FEATURES OF UNIX

1) **Multi user Capability**-

   UNIX provides multi user capability. It allows users to use the resources of main computer. In other words, at a time same computer resources such as hard disk, memory, printer etc. are available to many users.

   The terminals are different types –

   a) **Dumb Terminal** –

      The terminal consists of only a keyboard and monitor is known as dumb terminal. It has no memory or disk so it cannot operate as an independent machine.

   b) Terminal Emulation -

      This terminal consists of keyboard and monitor with its own processor, memory and hard disk drives. The software that makes the terminal is called Terminal Emulation software – PUTTY, XTALK and Vterm.

   c) Dial in Terminal – if the host machine is at remote location then dial in terminal is used. The terminal is connected with host machine via telephone lines.

2) Multi tasking capability

   This feature allows user to carry out more than one job at the same time. Only one job runs in the foreground while the other jobs run in the background. We can switch jobs between background and foreground, suspend or terminate them.

3) **Inter process Communication**- this feature allow user to communicate with another user. It allows passing data, exchange email or program to another user within the network.

4) **Security** – UNIX provides three levels of security-

   a) **System Level Security** – In UNIX, every user has allocated login id and password. When the system administrator opens an account for users, an entry is created in the system password file, called as /etc/passwd. Only authorized users can access the system.

   b) **Directory level security** – in UNIX, everything treated as file even directories or devices are also considered as files. There are read, write and execute (rwx) permission to each file, which decide who can access file, who can modify and who can execute it.

   c) **File Level Security –** In this, there is file encryption. Encryption encodes the file into an unreadable format so that even if someone succeeds in opening file, information will be safe.

5) **Portability** –

   UNIX is written in High level language. It makes it too easy to understand, change and move to other machine

6) **Open system** –

   It has an Open architecture. Modification of the system is easy because the source code is always available and free to download. No license fee is required.

7) **Windowing System** –

Initially, UNIX has a pretty weak interface as it is command driven. Later, UNIX comes up with GUI version. It replaces the command line with screenful objects in the form of menus, dialog box, icons and buttons.

8) **System call and Libraries**
UNIX is written in C. there are many commands available in UNIX that handles special functions called as system calls. These calls are built into the kernel used to communicate with kernel directly.

9) **Programming Facility –**
UNIX is highly programmable. It provides control structure, loops and concept of variable. It provides Shell script and AWK programming.

10) **Networking**
It allows users at one location to log into the system at the other ends and enables user to access the resources of host machine/Server.

ARCHITECTURE OF UNIX

UNIX is a layered Operating system.

There are following major components of UNIX OS.

1] Kernel

The kernel of UNIX is the hub (or core) of the UNIX operating system. Kernel is a set of routines mostly written in C language.

User programs that need to access the hardware (like hard disk or terminal) use the services of the Kernel, which performs the job on the user's behalf.

User interacts with the Kernel by using System calls. Kernel allocates memory and time to programs and handles the file store and communications in response to system calls.

The Kernel is the heart of the UNIX OS. It is a software application that provides the interface between the hardware and the user. It handles the process, memory, file, and device and network management for the operating system. The kernel is responsible for ensuring that all system and user tasks are performed concurrently.

A kernel is a program that:

- Controls all computer operations.
- Coordinates all executing utilities
- Ensures that executing utilities do not interfere with each other or consume all system resources.

