of the record. Variables (fields) declared within the braces of the structure definition are the structure's members. Members of the same structure must have unique names. The structure definition must end with a semicolon. The structure definition does not reserve any space in memory, that is, memory will not be allocated when defining a structure and hence the members of the structure cannot be initialized within the structure definition.

In the above example, the structure type is **struct student**. Now **struct student** becomes a new user-defined data type. The structure definition creates a new data type that is used to declare variables. Structure variables are declared like variables of other types.

Example:

struct student x, y;

x and y are the variables of type struct student. Each variable has three fields as defined in the structure. A total of 28 bytes will be allocated for each variable of type **struct student**.

It is always better to define the structure as a global entity so that its definition is available to all functions in the program.

The structure variables can be declared global as well. For example,

```
struct student
{
   int rollno;
   char name[24];
   init age;
} x, y;
```

Here the structure variables are declared while defining the structure itself. It is also possible to define a structure without tag name. Then variables must be declared while defining the structure itself.
Without tag name, the structure definition will not be acceptable as a user-defined data type.

Example:

```
struct
{
   int empno;
   float salary;
} x, y ;
```

By using the definition stated in the above example, it is not possible to create new structure variables other than x and y. The structure definition without tag name as well as without declaring variables will become useless or anonymous.

It is not possible to do anything with the following structure definition:

```
struct
{
    int empno;
    float salary;
};
```

4.8.1. Accessing the members of the structure

To access the members (fields) of a structure, dot operator is used. The structure variable is used as a qualifier while accessing its members along with the dot operator. For example, to access the roll number of the student, that is, the rollno field using the structure variable x, the following notation is used:

x.rollno

Fields can be assigned to or can be read from. The following statements can be used to assign the roll numbers for the students x and y.

```
x.rollno  = 1000;
y.rollno  = 1001;
```

To read the members of the student record, the function scanf() can be used as follows:

```
scanf("%d%s%d", &x.rollno, x.name,&x.age);
```

Since the **name** field in the structure (an array of 24 characters) itself provides the starting address of an array, there is no need to provide an explicit & operator in the scanf() function while reading the name of the student x.

4.8.2 Pointers to Structures

Like primitive pointers, structure pointers can also be declared. For example,

struct student *ptr;

In the above declaration statement, **ptr** is a pointer to the user-defined data type struct student. It can be assigned with an address of another structure of the same type.

Example:
struct student s1;
ptr = &s1; /* ptr points to the structure s1 */

To access the members of the structure s1 using the pointer ptr, arrow operator (->) should be used instead of dot operator.

That is, to access the rollno field, the expression ptr -> rollno should be used. (*ptr).rollno is the old syntax.

4.8.3 An Array of Structures

An array of structures can be declared as follows:

struct student x[5];

Here, x is an array of five structure elements. x[0], x[1] ..., x[4] are individual structure elements of type struct student. The members can be accessed as

278

x[0].rollno, x[0].name, x[0].age

To read members for the second element in the array, you can use

scanf("%d%s%d", &x[1].rollno, x[1].name, &x[1].age);

To read an array of 5 student records, a **for** loop can be used

for(i=0;i<5;i++)
scanf("%d%s%d", &x[i].rollno, x[i].name, &x[i].age);

Similarly, a **for** loop is used to print the student records one after the other. The records can be sorted in roll number order, alphabetical order or age wise and separate reports can be printed. Structures are generally used in database operations as they handle records.
Exercises

I State True or False

1. It is a must to draw a flow chart before writing a program.
2. Flow charts are easier than the pseudo codes.
3. An algorithm should have a finite number of steps.
4. An index variable is used in the indefinite iteration.
5. Every time a condition is checked in the indefinite iteration.

II Fill in the blanks

1. A flow chart is drawn _____ writing a program.
2. Understanding a pseudo code is _____ than understanding a flow chart.
3. Every program _____ be represented by a flow chart.
4. A walkthrough _____ find all the bugs in the design.
5. Each step in an algorithm must take a _____ amount of _____ and _____ .

III Answer the following

1. Give two important differences between the flow chart and the pseudo code.
2. Draw the different types of boxes used in the flow chart. Explain each one of its roles.
3. Give two examples of a multi-way branching. Use pseudo code.
4. Give two examples of a two-way branching. Use a flow chart.
5. Give two examples and illustrate the use of an index variable.
6. Using two examples to illustrate definite iteration.
7. Using two examples illustrate indefinite iteration.
8. State three differences between definite and indefinite iterations.
9. Give two examples where multi-way branching is more natural than two-way branching.
10. Explain the fundamental control structures using pseudo code.
11. Give the properties of an algorithm.

IV Programming Exercises (For each problem, draw appropriate flow chart and write pseudo code)

1. Write a C program to print your name 5 times on the monitor.
2. Write a C program to interchange the values of two variables.
3. Write C programs to do the following:
   i. To find the area of a triangle
   ii. To convert the temperature from Fahrenheit to Celsius
   iii. To convert the time in hours : minutes : seconds to seconds
4. Write a C program to find the maximum among two integers
   i. Using if statement

ii. Without using if statement

(*Hint: max = ((a+b) + abs(a-b))/2*

5. Write a C program to find your age in terms of years, months and days till today. (For simplicity, assume each month has equally 30 days)

6. Write a C program to find the sum of the first 10 natural numbers, that is to find $1 + 2 + 3 + ... + 10$

7. Write a C program to generate the Fibonacci series of 15 terms,

0, 1, 1, 2, 3, 5, 8............, (15 terms)

8. Write a C program to count the number of vowels present in your name.

9. Write a C program to find the greatest common factor which will divide the given two numbers.

(*Hint: Use the following user-defined function to find gcf)*

```
int gcf(int first, int second)
{    int temp;
     while(second > 0)
     {
          temp = first % second;
          first = second;
          second = temp;
     }
     return (first);
}
```

10. Write a C program using the above gcf() function to simplify the given fraction. If the input is 16 / 64 then the output is 1 / 4.

