# Chapter 4
# Software

Software provides necessary instructions to the CPU to perform specific tasks. A set of computer instructions is called a program. These programs are permanently stored in ROM chips or a variation of it, flash memory, or in secondary storage media. A program or portion of it at a time will be brought into the CPU accessible memory (RAM or cache) and will be fetched by the control unit, a component of the CPU, one instruction at a time to be executed. One such instruction from a program will be converted to several microinstructions by the CPU, which in turn activates the electronic gates to move electric signals around. There are three major categories of programs, application programs, system programs (Operating Systems) and programming languages.

## Application Programs

Application programs are those that accomplish a specific set of tasks such as word processing, accounting, or architectural plans. There are some application programs that can be used by just about anyone and there are others that can be only used by specialized group of people. Those application programs that are common to most can be integrated such that one can exchange information with another with ease and the menus and shortcuts look very similar between different applications; these integrated programs are called suits. Integrated programs that are commonly used in all kinds of offices are called office suits, the most popular one being the Microsoft Office Suite. We will briefly mention the basic capabilities of each of the modules included in an office suite and discuss in some detail the advanced capabilities of them. If you need to use any of these capabilities, you should refer to a "how to" book that explains specific application development step by step. There are many such books available in the market. An office suite generally will include a document preparation program, a spreadsheet program, a presentation software and even a database program.

A document preparation program can be just an editor, a word processor or a desktop publisher. Editors are included with most operating system and they vary from line editors to page editors. Microsoft Windows are shipped with two editors, WordPad and Notepad. These editors are very useful for creating ASCII files. Word processors and desktop publishers have many features that an editor does not have. Word processors allow us to create, edit, format, print, save and retrieve documents. When word processing programs appeared in the market in the 1980s, they only processed characters not graphics. A desktop publishing software was required to combine and layout text with pictures. It is difficult today to distinguish between word processors and desktop publishing software when used for simple tasks such as a brochure.

Features of word processors include spell check, grammar check, thesaurus look up, table manipulation, formula composer, pagination, header and footer, mail merge, sort, drawing capabilities, tracking changes to a document, comparing documents, password protection, envelop preparation, foreground and background colors, and html document creation to name a few.

Spreadsheet
Presentation Programs
Others

## Operating Systems

Early computers did not have an operating system. Jobs such as finding the appropriate tape, loading the tape, starting the tape drive, instructing the computer to read the program, starting the program, etc. were all performed by human operators. In the early 1950s some control programs were written for interpreting and carrying out batch jobs. In 1964 IBM introduced the OS/360, which was the first operating system that was used by a family of computers.

The operating system is a collection of system programs that manage the resources of a computer including memory, files and peripheral devices, schedule processes for the CPU, and protect programs from each other. The operating system also manages all the input and output devices the use of device drivers provided by the manufacturer of the device. An operating system has two major parts the kernel and the shell. The kernel is concerned with the process control, memory management, secondary storage management, input/output control, file management and networking. The shell interacts with the user, accepts commands from the user, interprets the commands and passes the commands to the kernel. A shell may come in different flavors and the user may choose the preferred flavor. A part of the shell may be programmed (scripts) by the user to carryout certain commands automatically. Many utilities come bundled with an operating system such as format, text edit, search and sort. The DOS a single user operating system and the UNIX, a multi-user, multitasking operating system, will be used as examples in this chapter.

A CPU can simultaneously handle several processes in any given period of time because it is much faster than rest of the computer system. A process is a running program and all the variables and register contents the program is using. Each process needs to have distinguishing program identification and its own process control block (PCB). A process control block contains information about the state of the process and a copy of its program counter, stack pointer, memory limits, open files, and other pertinent information. All computers today execute several processes in parallel. In reality it only executes one process at a time. But, switches from one process to another and makes use of the PCB to store information about each process in order to continue operation on that process later.

Each process is given a time slice determined by a timer. The timer generates an interrupt when the time for current process runs out. The context switch time is the time it takes to switch from executing one process to another. A process is moved from one queue to another by the operating system scheduler. Several CPU scheduling algorithms exist. A priority number may be assigned to each process depending on the user of the process or the size of the process and the process with highest priority will be allocated to the CPU. A low priority process that does not get allocated to the CPU will be assigned a higher priority as it ages. The round robin scheduling or first-come first served scheduling are alternatives to the priority scheduling.