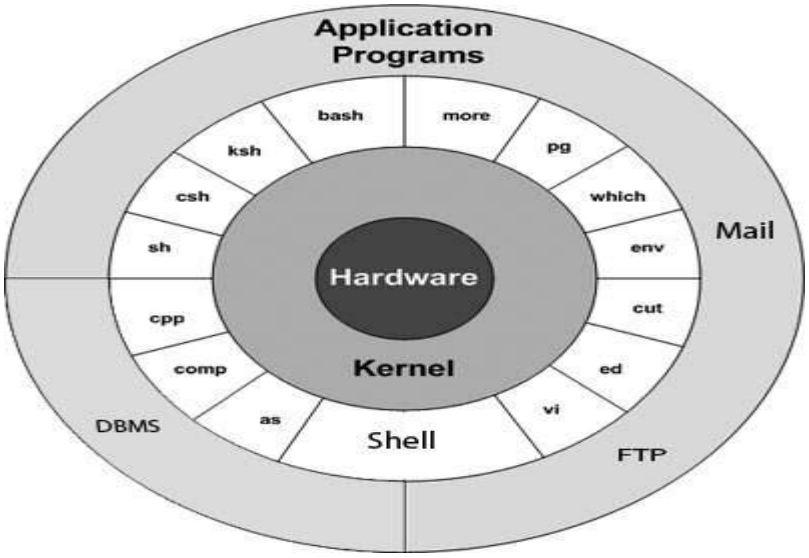
- Schedules and manages all system processes
- Manages files, memory and handle interrupt
- Scheduling of various programs.

2] Shell

The shell is the program that sits between the user and the kernel. It is the interpreter that translates the commands that are typed into the terminal session. Users can type commands directly into the terminal, or they can create a text file containing a series of commands that can be sent to the shell. The series of commands are called a shell script.

There are multiple shells that are used by the UNIX OS. They include the Bourne shell (sh), the C shell (csh), the Korn shell (ksh) and the Bourne Again shell (bash). Each shell has own set of shell commands. Operating system commands are the same across all the shells.

The initial shell that the user logs into is defined by the system administrator. The user can change her default shell by using the "sh" command.



3] Utilities and Application

The final layer of the UNIX OS is the Utilities and Applications layer. This layer includes the commands, word processors, graphic programs and database management programs. Traditionally, these programs were accessed by typing the commands to start the program on the command line. They can still be accessed in this way, but they can now also be accessed through the GUI.

4] Hardware

The innermost layer is a hardware layer that provides service to operating system.

ARCHITECTURE OF KERNEL

The Unix Operating system has three levels:

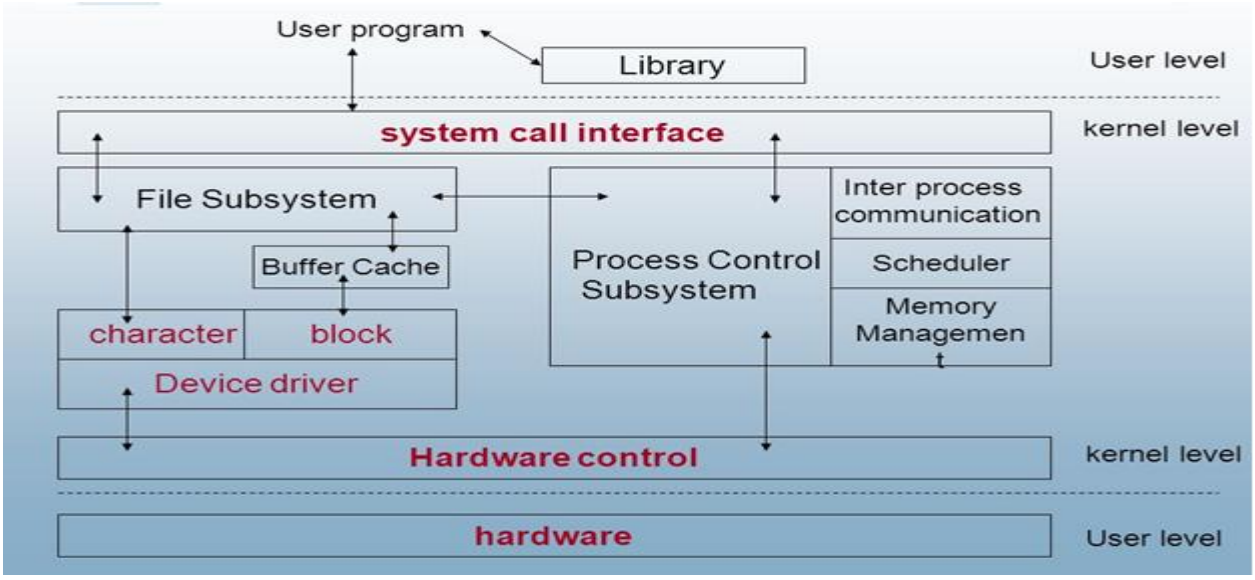
1) **User level**
2) **Kernel level**
3) **Hardware level**

1] **User level:**

The user level includes all the user programs, libraries and the system call interface using which the users interact with the kernel.