11. The number 3025 has a property that when we square the addition of the first half of the number and the next half of the

number, the result will be the same number. That is, square of (30 + 25) is 3025. Write a C program to print all such four digit numbers

12. An Adam number is one which satisfies the following: The reverse of the square of a number is the square of the reverse of that number. That is, if the number is 12 then $12^2 = 144$, the reverse is 441 which is 21^2. Write a C program to find all such numbers between 10 and 100.

13. Write a C program to check whether the given number is in Fibonacci sequence or not.

14. Write a C program to convert the string of lower case letters into upper case and vice versa.

15. Write a C program to convert the decimal number into its equivalent binary number, octal number and hexadecimal number.

16. Write a program to convert binary number into its equivalent decimal.

17. A perfect number is a number that is the sum of all its divisors except itself. Six is the perfect number, since the sum of the divisors of six except itself is $1 + 2 + 3 = 6$. Write a C program to find the perfect numbers between 1 and 10000.

18. An abundant number is one that is less than the sum of all its divisors except itself (12 is an abundant number, since 12 < 1 + 2 + 3 + 4 + 6). Write a C program to find the abundant numbers between 1 and 10000.

19. A deficient number is one that is greater than the sum of all its divisors except itself (9 is a deficient number, since 9 > 1 + 3). Write a C program to find the deficient numbers between 1 and 10000.

20. Write a C program to reverse the contents of a string. If the input string is **rama**, the output should be **amar**.
21. Write a C program to check whether the given string is palindrome or not.
22. Write a C program to arrange the array of 10 numbers in non-increasing order.
23. Write a program to interchange the values of two variables using a function.
24. Write a C program to find the average of 10 numbers.
25. Assume array **a** contains **n** values that are supposed to be in ascending order. Write a C program to perform a sequence check and write an error message if the values are out of sequence. If the values are in sequence, write an appropriate message.
26. Write a C program to find sum of the diagonal elements of a matrix.
27. Write a C program to get the transpose of a matrix.
28. Write a C program to perform matrix multiplication.
29. Write a C program to find the maximum and minimum values of each column of a given matrix. Write functions to find the maximum and minimum. Each function takes an integer pointer as its parameter. (*Hint: Each column is passed as an array to the function – store each column elements in a temporary array and then pass to the function*)
30. Write a C program to store 10 names in an array and print them each in one line.

CHAPTER 5

INTRODUCTION TO WEB DESIGN

5.1   Introduction

In order to learn Web design and HTML (Hypertext Markup Language), you must first understand how your computer interacts with the Internet. Internet is a network of networks. It has no central control. All the nodes in the network are be equal in status to all other nodes. Each node has the authority to originate, pass and receive messages. On the net, you can find computers. On the World Wide Web (WWW), you can find documents.   The World Wide Web is a collection of documents. The World Wide Web is most often called as Web. On the net, the connections are cables between computers. On the Web, the connections are hypertext links.   The Web exists because of the programs that communicate between computers on the net. The Web cannot exist without the net. The Web made the net useful because people are really interested in information.

The birth date of Hypertext Documents is June 12, 1991. Hypertext documents that are shared on the Internet are called web pages. Web pages are files stored on computers called Web Servers. Web clients are the computers that are requesting the web pages from the web servers. Web clients can view the web pages with a program called web browser. Web pages are created using Hypertext Markup Language (HTML). Hypertext Transfer Protocol (HTTP) is the communication protocol used by the Internet to transfer hypertext documents. A protocol is a rule, which guides how an activity should be performed. More precisely, a protocol is a formal description of message formats and the rules that two computers must follow to exchange those messages. The location address of the hypertext documents (Web pages) is known as a Uniform Resource Locator (URL). Web pages usually contain text, graphics, multimedia, and links to other pages. The HTML and HTTP standards are defined by

284

WWW consortium. HTML is the widely accepted format to create and to viewinformation on the net. The two most commonly used browsers are Microsoft Internet Explorer and Netscape Navigator.

5.2   Elements of Hypertext Markup Language

The Hypertext Markup Language is composed of **tags** that instruct a Web browser how to format and process a hypertext document. Web pages are created using HTML. Web pages can be created by using a simple text editor program such as Notepad or in a Web page editor such as FrontPage. The HTML elements are defined using HTML tags.

Example:

A Web document starts and ends with the following tags.

<html>
………
………
</html>

The HTML tags are always enclosed within angular brackets **<...>**. Each starting tag must have a proper closing tag. In the above example, <html> is the starting tag and </html> is the ending tag. A forward slash (**/**) character is used after the opening angular bracket (followed by the corresponding tag name) to represent the closing tag. The tags are not case sensitive. Lowercase letters are used for tags in this book.

The text between the start and end tags is the element content that is to be manipulated by the tags.

A sample HTML element:

<body>
This is my first HTML document
</body>

285

The HTML element begins with the starting tag <body> and ends with the ending tag </body>. The entire web document is contained within this HTML element.

Actually there are two parts of a Web document.

- Heading Section
- Body Section

The heading section is identified by a pair of head tags (**<head>** and **</head>**) and the body section is identified by a pair of body tags (**<body>** and **</body>**). Comments can be provided any where in a HTML file. The comment can be included using **<!— comment — >**. The comments are ignored by the browser. Comments are used only to improve the readability of the document. The **<head>** tag is optional.