The operating system keeps a number of queues. A ready queue, for example, keeps a list of all PCBs of all processes that are ready and waiting to be executed as a pointer based job queue. The header of the queue will contain pointers to the first and last PCBs. Each PCB will point to the next PCB in the queue.

A process may spawn one or more new child processes as in the fork call in UNIX. A child process may be destroyed automatically or may continue as in independent process when a parent process is destroyed, depending on the operating system. A process is terminated by the operating system when it finishes executing its final statement. A parent may terminate any of its child processes or the user with appropriate rights may request termination of a process. Processes that stay in the background to handle various tasks that need to be carried out periodically are called daemons. A process may communicate or synchronize with another through pipes, signals, interprocess communication or shared memory.

A process can have multiple threads, each handling a different task, giving concurrency within a process. A multithreaded application may allow a portion of a program to continue executing while another part of the program is performing a lengthy operation. Each thread will have a thread-identification, a program counter, a register set, and a stack associated with it. Threads created by a process share the same memory space thereby running a process more efficiently. The creation of a thread is many times faster than creation of a process. In a multiprocessor system different threads can run on different processors.

A program must be brought into the main memory from secondary memory before it can be operated on. When multiple programs are running in a computer there may not be sufficient main memory to keep the entire program and data. A solution to this problem is to only load a portion of the program code under execution into the RAM and to keep the remaining code in a special area of the hard drive called the swap space or virtual memory space. To maximize the efficient use of the RAM without leaving too many unused holes, the RAM is divided into pages or frames having a fixed number of blocks in each. At any given time, only a few pages of a particular process are in the RAM. Pages are swapped between the memory and virtual memory as needed. When the CPU issues a fetch, if that instruction is not loaded into the memory, a page fault occurs and one or more pages are swapped out and needed pages are loaded into the memory. While performing this swapping, the CPU may execute other processes.

## Disk Operating System (DOS)

The Disk Operating System was designed for the Intel 8080 its precursor the Zilog Z-80 microprocessors. Prior to IBM's introduction of the PC in 1981, CP/M-80 by Digital Research was the predominant operating system for the Z-80 based computers. Tim Patterson wrote a very similar operating system and called it 86-DOS which was bought out by the Microsoft Corporation in 1981 and renamed it MS-DOS. The DOS was originally released in 1981 as version 1.0. Several revisions were made to it since then. It was discontinued after version 6.0. Each edition of the DOS is given a version number; the major editions are numbered 1,2,3,4, 5, and 6 and the minor revisions are given decimal numbers. For example, DOS 3.3 means that it has gone through 3 major revisions and 3 minor revisions.

The DOS has three major components: the Basic Input/Output System, the kernel, and the command processor. The Basic Input/Output System (BIOS) is specific for each computer and is provided on ROM chip by the manufacturer. It contains the hardware dependent drivers for the console display and keyboard (CON), line printer (LPT), auxiliary device (AUX), clock, and the boot disk device. Additional drivers may be installed by the user and registered in the config.sys file. The kernel handles file and record management, memory management, character-device input/output, spawning of processes and access to the real time clock. The command processor accepts user inputs, verifies, and passes the inputs to the kernel. The command processor consists of a resident portion, which is loaded into the protected memory, and a transient portion, which is loaded into the unprotected area and could be overwritten by a process and reloaded upon completion of the process.