User program interact with the kernel through a set of standard system calls. List the system calls used for process management:

System calls	**Description**
fork()	It create a new process
exec()	It execute a new program in a process
wait()	To wait until a created process completes its execution
exit()	To exit from a process execution
signal()	it control the process response to extra ordinary events.
brk()	To increase/decrease the data segment size of a process



2] Kernel Level: The Kernel has two major sub system:

1) **The file subsystem**

   It uses devices drivers to read or write data to or from the disk files. This is done using high speed cache memory which regulates the data flow between the kernel and the secondary storage devices.There are two types of devices files –

   a) **Character Device drivers**

      Character device drivers are used to interact with character devices like terminal. Character devices read or write one character at a time so they do not need buffer cache. An example of character device is terminal.

   b) **Block Device Drives**

      Block device drivers requires high speed memory to transfer data to or from the kernel. Examples of block devices are disk and tape.

2) **Process control subsystem**

   The process control subsystem and the file subsystem interact with each other whenever a process needs an executable file to be loaded into the memory for execution Otherwise the process control subsystem is responsible for inter process communication, CPU scheduling to processes and memory management.

   The **memory management** module controls the allocation of memory to processes. Thus if the kernel finds that the main memory space is not sufficient for the processes, then the kernel moves the processes between the main memory and secondary memory.

   The scheduler module allocates the CPU to processes .this scheduling of the CPU is done in turns, where each processes is given a time quantum. The moment the time quantum expires for any process, the next high priority processes will be run. The previous processes will again get CPU time as and when it becomes next high priority process. **Inter process communication** is implemented using advanced feature of piping between processes.

3] Hardware level

The hardware control consists of module for handling interrupt caused by different devices like disk, tape or terminal.

BOOTING SEQUENCE

1] **ROM diagnostics** are run here hardware and memory tests are performed.

2] **Boot loader** is loaded the ROM BIOS reads the boot loader from boot block and loads it into a RAM.

3] **Kernel** is loaded the kernel program is called by the user recall that the user needs to enter the name of the program. By default UNIX is loaded into the memory by bootstrap program. Now the control is hand over to the kernel.

4] **Kernel initialization** takes place it involves performing memory tests, initializing the devices though device driver, swapper scheduler processes, the init processes and many more.

5] **The init program** resides in/etc directory which is invoked by the kernel. This program takes over the system control. The init program has the PID (process identification) 1 which is the second processes of the system.

The init program reads the instruction of etc/inittab (configuration file used by init program) file to carry out processes like identifying the run level i.e. the mode in which system should run single user/multiuser, maintaining files to track activities etc.

6] **The getty process** the init program also invokes the getty (it is responsible for print the login prompt on the respective terminal and then goes off to sleep) program which establishes a communication with the terminals of the Unix system.

The getty program uses the file called /etc/gettydefs for instruction to provide the login prompt at each terminals connected to system and goes into suspended (sleep) mode and activated when user attempt to login.

7] **The login program** once the user types login name and password ,getty transfer control to a login program, to verify the login name and password entered by user. Thus if the login is successful the $ prompt appears on the screen.

It is essential to know that it is the /etc/profile that enables the system administration to provide information like time and date, system name, number of users etc. This file can be modified by the system administrator to accommodate relevant message for the users**.**

- **Structure of /etc/passwd**

All user information except the encrypted password is stored in /etc/passwd. The encrypted password is stored in /etc/shadow. The /etc/passwd shows the entry of all registered users. Each entries consisting of seven filed. Each filed is separated by colon (:)

Username: password: UID: GID: comment: home directory : login shell

Significance all of seven fields is as follow:

1) Username: it is login name of the user.
2) Password: encrypted password of the user, but it contains an x. The encrypted password is stored in /etc/shadow file.
3) UID: It stands for user identification number. It is unique number; no two users have the same UID
4) GID: It stands for group identification number. It is also unique number.
5) Comment: this filed contains detailed information about the user such as full name of the user, address etc.
6) Home directory: it is the home directory of a user. When user login to the unix system, initially he is placed in this directory.
7) Login shell: when the user logs in then default shell is allocated. Using this shell user communication with kernel.