5.3   Heading Section

5.3.1   Title tag

The heading section can contain nested HTML tags. A **<title>** tag is used to provide a name to a web document.

```
<html>
<head>
<title>First Web Document</title>
</head>
</html>
```

The title "First Web Document" is placed in the title bar on the browser as shown below:

286

title

First Web Document – Microsoft Internet Explorer

File   Edit   View   Favorites   Tools   Help

Back   Search   Favorites   Medi

5.3.2   Meta tag

The meta tag is used to provide additional information about the page that is not visible in the browser. The meta tags are always placed within the heading section of the Web page. This tag can be used to identify the author's name of the web document and to identify the keywords that describe the site. Keywords are the group of words that are frequently used in the document and they have to clearly indicate the context of the document. Search engines use these keywords to group the web sites.

Example:

<meta name="Author" content="Albert">
<meta name="keywords" content="books, definitions">

The **name** attribute of the **<meta>** tag is used to identify the user-defined variables and the **content** attribute is used to identify the values of those variables.

In most browsers, user can reload the page by clicking the button Reload or Refresh. In some web documents, the contents are changing periodically. So we have to refresh or to reload the page again to view the new contents. But the **<meta>** tag can be used for automatic reloading of pages at specific intervals. The attribute **http-equiv** is used for this purpose.

287

Example:

<meta http-equiv="refresh" content="30">

The current web page, which is shown on the browser, is automatically reloaded every 30 seconds. It is also possible to redirect to another web page after some specified time using **<meta>** tag.

Example:

 <meta http-equiv="refresh"
    content="5; url=www.yahoo.com">

The above **<meta>** tag redirects to yahoo page after displaying the current Web page for 5 seconds. The content attribute identifies the URL of the Web page and the number of seconds that the browser waits before reloading the Web page. A semicolon is used to separate the waiting time and the URL.

5.3.3   Style tags

The style tags are also used within the heading section. A style tag is used to change the default characteristics of a particular tag in the entire web document wherever that tag is used. A style tag has two segments – a selector and a property.

Example:

    <head>
    <style>
     h2 { color:blue }
    </style>
    </head>

The **<h2>** tag is a heading tag that has a predefined formatting style used for headings within a web page. In the above example, the

default characteristic of the **<h2>** tag has been modified. The browser renders the element content of the **<h2>** tag with bold font and black color. In the above style tag, the **selector** is **h2** and the **property** is color whose value is blue. Hence, wherever **<h2>** tag is used in the body section of the web document, the element content within the **<h2>** tag will be rendered as blue by the browser. But the default color is black. For a selector there may be more attributes and semicolon is used to separate the attributes.

<style>
h2 { color:red;
      font-size:12pt;
       font-family:arial;
   }
</style>

Now the formatting style of **<h2>** tag is very much changed

5.4   Body Section

A pair of body tags **<body>** and **</body>** is used to identify the body section. The body section of a web document can contain many HTML tags. Some tags are used to format a line of text. Some tags are used to insert images, tables and forms and to create hyper links. The most frequently used tags and their attributes are described here.

5.4.1 Body tag

As already mentioned, a body tag is used to identify the body section. We can specify attributes to many of the HTML tags and hence enhancing the usage of those tags. The body tag contains several attributes. To change the background color of a Web page, the attribute **bgcolor** is used.

     <body bgcolor=#FFFFFF>
     </body>

The above code changes the color of the background to white. The attribute **bgcolor** has the value #FFFFFF. The hexadecimal number associated with the color white is #FFFFFF. The color range can be obtained by using the combination of Red, Green and Blue (**RGB** combination). The color values range from 0 to 255 in decimal and 00 to FF in hexadecimal. The RGB combination of white is as follows:

```
FF   FF   FF
R    G    B
```

Similarly for black,

```
00   00   00
 R    G    B
```

The color combination #99BDFF will represent a light blue color and #7FFFD4 is Aquamarine color. Try with different RGB color combinations in your browser.

You can also use the name of the color instead of the corresponding RGB value to indicate some basic colors. For example, "black", "red", "pink", "blue", and "green" are all valid for use in place of RGB values.

<body bgcolor=blue> .. </body>

The background color of the web page can also be modified by using the selector as body and the property as background-color: #RGB (or name of the color) in the <style> tag.

Example:

```
<html>
<head>
<style>
 body {
        background-color: #99BDFF;
      }
</style>
<body>
 ……...
 ……...
</body>
</html>
```

The <body> tag is rendered by the web browser and the background color of the web page is light blue in this case.

You can use the **background** attribute to load a background image on the Web page.

<body background=tnlogo.gif>

Images that are commonly supported by browsers have **.gif** or **.jpg** extension. We can use the **text** attribute to change the color of the text in the entire body section.

<body text=red>

The default text color is black, but it has been changed to the color red for this Web page.

5.4.2   Heading tags

Heading tags in the body section are different from the head tag of the Web document. If it is needed to provide a heading for the Web document, heading tags can be used. There are six heading tags with different font characteristics, <h1>, <h2>, <h3>, <h4>, <h5> and <h6>. The **<h1>** tag specifies the use of first-level heading in a body of text. The closing tag is must for heading tags. Formatting returns to the default after the closing tag.