The boot process is what happens when a PC is turned on. All registers inside the 8086 family of CPU are blanked and execution begins at 0FFFF0H. The BIOS is present at that location in a ROM chip. The BIOS will run a necessary check of all memory for errors, and all connected devices to ensure that everything is present and functioning properly. This is called the Power On Self Test (POST). After completion of these system checks BIOS will search for the Operating System, first in the floppy disk drive. If a disk exists in the floppy disk drive, the PC checks the disk for the boot sector. If a disk is not in the drive, the PC searches for a hard disk from which to boot DOS. If a hard disk does not exist, the PC displays an error message asking the user to insert a system disk. In the newer machines the order of drives in which the BIOS checks for operating system can be changed by the user. The first sector on a bootable floppy disk or hard disk is called the boot sector. The program in the boot sector is read into memory and executed. Next, DOS searches the boot disk for a file named CONFIG.SYS. If found, the commands contained in the file are executed. These commands add device drivers to DOS, allocate disk buffers and file control blocks for DOS and initialise the standard input and output devices. Lastly, the command processor COMMAND.COM is loaded and control is passed to it. The COMMAND.COM processes the AUTOEXEC.BAT file, if one exists. The AUTOEXEC.BAT file is created by the user that contains a list of commands to be executed at startup. The COMMAND.COM can process internal commands that are always loaded into the memory such as COPY and DIR, and external commands that reside on the secondary storage as .COM, .EXE, or .BAT files.

The Intel 8088 CPUs were capable of addressing only 1024KB of memory, and the DOS was designed to take advantage of this amount of memory. The maximum amount of memory the DOS makes available to a program is 640KB. The remaining 384KB is reserved by DOS for the use of BIOS, hardware drivers and video, and is referred to as the upper memory. Even though later CPUs can address many megabytes of memory, when running DOS, it runs in the real mode, and only 1024KB of memory, known as the conventional memory, could be used. The additional memory beyond the 1MB is referred to as extended memory.

## Some DOS Examples

How to format a disk: Most disks are not ready to be used when they are purchased; they need to be formatted. Re-formatting is seldom necessary. Do not re-format the hard disk unless it is absolutely necessary! You can format a floppy disks as described below.

You need to have access to DOS programs to run the format program. Type **format A:** and press return. If it gave an error message (Bad command or file name), then you do not have access to DOS programs. DOS programs are usually kept in DOS sub-directory, if so, type **CD \DOS** and press enter (CD stands for change directory). Then try formatting the disk again.

When a disk is formatted a File Allocation Table (FAT) is created on that disk by the operating system. Fat is a way of organizing the storage space on a disk. The operating system uses the FAT to look up a file and find which clusters that file is written to on the hard disk. FAT is probably the most widely recognized disk format, being read by most operating systems. A 12-bit FAT was first used by Microsoft for managing floppy disks and logical drives smaller than 16 MB. A 16-bit FAT was introduced in MS-DOS version 3.0 for larger drives. The FAT32 supports drives of up to 2 terabytes in size. It uses a smaller cluster size enabling programs to load faster.

How to name a file: A file name has three parts: device, file identification, and an extension. Examples:

A:WORKFILE.TXT

C:FORMAT.EXE

B:MYFILE.PAS

The device name should have colon. The file identification can have up to 8 characters. The extension can have up to three characters and is optional.

The extension is usually used to indicate the type of file (such as text file, executable file, or Pascal file). If you do not specify the device name, then DOS will use the current drive you are using (default drive).

Working with Directories: Upon booting the system the computer uses the root directory. The root directory is indicated by a back slash (\). In order to see the files in the root directory just type DIR and press enter. File names and subdirectory names will be displayed. In order to change the working directory use the command CHDIR (OR CD). For example, to change working directory to WP (Word Perfect) type:

```
CD \WP <PRESS ENTER>
```

A new subdirectory can be created with the use of MKDIR or MD. For example to create a letters directory type:

```
MD \WP\LETTERS
```

In order to see the working directory prompt all the time, use the following command.

```
PROMPT $S$P$G$
```

Creating a file without the use of an editor: Whatever that is typed at the console (keyboard) can be copied to a file using the COPY CON. For example, one wishes to create a small file containing the name, social security number and assignment information, follow these steps:

```
COPY CON ASSNMENT.TXT          PRESS <ENTER>

JAIME GARZA                    PRESS <ENTER>

432-33-4892                    PRESS <ENTER>

CS 1380.3                      PRESS <ENTER>

LAB ASSIGNMENT #3              PRESS <ENTER>

^Z                             PRESS <ENTER>
```

The control Z (^Z) is an end of file marker. When the copy program sees this control character, it stops reading the keyboard and writes whatever that was typed in to a file called ASSNMENT.TXT.