- **Structure of /etc/shadow**

The /etc/shadow stores actual password of the user in encrypted format. The general structure of /etc/shadow file is shown. Each entry of shadow file is separated by colon (:

Username: pwd: Last pwd change: Minimum: Maximum: warn : inactive: expire: reserve

1) User Name: it is the login name of the user.
2) Pwd: it contain encrypted password. The password should be minimum 6-8 character long including special character/digits.
3) Last password change: days since Jan 1, 1970 that password was last charged.
4) Minimum: the number odd days left before the user is allocated to change password or minimum days before password may be changed.

5) Maximum: the maximum number of days the password is valid or days after which password must be changed.
6) Warn: the number of days before password is to be expire the user is warned that his/her password must be changed or days before password is to expire that user are warned.
7) Inactive: the number of days after password expires that account is disabled.
8) Expire: days since Jan 1, 1970 that account is disabled i.e. an absolute date specifying when the login may no longer be used.
9) Reserve: it is a reserved filed.

FILE SYSTEM

Unix file system is a logical method of **organizing and storing** large amounts of information in a way that makes it easy to manage. A file is a smallest unit in which the information is stored. Unix file system has several important features. All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the file system.

Files in Unix System are organized into multi-level hierarchy structure known as a directory tree. At the very top of the file system is a directory called “root” which is represented by a “/”. All other files are “descendants” of root

Hierarchical File Structure

- All of the files in the UNIX file system are organized into a multi-leveled hierarchy called a directory tree.
- A family tree is an example of a hierarchical structure that represents how the UNIX file system is organized. The UNIX file system might also be envisioned as an inverted tree or the root system of plant.
- At the very top of the file system is single directory called "root" which is represented by a / (slash). All other files are "descendents" of root.

Directories or Files and their description –

- **/ :** The slash / character alone denotes the root of the file system tree.
- **/bin :** Stands for “binaries” and contains certain fundamental utilities, such as ls or cp, which are generally needed by all users.
- **/boot:** Contains all the files that are required for successful booting process.
- **/dev:** Stands for “devices”. Contains file representations of peripheral devices and pseudo-devices.
- **/etc:** Contains system-wide configuration files and system databases. Originally also contained “dangerous maintenance utilities” such as init, but these have

typically been moved to /sbin or elsewhere.

- **/home:** Contains the home directories for the users.
- **/lib:** Contains system libraries, and some critical files such as kernel modules or device drivers.
- **/media:** Default mount point for removable devices, such as USB sticks, media players, etc.
- **/mnt :** Stands for "mount". Contains file system mount points. These are used, for example, if the system uses multiple hard disks or hard disk partitions. It is also often used for remote (network) file systems, CD-ROM/DVD drives, and so on.
- **/proc :** procfs virtual file system showing information about processes as files.
- **/root :** The home directory for the super user "root" – that is, the system administrator. This account's home directory is usually on the initial file system, and hence not in /home in case specific maintenance needs to be performed, during which other file systems are not available. Such a case could occur, for example, if a hard disk drive suffers physical failures and cannot be properly mounted.
- **/tmp :** A place for temporary files. Many systems clear this directory upon startup; it might have tmpfs mounted atop it, in which case its contents do not survive a reboot, or it might be explicitly cleared by a startup script at boot time.
- **/usr :** Originally the directory holding user home directories, its use has changed. It now holds executables, libraries, and shared resources that are not system critical, like the X Window System, KDE, Perl, etc.
- **/usr/bin :** This directory stores all binary programs distributed with the operating system not residing in /bin, /sbin or (rarely) /etc.
- **/usr/include:** Stores the development headers used throughout the system. Header files are mostly used by the **#include** directive in C/C++ programming language.
- **/usr/lib:** Stores the required libraries and data files for programs stored within /usr or elsewhere.
- **/var :** A short for "variable." A place for files that may change often – especially in size, for example e-mail sent to users on the system, or process-ID lock files.
- **/var/log:** Contains system log files.
- **/var/mail:** The place where all the incoming mails are stored. Users (other than root) can access their own mail only. Often, this directory is a symbolic link to /var/spool/mail.
- **/var/spool :** Spool directory. Contains print jobs, mail spools and other queued tasks.
- **/var/tmp :** A place for temporary files which should be preserved between system reboots.