291

Example:

<html>
<head>
<title>Heading Tags</title>
</head>
<body bgcolor=#99BDFF>
<h1>Computer</h1><p>
<h2>Computer</h2><p>
<h3>Computer</h3><p>
<h4>Computer</h4><p>
<h5>Computer</h5><p>
<h6>Computer</h6><p>
</body>
</html>

In the above example, a paragraph tag **<p>** is used which is explained in the section 10.4.3. See the following screen to understand the various font characteristics of heading tags.
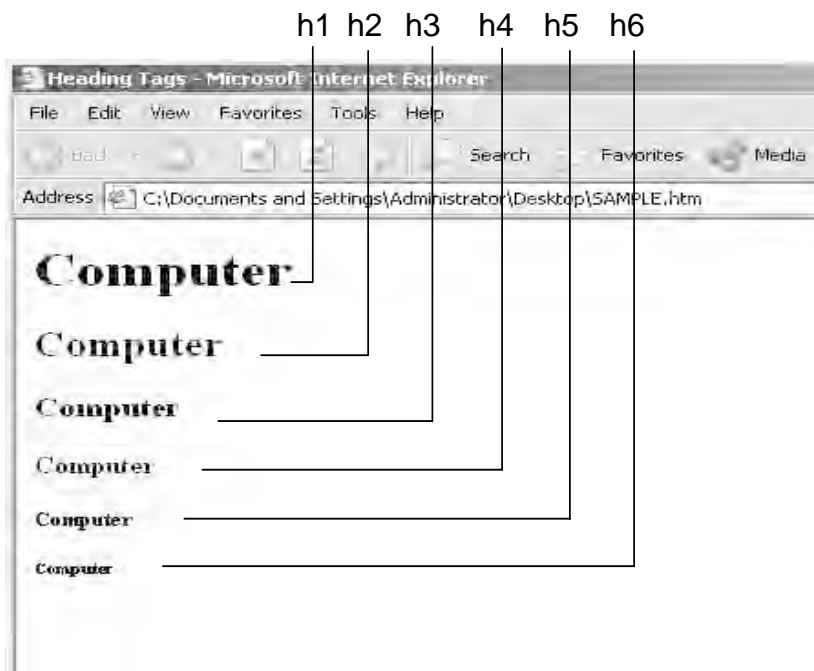
The font size for h1 is very big and for h6 is very small. The h2 is second-level and h3 is third-level and so on. To centre the heading in a Web page, an "align" attribute can be used.

<h1 align=center>Computer</h1>

A style attribute can also be used with some HTML tags to change the characteristics of the tag on which it is applied. If the style is used as an attribute, it will affect only that tag in which it is specified.

Example:

<h2 style="color:pink">This text will be rendered in pink**</h2>**

The element content of h2 tag is now rendered in pink color.

5.4.2 Other HTML tags

The general syntax of any HTML tag is as follows:

<tagname attribute=value>element content</tagname>

The most predominant tags are described below:

Paragraph Tag

The paragraph tag **<p>** defines a paragraph. It starts a new paragraph in a new line. By default, the paragraphs are aligned to the left side of the Web page. The align attribute of the paragraph tag allows you to align the paragraph to right, center or left or to justify it.

<p align=center>Computer</p>

Break Tag

The break tag **
** is used to insert a line break. The break tag need not have any attributes and a corresponding closing tag.The <br> tag is an empty tag and it does not have a closing tag.

Bold, Underline and Italic tags

The bold tag **<b>** formats text in boldface. The tag **<u>** underlines the text and the tag **<i>** italicizes text. These tags must need corresponding closing tags. If you are not providing the closing tags, the effect will continue till the end of the Web page.

Center and Horizontal Ruler tags

The center tag **<center>** is used to center the text, image and the other contents until a closing **</center>** tag is encountered. The horizontal ruler **<hr>** tag inserts a horizontal line.

Font Tag

The **** tag can be used to render the text in specific font type, size and color. In most web browsers, the default font type for an HTML document is Times New Roman. The tag can be used with its face, size and color attributes, to change the font type of characters to be displayed by a web browser.

Example:

****4PT font size
rendered in Arial Type in black color****

The output will be:

4PT font size rendered in Arial Type in black color

The **face** attribute directs the browser to render text in a specified font face or font family. The **size** attribute is used to change the relative size of the font. The **color** attribute specifies the color of the text rendered.

Image Tag

To insert a graphic, an **<img>** tag can be used.   This tag must have an attribute **src**. The **src** stands for "source". The value of the **src** attribute is the URL of the image you want to display on your page. The image tag is an empty tag, ie., it does not have a closing tag.

<img src=tnlogo.gif>

To control the size of an image, the width and height attributes are used.

<img src=tnlogo.gif width=100 height=120>

You can make the image larger or smaller by changing the values in the "width" and "height" attributes of the <img> tag.

Aligning Images within Text

Text messages can be followed by an image. By default, the text messages are aligned at bottom. The following <img> tag

<img src=tnlogo.gif>Government of Tamil Nadu Emblem
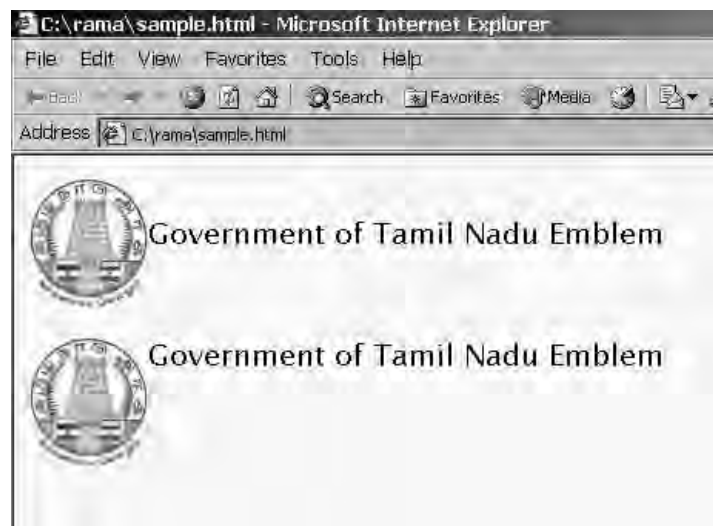
will format the page as follows:

The text after the image is aligned at bottom by default. To align the text in the middle or at the top which follows the image, use the following HTML code and see the output

<img src=tnlogo.gif align=middle>Government of Tamil Nadu Emblem

<p>

 <img src=tnlogo.gif align=top>Government of Tamil Nadu Emblem

If the align attribute of the image is set to "left", the image will go to the left of the text or if the align attribute is set to right, the image will go the right of the text. Try the following code with your browser. Assume any image.

```
<html>
<body>
<p>
<img src ="tnlogo.gif" align ="left" width="100"
height="50">
The align attribute of the image is set to "left". The image will
go to the left of this text. </p>
<br>
<p>
<img src ="tnlogo.gif" align ="right" width="100"
height="50">
The align attribute of the image is set to "right". The image
will go to the right of this text.
</p>
</body>
</html>
```

Anchor Tag

The anchor **<a>** tag is used to create a hyperlink to another document. When the user clicks the element content between **<a>** and **** tags, the browser opens the page identified by the **href** attribute. The **href** attribute indicates the URL for the hyperlink. The **<a>** tag links the user to another location within the same HTML document or to another URL. The following code creates a hyperlink to another document.

Yahoo Home Page

We can make the browser to open the Web page in a new window by specifying the target window. The target window can be identified as **_blank, _top, _self,** or **_parent**.

A hypertext link can consist of text, an image, or a combination of both.

The following code makes you to click an image "tnlogo.gif" which will link you to the home page of Government of Tamil Nadu web site.

```
<body>
To view Government of Tamil Nadu Home Page, click the
Government's logo.
<p>
<a href = "http://www.tn.gov.in" > <img src=tnlogo.gif></a>
</p>
</body>
```

Try the above HTML code in your browser and when you move the mouse cursor over the Government's logo, the cursor changes into hand symbol and if you click on the image, you will see the Government of Tamil Nadu Web page on your browser. To test this, your computer should be connected with the Internet.

It is also possible to create links to different parts of the same Web document. To do that, first we must place a pointer in the document where we want to link to. The name attribute of **<a>** tag is used for this purpose. The pointer looks like

<a name="Department">

Then **<a href="#Department">** tag can be used to link to that part.

For example, you want to have a link from the University section to the Department section on the same Web page. Right before "Department" you need to type **<a name="Department">**. At the University section of your page, add the following link:

<a href="#Department">. The # symbol tells the browser to look for the link within the same document instead of looking for another file.

Bgsound Tag

The bgsound **<bgsound>** tag directs the browser to play a sound file. The audio file should be specified using the **src** attribute. The number of times the audio file to be played can also be specified. The acceptable audio file formats are: **.au, .wav,** and **.mid**.

The code to introduce a background sound in your Web document is:

<bgsound src=music.au loop="infinite">

The loop attribute specifies the number of times the audio file is played. The value "infinite" directs the browser to play the sound indefinitely.

To play a movie in the browser, the <img> tag can be used with **dynsrc** attribute.

<img dynsrc=music.dat width=150 height=150>

The music.dat is a video file. The player width and height are specified as attributes of the <img> tag.

5.4.2 Advanced HTML tags

Lists

There are three kinds of lists in HTML:

- Unordered lists ** …. **
- Ordered lists   ** …. **
- Definition lists   **<dl> …. </dl>**

Unordered Lists

This list starts with an opening list **
** tag and ends the list with a closing list **
** tag. Between the <ul> and , you enter the **
** (list item) tag followed by the individual item. The <li> tag identifies an item in a list. No closing tag is needed. For example:

<ul>
<li> Name
<li> Phone
<li> ID
**
**

In the web browser, the three list items are appearing as follows:

- Name
- Phone
- ID

Ordered Lists

An ordered list is similar to an unordered list, except it uses **<ol>** instead of <ul>:

<ol>
<li>Primary School
<li>Elementary School
<li>High School
**
**

The output will be:

1. Primary School
2. Elementary School
3. High School

Definition Lists

A definition list starts with **<dl>** and ends with **</dl>**. It creates a list with no bullets or numbers. The definition list consists of a definition term **<dt>** tag and a definition-definition **<dd>** tag. The definition is indented below the definition term.

A definition list may be as follows:

<dl>
<dt>Protocol:**</dt>**
<dd>A system of rules and procedures governing
    communication between two devices
</dd><p>
<dt>Pretty Good Privacy:**</dt>**
<dd>It is a program that encrypts files**</dd>**
</dl>

The browser formats the above HTML code and displays like:

Protocol:
    A system of rules and procedures governing communications between two devices

Pretty Good Privacy:
   It is a program that encrypts files

Table Tag

The **<table>** tag is used to create a table on a Web document. The table row tag (**<tr>**) is used to insert a new row in the table. The table header tag (**<th>**) is used to insert a new cell inside a table row to represent the column heading. The table data (**<td>**) tag inserts a new cell inside a table row to represent an entry (value) in the table. The **border** property of the table tag is used to create a border around all the cells in the table. The **bgcolor** property is used to assign a color to the entire table. There can be as many rows and columns as you want and as will fit on the screen. If you want a cell to span more than one column, enclose it
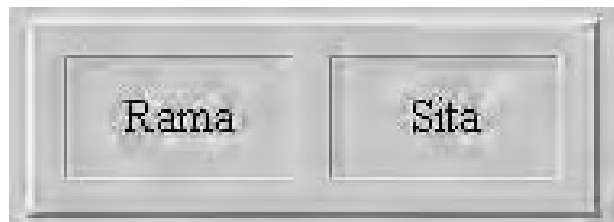
in **<td colspan=x></td>**, where x indicates the number of columns to span. Similarly, **<td rowspan=x> </td>** will cause the cell to span x rows.