Printing contents of a file: There are different ways by which the contents of a file can be printed. Only text files (ASCII) can be printed; Binary files and graphics cannot be printed using the DOS commands given below. Here are some examples:

| Command format | Comments |
|---|---|
| COPY ASSNMENT.TXT CON | Displays on the screen |
| COPY ASSNMENT.TXT PRN | Sends to the default printer |
| TYPE ASSNMENT.TXT | Displays on the screen |
| TYPE ASSNMENT.TXT >PRN | Sends to the default printer |
| TYPE ASSNMENT.TXT >LPT2: | Sends to printer #2 |
| PRINT ASSNMENT.TXT | Queues to the default printer |

Copying a file: An existing file can be copied to another sub-directory or to same directory using a different file name. To do this, use the copy command. The first file name is the source and the second file name is the destination. Examples:

| Command format | Comments |
|---|---|
| COPY ASSNMENT.TXT C:\PASCAL\ASSNMENT.TXT | The ASSNMENT.TXT in the default directory will be copied to the PASCAL subdirectory in the C: drive |
| COPY SSNMENT.TXT C:\PASCAL | If destination name is same, the name can be omitted. |
| COPY ASSNMENT.TXT FIRST.TXT | Destination name is different. Copied to the default directory |
| COPY ASSNMENT.TXT B: | Copies to a different drive. |
| COPY \DOS\COMMAND.COM A: | Copies Command.com file from the DOS Subdirectory to disk in drive A: |

## Windows Operating Systems

To overcome the memory and multitasking limitations of the DOS and to add Graphical User Interface (GUI), Microsoft announced the Windows in 1983 and released the first version in 1985. The first popular versions were the Windows is 3.1 and the Windows for Workgroups 3.11. The Windows 3.X was not true operating system and it ran on top of the MS-DOS.

The Windows 95 was the first stand-alone Windows operating system. It is a 32-bit operating system and included a built-in peer-to-peer networking. It supported all DOS applications through the DOS prompt. Several DOS windows could be opened up running different programs. The Windows 95 implemented the concept of a Registry to maintain configuration information about all hardware and software controlled by the system. The Windows 95 integrated a 32-bit TCP/IP (Transmission Control

Protocol/Internet Protocol) stack for built-in Internet support, dial-up networking, and new Plug and Play capabilities that made it easy for users to install hardware and software.

The Windows 98 is fixed many of the problems encountered with Windows 95. The registry services were improved. It had better plug and play support for the hardware and included support for the Universal Serial Bus (USB). Many networking and web-browsing features were also added to Windows 98.

The Windows Me was introduced in 2000 and offered enhancements for music and video. The Windows Me was the last Microsoft operating system to be based on the Windows 95 code base. Microsoft announced that all future operating system products would be based on the Windows NT and Windows 2000 kernel.

Windows NT released in 1993 was the operating system targeted at businesses. The NT stands for New Technology. Windows NT was the first Windows operating system to combine support for high-end, client/server business applications with the industry's leading personal productivity applications. It was initially available in both a client version and a server version. It provided enhancements in security, operating system power, performance, desktop scalability, and reliability. New features included a preemptive multitasking scheduler for Windows–based applications, integrated networking, domain server security, OS/2 and POSIX subsystems, support for multiple processor architectures, and the NTFS file system. The last major revision of Windows NT was in 1996.

Windows 2000 Professional replaced the Windows NT and all other previous Windows operating system products. Built on top of the Windows NT Workstation 4.0 code base, Windows 2000 added major improvements in reliability, ease of use, Internet compatibility, and support for mobile computing.

Windows XP and Windows XP Professional, introduced in 2001, merged Microsoft’s two Windows operating system lines for consumers and businesses, uniting them around the Windows 2000 code base. Windows XP Professional enhanced reliability, security, and performance. It includes remote desktop support, an encrypting file system, and system restore and advanced networking features.