FILE SYSTEM COMPONENTS :-

When you create a file system on a disk it is organized into a sequence of logical blocks. On most system, the smallest block that can be read and written by the disk controller is 512 bytes. This block is called physical block. The kernel reads and writes data using different block size. This block is referred to as logical block. The logical block size varies

from 1024 to 8192 bytes.

All the blocks belonging to the file system are logically divided into four parts boot block, super block, i-node block and data block.

1] Boot block/Block 0:

Boot block is smallest and first part of Unix file system. This block is normally reserved for booting producers. It is also called master boot record (MBR). The bootstrapping program and partition table are stored on the boot block of the root (track zero) file system. When the system is booted, the system BIOS checks for the existence of the hard disk and loads the entire segment of the boot block into main memory. Now the control hands over to the bootstrapping. The bootstrapping program is responsible to load the kernel into main memory. The kernel of a Unix operating system is usually stored in root directory of file system.

2] Super block:

The super block or block1 is the 'balance sheet' of every Unix file system. It stores the characteristics of a file system. It mainly contains-

- The size of the file system.
- The size of logical block of the file system.
- Last time of updating.
- Free data block such as the number of free data block available, list of free data blocks, pointer to first free data block on free data list.
- Free i-node such as the number of free i-node available, list of free i-nodes, pointer to first free i node list.
- The state of the file system whether the file system is 'clear' or 'dirty'.

3] Inode block:

All entries in Unix are treated as files. I-node block stores information about the files and the location of the data block where the content of files is actually stored. The information related to all these files is stored in I-node table on the disk. For each file, there is an I-node entry in the table. Each entry is made up of 64 bytes and contains the relevant details for that file.

- File type: regular, directory, devices etc.
- Number of links i.e. the number of aliases of the file.
- Numeric UID of the owner.
- Numeric GUID of the owner.
- File access permission : read, write, execute for owner, group and others
- Size of the file.
- Date and time of last modification of file data.
- Date and tine of last access of file data.
- Date and time of last change of the I-node.
- Table of content: disk addresses of disk block containing the file data.

The Inode is accessed by a number called the I-node number. An index node is a block that holds pointer to the entire block used by a file. This number is unique for every file in a single file system. As with super block, the kernel also maintains a copy of the I-node block in the memory. Therefore when a file is opened its I-node is copied from the hard disk to the system's i-node table which is maintained in memory. The kernel always works with the memory copy, but it periodically updates the disk copy with the content of the memory copy.

4] Data block:

The remaining space of the file system is taken up by data block or storage blocks. Data block is largest part among other part. These blocks store the content of the file in the case of a regular file. For a directory, each data block contains 16 bytes entries. Each such entry would have a file or subdirectory name up to 14 bytes, and 2 bytes would be taken for i-node. Thus for directories the list of filename and i-node are available.

FEATURES OF SHELL

The primary advantages of interfacing to the system through a shell are:

- **Wildcard substitution in file names (pattern matching)**

  Carries out commands on a group of files by specifying a pattern to match, rather than an actual file name.

- **Background processing**

Sets up lengthy tasks to run in the background, freeing the terminal for concurrent interactive processing.

- **Command aliasing**

Gives an alias name to a command or phrase. When the shell encounters an alias on the

command line or in a shell script, it substitutes the text to which the alias refers.

- **Command history**

Records the commands you enter in a history file. You can use this file to easily access, modify, and reissue any listed command.

- **File name substitution**

Automatically produces a list of file names on a command line using pattern-matching characters.