The **cellspacing** attribute refers to the space between cells and should be in pixels. The **cellpadding** attribute refers to the spacing within the cell in pixels (the space between the cell walls and the contents of the cell).

Example:

<table border=2 cellspacing=10 cellpadding=10>
<tr>
<td width=50 align=center>Rama</td>
<td width=50 align=center>Sita</td>
</tr>
</table>

The above code will produce the output as follows:
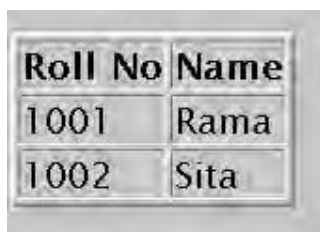
We can represent the table of roll numbers and the names of the candidates in a web document as follows:

<html>
<body>
<table border=2>
<tr><th>Roll No</th><th>Name</th></tr>
<tr><td>1001</td><td>Rama</td></tr>
<tr><td>1002</td><td>Sita</td></tr>
</table>
</body></html>

In the above example, table headers are used to represent the column headings using **<th>** and **</th>** tags. The above document will produce the result as follows:

Roll No	Name
1001	Rama
1002	Sita

Form Tag

Forms are used to receive information from the user. Forms are commonly used to allow users to register on a Web site, to log in to a Web site, to order a product, and to send feedback. In search engines, forms are used to accept the keywords for search. The **<form>** tag is used to create a form. Forms contain many types of form elements, such as text boxes, radio buttons, check boxes, buttons and drop-down lists. The form has a special element, which is **submit** button, which will submit the entries of a form to a server application to process the entries. Each element in the form is assigned a name using the name attribute. Users enter values into the text boxes, or make selections from the radio buttons, check boxes, and drop down lists. The values they enter or select are passed with the name of the corresponding form element to the Web server.

The important attributes used with the <form> tag are **method** and **action** attributes. The method attribute of the form tag is used to identify how the form element names and values will be sent to the server. The **get** method will append the names of the form elements and their values to the URL. The **post** method will send the names and values of the form elements as packets.

The action attribute identifies the server side program or script that will process the form. The action will be the name of a Common Gateway Interface (CGI) program written in programming language called Perl or Java servlets or Active Server Pages.

The general syntax of the **<form>** tag is

<form method=get action="serverscript">

A form element can be created by using an **<input>** tag, which is a form related tag. The **name** attribute is used to name the input element. The **type** attribute identifies the format of the input tag. The possible type attributes are **text, password, hidden, checkbox, submit, reset, file, image** and **button**. The type "text" attribute creates a text box field. The type "hidden" attribute creates a form field that is not visible in the browser. The type "submit" attribute creates a submit button and when user clicks this button, the form elements' names and their corresponding values are sent to the server side program specified in the action attribute of the form tag. The **value** attribute provides a default value for the input tag. The value will be displayed with the form element in the browser.

A sample HTML form is shown below:

<form method=post action="server side program name">
<input type=text name=empname value=rama>
<input type=text name=age value=23>
<input type=submit>
</form>

The above HTML form has two text fields with default values and submitted to the server side program when the user clicks the submit button. The server side program processes the submitted values and sends the results back to the browser or stores the received values on the database.

Frame Tag

Frames divide a web page into sections that each has a different HTML source page and their own set of scroll bars. They can   be useful for any site that requires part of the screen to remain static while

304

the remainder of the screen can be scrolled. One example is site navigation where links can be placed in one frame and the scrolling page content is placed in another. With frames, we can put a number of HTML pages into a single window; each of frames can display a page. Frames are defined using **<frameset>… </frameset>** tags. The <frameset> tag has two modifiers: **rows** and **cols** to define the size of each frame. A Web page with frames should not have body section. The **<body>** tag and **<frameset>** tag cannot come together.

<html>
<frameset rows="64,*">
<frame src="top.html" name="banner" scrolling="no"
    noresize>
<frameset cols="150,*">
<frame src="menu.html" name="contents">
<frame src="home.html" name="main">
</frameset>
</frameset>
</html>

The attribute **rows="64,*"** means that the first frame will take up 64 rows of the window and the second frame will take up the rest. An asterisk means that the row will take up whatever space is left. In the above example, the height of the 64 rows is the height of the 64 pixels put together one by one. We can use percentage to replace length. For example: **rows="30%,60%"**. Actually there are two rows in the web page. The first row (ie., the first frame) is loaded with the file top.html. We can assign a name to each frame using **name** attribute. Naming the frame is useful for future reference The **src** attribute tells which page will be loaded in the frame. The **scrolling** attribute allows us to control the scroll bars on the frame. This attribute has the value **"yes|no|auto"**. The value "yes" forces the frame to have scroll bars always. The value "no" forces the frame to have no scroll bars. The value "auto" allows the browser to decide if scroll bar is necessary. The default value is "auto". The attribute **noresize** does not allow you to resize the frame and it makes the frame fixed.

305

If the files top.html, menu.html and home.html have the following code (each line in one file respectively),

<h2>TOP</h2>   <!—top.html —>

<h2>MENU</h2>   <!—menu.html —>

**<h2>HOME</h2>   <!— home.html —>**

the above frame based web page will be displayed as follows:

Each frame can be further subdivided into rows and columns. The main advantage of HTML frames is that documents can be presented in multiple views, which may be independent windows or subwindows. Multiple views offer web page designers a way to keep certain information visible, while other views are scrolled or replaced.