## UNIX/LINUX Operating System

The UNIX operating system is an interactive, multiuser operating system. Universities and colleges around the world have contributed in the wide acceptance of this operating system. In particular the University of California at Berkeley made several significant enhancements to the UNIX system. A version of UNIX was distributed as Berkeley Software Distribution (BSD). With the overwhelming acceptance of UNIX it was totally commercialized and the source code was withdrawn from circulating among the colleges and universities. Andrew S. Tanenbaum wrote an UNIX like operating system for teaching purposes for Intel 8088 based machines and called it Minix. The source code for Minix was available to all colleges and universities. Linus Tovalds, a 21-year-old computer science student in Finland started to develop an operating system for Intel-386 based machines following the Minix. He called this operating system Linux. Linus Tovalds invited anyone who is interested to contribute to the code. The Linux has become one of the most sophisticated operating system for personal computers. This operating system and its source code are freely downloadable.

The UNIX consists of the kernel, the shell, and utilities. The kernel handles the process control and resource management. The kernel creates, suspends, terminates, and maintains the states of all processes. It also schedules multiple processes to be executed by the CPU simultaneously and allows inter-process communication through a pipe, a socket, or other communication mechanisms. The kernel also manages memory, files and directories and disks.

The shell sits between the kernel, and interprets user inputs and submits to the kernel for execution. The shell also has a part that can be programmed by the user by the way of shell scripting. There are hundreds of utilities for UNIX and they are expanding each year. As an illustration of the way that the shell and the kernel work together, suppose a user types **`rm letter.txt`** (which has the effect of removing the file **letter.txt**). The shell searches the filestore for the file containing the program **`rm`**, and then requests the kernel, through system calls, to execute the program **`rm`** on **letter.txt**. When the process has finished running, the shell then returns the UNIX prompt to the user, indicating that it is waiting for further commands. In order to try the **rm** example, **letter.txt** must exist. If it does not exist, "**No such file or directory**" will be displayed. This file can be created by typing **touch letter.txt**.

Users communicate with the kernel through commands. A command may include options and arguments. An option modifies the command execution and its output. Options are given after a – or +, as in **ls –a**. Multiple options can be given for each command as in **who –uH**. An argument is a datum that would provide additional information to the command as in **ls letter.txt**. Arguments can provide more information, object identifiers, or names of files. Detailed help on each command is available in the manual pages online. Even help about the manual pages are available with the command **man man**. The most commonly used command and other useful commands are given in the following figures.

**<u>Most frequently used commands</u>**

| **COMMAND** | **TASK DONE** | **EXAMPLE** |
|---|---|---|
| ls | list the files and directories | ls –al |
| mkdir | makes a directory | mkdir unixscript |
| cd | change current working directory | cd unixscript |

| | | |
|---|---|---|
| logout | logs off the system | logout |
| pwd | shows the present working directory | pwd |
| man | shows the help on a specific command | man pwd |
| who | shows who is logged on to the system | who |
| passwd | used to chage the password | passwd |

<u>Other Useful Commands</u>

| COMMAND | TASK DONE | EXAMPLE |
|---|---|---|
| rmdir | removes a directory | rmdir –r dir_name |
| rm | removes files | rm -r file_name |
| cp | copy one file to other | copy sourcefile destinationfile |
| mv | moves files from one directory to another | mv source destination |
| mv | renames files | mv oldname newname |
| more | displays the contents of a file pausing at each screen | more file_name |
| cd.. | change to the parent directory | cd.. |
| lpr | used to print a file | lpr –P printername filename |
| pico | used to edit a file | pico file_name |
| vi | used to edit a file | vi file_name |
| chmod | used for modifying the access permissions | chmod o+r filename |
| cal | displays the calendar | cal 1 2003 |
| date | displays the date | date |
| man –k keyword | used to get the help on a key word | man –k mail |
| pine | used to send and receive mails | pine |
| printers –L | used to list all the printers | printers –L \| more |
| env | used to display the environment variables | env |
| telnet(host) | used to connect to another host | telnet 192.122.0.0 |
| ftp | used to connect to a ftp site | ftp 192.122.0.0 |
| lynx | a textual world wide browser | lynx http://www.cs.panam.edu |
| gopher | a gopher data base browser | gopher http://www.panam.edu |
| grep | searches for a string in a file | grep string file_to_search |
| tail | used to show the last few liens of a file | tail file_name |
| w | to show who is logged on and what they are doing | w |
| bc | a simple calculator | bc |
| finger | shows more information about a user | finger user_name |
| df | shows the amount of disk spae available on the system | df |

| | | |
|---|---|---|
| du | shows how much disk space is been used by the folders | du |
| make | compiles source code | make file_name |
| gcc file.c | compiles C source into a file named a.out | gcc addition.c |
| | | |
| | | |