- **Input and output redirection**

Redirects input away from the keyboard and redirects output to a file or device other than the terminal. For example, input to a program can be provided from a file and redirected to the printer or to another file

- **Piping**

Links any number of commands together to form a complex program. The standard output of one program becomes the standard input of the next.

- **Shell variable substitution**

Stores data in user-defined variables and predefined shell variables.

TYPES OF SHELL

What are the different Shells?

1. The Bourne Shell

The Bourne shell (sh), written by Steve Bourne at AT&T Bell Labs, is the original UNIX shell. It is the preferred shell for shell programming because of its compactness and speed. A Bourne shell drawback is that it lacks features for interactive use, such as the ability to recall previous commands (history). The Bourne shell also lacks built-in arithmetic and logical expression handling.
The Bourne shell is the Solaris OS default shell. It is the standard shell for Solaris system administration scripts. For the Bourne shell the:

- Command full-path name is **/bin/sh** and **/sbin/sh**.
- Non-root user default prompt is **$**.
- Root user default prompt is **#**.

2. The C Shell (csh):

- It is a UNIX enhancement written by **Bill Joy** at the University of California at

Berkeley.

- Incorporated features for interactive use, such as **aliases** and **command history**.
- Includes convenient programming features, such as **built-in arithmetic** and C**-like expression syntax**.

For the C shell the:

- Command full-path name is **/bin/csh**.
- Non-root user default prompt is hostname **%**.
- Root user default prompt is hostname **#**.

3. The Korn Shell(ksh):

- It was written by **David Korn** at AT&T Bell Labs
- Is a superset of the Bourne shell
- Supports everything in the Bourne shell.
- Has an interactive feature comparable to those in the C shell.
- Includes convenient programming features like **built-in arithmetic** and **C-like arrays**, **functions**, and **string-manipulation facilities**.
- Is faster than the C shell.
- Runs scripts written for the Bourne shell.

For the Korn shell the:

- Command full-path name is **/bin/ksh**.
- Non-root user default prompt is **$**.
- Root user default prompt is **#**.

4. The GNU Bourne-Again Shell (bash):

- Is compatible to the Bourne shell.
- Incorporates useful features from the Korn and C shells.
- Has arrow keys that are automatically mapped for command recall and editing.
- Command full-path name is **/bin/bash**.

TYPES OF FILE

File Types:

The UNIX file system contains several different types of files:

- **Ordinary Files**
  - Used to store your information, such as some text you have written or an image you have drawn. This is the type of file that you usually work with.
  - Always located within/under a directory file
  - Do not contain other files
- **Directories**

  A directory file contains an entry for every file and subdirectory that it houses. If you have 10 files in a directory, there will be 10 entries in the directory. Each entry has two components.
  (1) The Filename

(2) A unique identification number for the file or directory (called the inode number)

- Branching points in the hierarchical tree
- Used to organize groups of files
- May contain ordinary files, special files or other directories
- Never contain "real" information which you would work with (such as text). Basically, just used for organizing files.
- All files are descendants of the root directory, (named /) located at the top of the tree.

- In long-format output of ls –l , this type of file is specified by the “d” symbol.
- **Special Files**

  - Used to represent a real physical device such as a printer, tape drive or terminal, used for Input/Ouput (I/O) operations
  - Unix considers any device attached to the system to be a file - including your terminal:
  - By default, a command treats your terminal as the standard input file (stdin) from which to read its input
  - Your terminal is also treated as the standard output file (stdout) to which a command's output is sent
  - Stdin and stdout will be discussed in more detail later
  - Two types of I/O: character and block devices files
  - When a character special file is used for device Input/Output(I/O), data is transferred one character at a time. This type of access is called raw device access.
  - When a block special file is used for device Input/Output(I/O), data is transferred in large fixed-size blocks. This type of access is called block device access.
  - In long-format output of ls -l, character special files are marked by the “c” symbol.
    - In long-format output of ls -l, block special files are marked by the “b” symbol.
    - Usually only found under directories named /dev
    - Each disk device is given its own ***major device number***, and each partition has an associated ***minor device number*** which the device driver uses to access the raw file system.
    - The major/minor device number combination serves as a handle into the device switch table. That is, the major number acts as an index, and the minor number is passed as an argument to the driver routines so that they can recognize the specific instance of a device

- **Pipes**

  - UNIX allows you to link commands together using a pipe. The pipe acts a temporary file which only exists to hold data from one command until it is read by another

- For example, to pipe the output from one command into another command:
  who | wc -l

  This command will tell you how many users are currently logged into the system. The standard output from the who command is a list of all the users currently logged into the system. This output is piped into the wc command as its standard input. Used with the -l option this command counts the numbers of lines in the standard input and displays the result on its standard output - your terminal.