5.5   Creating Web pages with Microsoft FrontPage

Microsoft FrontPage is the Web authoring program for Microsoft Windows. It is the most widely used Web authoring application.
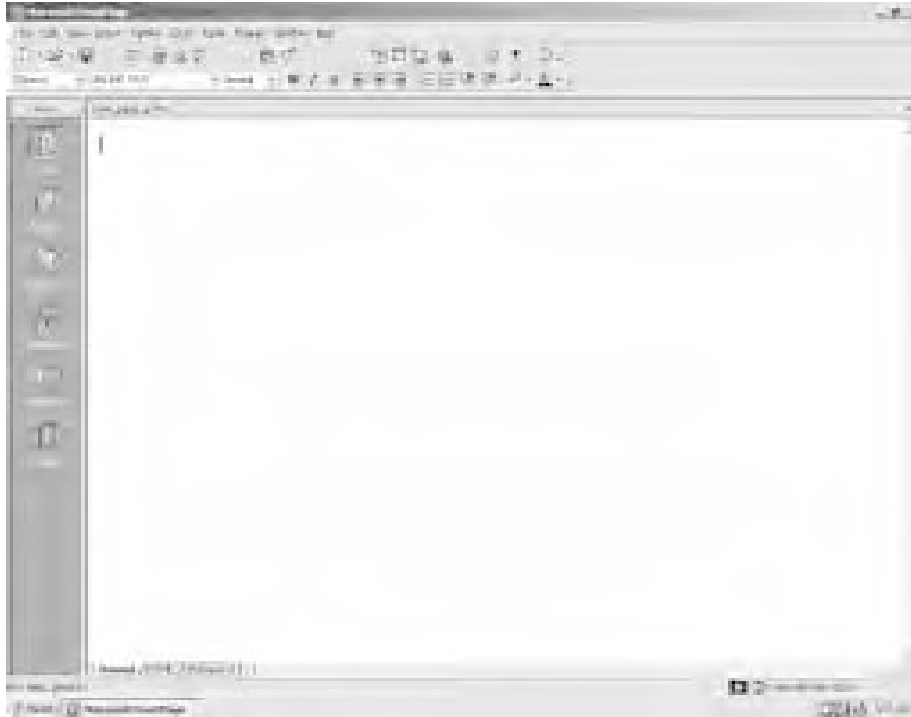
5.5.1 Views

Page view gives you a WYSIWYG (What You See Is What You Get) editing environment for creating and editing web pages. **Folders view** lists all of the files and folders in your Web for easy management. **Reports view** identifies problems with pages and links in the Web. **Navigation view** lists the navigation order of the site and allows you to change the order that a user would view the pages. **Hyperlinks view** allows you to organize the links in the Web pages. **Tasks view** provides a grid for inputting tasks you need to complete in your Web.

5.5.2 Creating a Web Using the Web Wizard

Open FrontPage and select **File|New|Web**... from the menu bar or click the small down arrow next to the New button on the standard toolbar and select Web.
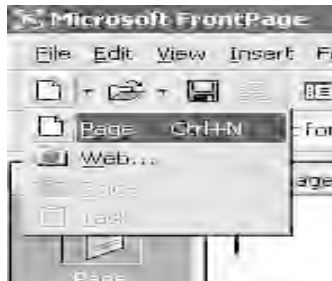
2.   Select the type of Web you want to create. It is usually best to create a simple One Page Web to which you can add additional blank pages, as you need them. Enter a location for the Web in the box provided beginning with "http://". This is the location where you can preview the Web on your computer. It will need to be copied to the server to be viewed to the world on the WWW.
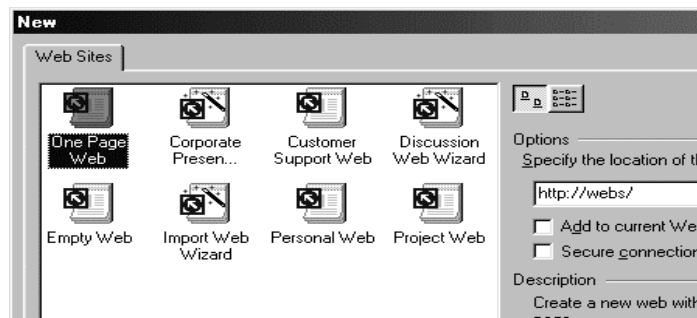
3.   Click OK and wait for FrontPage to finish creating the Web.

Now, explore the created Web. Click Folder view to see the initial page (default.htm) that was created along with two folders. The "images" folder is where you will place all your graphics and photos.
Click on Reports view to see a list of reports for the site. As you con construct your Web, this page will be much more useful. From here, you can identify and correct broken hyperlinks.
View the navigation layout of the Web by clicking Navigation view. Right now, there is only one page listed (the home page). As more pages are added, this page becomes helpful to see how all your pages are linked together.
Hyperlinks view allows you to manage the links on your pages.
Make pages and save them, marking them as completed in the task view. Click Folders view to locate and open the next page to work on.

FrontPage provides many individual page templates that can be added to any Web. To open a Web you have already created, select **File|Open| Web...** from the menu bar. Select the web folder from the list and click Open. You can save all the pages within the Web that was created by the FrontPage. The FrontPage will automatically provide the HTML code for all the pages created by it.

5.5.3   Adding Text to your Web Page

Using text in FrontPage seems to be the same as using text in any other word processor. However, there are a few differences.
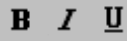
1. You cannot create indented paragraphs
2. The program allows you to select from a series of fonts. However, they may not always display correctly on another computer that does not have that font.