**UNIX shells**

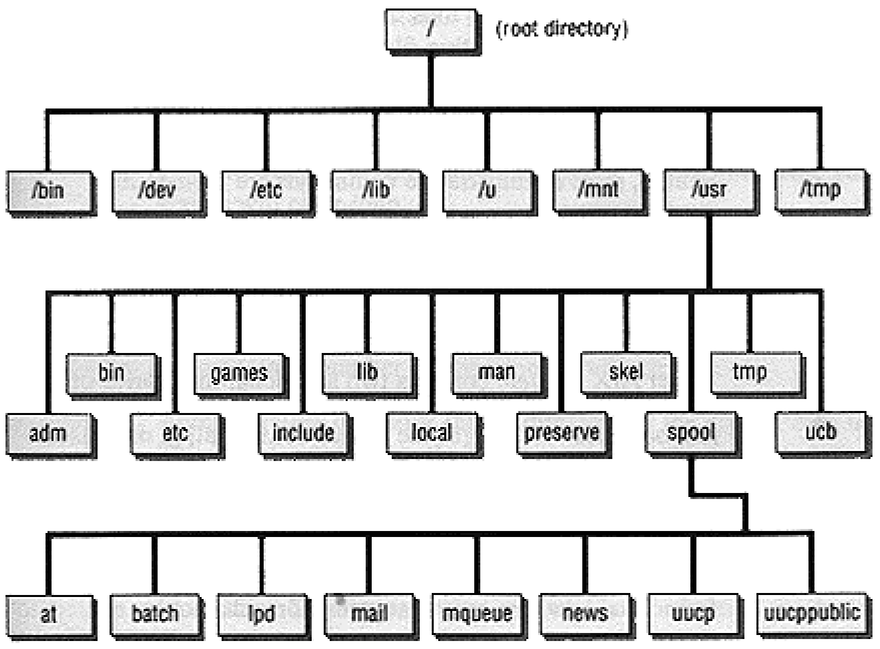
In UNIX, users have their choice of shells and a user may change the shell at any time, though not advisable. Save all configuration file, before changing the shell. Different shells give different OS prompts such as $ for the Korn, Bournre, and Bash shells or % for the C shell. The most popular shells are the Bourne (Sh), Korn (Ksh) and the C (Csh) shell. The Korn shell is a superset of the Bourne shell. Each shell has its own program command set. It is advisable to learn one shell thoroughly and adhere to it. Among the shells there are some speed differences and the newer shells have completion features thereby saving some keystrokes. The Bourne shell, the earliest shell, was written by S. R. Bourne. The University of California at Berkley created the C-shell. This shell has syntax like the C language and is good for interactive use. David Korn incorporated the features of C shell into the Bourne shell and sold it as the Korn shell. About this time the GNU (not UNIX) project was underway and a free shell was written taking the features from all three prior shells and called it the Bourne Again Shell (Bash).

**Booting UNIX systems**

The initial boot procedures are similar to the ones described under DOS. The UNIX can be booted from a local drive (ufsboot) or from a network drive (inetboot). When booting from a local disk the ufsboot blocks are read and executed. Then the ufsboot program is accessed and from the root filesystem. Upon execution of the ufsboot program the kernel programs are read from **/kernel/unix**. At the tail end of the boot process the UNIX operating system runs **ini**t as the process number 1. This process looks for an **initdefault** entry in the **inittab** file. This entry specifies the initial run level either as single-user or multiuser. In the single-user mode is used to check the file system consistency and for other administrative functions. Now, the remaining entries in the **inittab** file are executed. For example, **init 2** will bring the system up in the multiuser mode. The startup script is executed next and generally does the following: set the hostname; set the time zone; run the file system consistency check; mount the system's disk partitions; start the daemons and network services; configure the network interface; and turn on the accounting and quotas. Under the multiuser mode, the **/etc/inittab** file forks the **getty** processes. Each **getty** process displays a login prompt.

**File Systems**

In UNIX, any source from data can be read or written is considered to be a file system. A file may reside on a disk, or the keyboard or a printer could be considered a file. A disk drive is first partitioned with certain number of blocks assigned to each partition. Each of these partitions is then formatted. The format allocates a boot block, a super block, an inode block, and many data blocks to each partition. The boot block is reserved to hold the boot program. The superblock holds the control information for the system including the total size and available blocks. Inodes contain information about each file in the data block. The data blocks hold directories and files. The UNIX file system is arranged in a tree, starting with the root, directories, subdirectories and actual files. A home directory is assigned to each user where a personal profile file is kept. Subsequent directories and files the user creates may cascade down from the home directory. A user may move around in the directories using the **cd** command. The directory a user is working with at any given time is called the working directory. All directories except for the root have a parent directory. When a file is traced from the root it is called a path of that file. It is easy to loose track of the current working directory; the **pwd** (print working directory) will display the position of the working directory from the root. Here is an example how a UNIX directory might be organized.



<u>Editing in UNIX/Linux</u>

Some sort of editor will be required when a user needs to change the contents of a configuration file in order to customize its functionality or write a new program. Any text based line or page editor will suffice. All UNIX/Linux operating systems come standard with one or more editors. A common editor is the full page Vi (*vi*sual) editor. The Vi editor operates in either the command mode or in insert mode. In order to enter text it must be in the insert mode, and it is accomplished by pressing the key i from the command mode. Pressing the ESC key will switch to the command mode. The Vi editor uses a buffer to read the input file and edit it. The original file will not be changed until a write command is issued by the user. A quick guide to Vi is given in figure_________. Other editors that are available for UNIX/Linux are: pico, a text-based editor with commands listed at the bottom of the screen, and emacs, a windows-based editor.

| **LAUNCHING, SAVING AND EXITING Vi (COMMAND MODE)** | |
|---|---|
| **vi std.c** | launches vi editor on file called **std.c** |
| **ZZ or :x or :wq** | Save file and exit |
| **:w** | Save file |
| **:100,200w newfile** | Save lines 100 to 200 in a file called newfile |
| **:q** | Quit |
| **:q!** | Discard changes and quit |
| **Vi –r std.c** | Recover the file **std.c** after a crash |

| **INSERT MODE COMMANDS** | |
|---|---|
| **i, a** | Insert text before cursor; insert text after curser |
| **I, A** | Insert text at the beginning of a line; insert at the end of a line |
| **O, o** | Open a new line above the cursor; open a new line below the cursor |
| **D** | Delete to the end of the line |
| | |

**Writing Shell Scripts**

When multiple commands need to be executed under the shell, these could be placed in a file executed as a script file, instead of running one at a time. Suppose three commands such as **date, ls**, **and who** need to be executed. These three commands could be placed in a file called **example1.sh**. After creating the file, its permissions need to be changed by **chmod a+rx example1.sh**. Any time these three commands need to be executed, the file **example1.sh** can be executed by invoking its name. Like any high level language, the shell scripts can handle variables, input/output functions, arithmetic operations, conditional expressions, selection structures and repetition structures. A script written for the bash should begin with the line #!/bin/bash, indicating that the script should be run in the bash shell regardless of which interactive shell the user has chosen.

Here is an example of a shell script:

```
#!/bin/bash
X=3
Y=4
empty_string=""
if [ $X -lt $Y ]          # is $X less than $Y ?
then
   echo "\$X=${X}, which is greater than \$Y=${Y}"
fi
```

```
if [ -n "$empty_string" ]; then
   echo "empty string is non_empty"
fi

if [ -e "${HOME}/.fvwmrc" ]; then                              # test to see if
~/.fvwmrc exists
   echo "you have a .fvwmrc file"
   if [ -L "${HOME}/.fvwmrc" ]; then                 # is it a symlink ?
                  echo "it's a symbolic link
   elsif [ -f "${HOME}/.fvwmrc" ]; then              # is it a regular file ?
                  echo "it's a regular file"
   fi
else
   echo "you have no .fvwmrc file"
fi
```