The descriptions of the various text editor buttons are shown below. You must first select the text in order to change the text using these buttons.
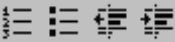
Times New Roman Pull down menu that allows you to select various fonts. Be sure to check these out through your browser.

A A Clicking on the large A, will increase your font size, clicking on the smaller A will decrease your font size.

B *I* <u>U</u> B=Bold, I=Italics, U=Underline

This button allows you to select a color for your font. After clicking on this button you will have the choice of several colors, and by clicking "Define Colors" you will have even more colors.

These tools allow you to select alignment for your text. (left, center, right)

These tools let you select a numbered list, bulleted list, push the text to the left, or push the text to the right.

5.5.4 Creating a Table

On web pages, tables can serve many functions.

A quick way to create a small table is using the table button on the standard toolbar. Click the button and drag the mouse over the grid, highlighting the cells that should appear on the table. When the table size has been selected, click the mouse button again.

309
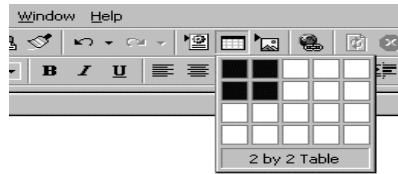
When the selection is made as shown in the figure above, a table with 2 rows and 2 columns will appear on the page as shown below:
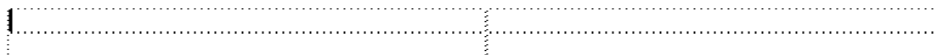
To change the table properties:
- Select Table|Properties|Table from the menu bar to modify the table's properties.
- Select Tables|Properties|Cell from the menu bar or Cell Properties from the shortcut menu to change the properties of   the table cells. Begin by highlighting the cells whose properties will be changed.
- Select Table|Merge Cells to merge two or more selected cells.
- To split the cell again, select the cell and choose Table|Split Cell from the menu bar.

FrontPage makes things easier and it will provide a way to design the Web site in an effective manner. Similarly, it is easy to insert an image and align the same properly in your Web document. Frames and forms are also included using the menu bar commands and tool bar buttons. All you need is a practical experience of using FrontPage to create your own Web site.

I Fill in the blanks
1.	The abbreviation HTTP stands for ———————————— and the abbreviation HTML stands for ————————————
2.	A Web document starts with ———————————— tag and ends with ———————————— tag.
3.	The two parts of a web document are ———————————— section and ———————————— section.
4.	The meta tags are always placed within the ———————————— — section of the Web page.
5.	The attribute ———————————— is used along with <body> tag to change the background color of the web document.
6.	The <body> tag uses ———————————— attribute to load a background image on the web page and ———————————— attribute is used to change the color of the text in the entire body section.
7.	There are ———————————— heading tags with different font characteristics.

8. HTML tables organize data into ——————————————
9. The table data <td> cells are —————————————— aligned by default and the table header <th> cells are ————————————— —— by default.
10. When a style tag is used in the ——————————————, it will change the default characteristics of the tag in the entire web document wherever that tag is used.
11. If the style is used as an attribute, that is, ————————————— style, it will affect only that tag in which it is specified.
12. The tags ——————————————, ————————————— and — ————————————— etc., are not having closing tags.
13. To control the size of an image, the attributes ————————————— —— and ————————————— are used along with <img> tag.
14. The target ————————————— loads the web page in a new blank browser window and the target ————————————— loads the web page in the same window.
15. If <frameset> tag is used, the ————————————— tag cannot appear in the same web document.
16. The tags that do not have corresponding ending tags are called ————————————— tags.
17. Three types of lists that are used to organize the information in the Web pages are —————————————, ————————————— —, and —————————————.
18. The tag ————————————— is used to create links in Web document. Links are also known as —————————————.
19. The popular image formats supported by the Web browsers are ————————————— and —————————————.
20. ————————— view, ————————————— view, and ————————— ————————— view are the three views provided by the Front page editor for a Web page.

II State whether the following statements are True or False

1 The HyperText Markup Language is the standard language used for creating web pages.
2 Both <body> tag and <frameset> tag can be used in a same Web document.
3 The background image in a Web document can be set using <img> tag.
4 Use of <head> tag is must in every Web document.
5 The <style> tag is used only in the heading section.
6 The protocol that Web clients and servers use to communicate with each other is called WWW.

311

7   HTML tags can be enclosed in simple parentheses.
8   All the tags in HTML are having attributes.
9   The table headers are optional in HTML tables.
10   The <meta> tag is the main HTML element that the search
       engines use to group the Web pages.

III Write Answers for the following questions

1.   How do you specify global styles for HTML tags?
2   How do you make a Web page to reload automatically for
       every 10 seconds? What is the need for such reloading?
3   Discuss the attributes associated with the <meta> tag and
       explain their purpose.
4   How do you make an image as a hyperlink?
5   Differentiate between the <style> tag and the style attribute
       used with some other tag.
6   What are the attributes used along with the <font> tag?
7   How do you align images within text?
8   How do you play background music in a Web document?
9   How do you play movie in a Web browser?
10   What are the different types of lists offered by HTML?
11   Write HTML code to create a table of 3 rows and 3 columns
       with appropriate border and show an image in each cell.
12   Design a Web site for you using frames. Include personal
       information page, educational details page, hobbies page and
       your achievements page with appropriate hyperlinks.
13   Discuss the attributes used along with the <frame> tag.
14   How do you create a Web site using Front Page editor?
15   How do you add text, image and table using Front Page editor?