  In long-format output of ls –l , named pipes are marked by the "p" symbol.

5. Sockets – A Unix socket (or Inter-process communication socket) is a special file which allows for advanced inter-process communication. A Unix Socket is used in a client-server application framework. In essence, it is a stream of data, very similar to network stream (and network sockets), but all the transactions are local to the filesystem.

In long-format output of ls -l, Unix sockets are marked by "s" symbol.

6. Symbolic Link – Symbolic link is used for referencing some other file of the file system.Symbolic link is also known as Soft link. It contains a text form of the path to the file it references. To an end user, symbolic link will appear to have its own name, but when you try reading or writing data to this file, it will instead reference these operations to the file it points to If we delete the soft link itself , the data file would still be there.If we delete the source file or move it to a different location, symbolic file will not function properly.

In long-format output of ls –l , Symbolic link are marked by the "l" symbol (that's a lower case L).

Hidden Files: have names that begin with a dot (.) For example:

.cshrc .login .mailrc .mwmrc

- Reserved Filenames:

/ - the root directory (slash)

. - current directory (period)

.. - parent directory (double period)

~ - your home directory (tilde)

Pathnames

- Specify where a file is located in the hierarchically organized file system
- Must know how to use pathnames to navigate the UNIX file system
- Absolute Pathname: tells how to reach a file beginning from the root; always begins with / (slash). For example:

  /usr/local/doc/training/sample.f

- Relative Pathname: tells how to reach a file from the directory you are currently in (current or working directory); never begins with / (slash). For example:

  training/sample.f
  ../bin
  ~/projects/report.001

  For example, if your current directory is /usr/home/johnson and you wanted to change to the directory /usr/home/quattro, you could use either of these commands:

cd ../quattro ***- relative pathname***
cd /usr/home/quattro ***- absolute pathname***

FUNCTIONS OF KERNAL

Kernel is the main component of most operating system. It provides an interface between application and actual data processing the hardware level.

Kernel is considered as heart of an operating system. It provides the lowest level abstraction layer for the resources that application must control to perform its function.

1. Resource Management/Allocation:

The kernel primary function is to manage the computers resource and allow program to run and use this resources. These resources are CPU, memory and IO device.

2. Process Management:

A process defines which memory portions the application can access. The main task of kernel is to allow the execution of application and support them with features such as hardware abstraction. To run an application, kernel first set an address space for the application, then loads the files containing the applications code into memory, then set up stack for the program and branches to given location inside a program. Thus, finally starting its execution.

3. Memory Management:

The kernel has full access to the systems memory. It allows processes to safely access this memory as they require it. Virtual addressing helps kernel to create virtual partitions of memory in to disjoint areas, one is receive for the kernel (kernel space) and the other for the application (user space).

4. I/O device Management:

To perform useful functions, process needs access to the peripherals connected to the computer which are controlled by the kernel through Device Drivers. A device driver is a computer program that enables the operating system to interact with a hardware device. It provides the operating system with information of how to control and communicate with a certain piece of hardware. A kernel maintains a list of available devices. A device manager first performs a scan on different hardware, buses, such as peripheral components interconnect (PCI) or universal serial bus (USB) to detect installed hardware device, then searches for the appropriate drives. To the kernel provides the I/O to allow drivers to physically access their device through some port or memory location.

5. Inter-Process Communication(IPC)

Kernel provides methods for Synchronization and Communication between processes called IPC. There are various approaches of IPC such as semaphore, shared memory, message queue, pipe (or named FIFO) etc.

6. Scheduling :

In multi tasking system, the kernel will give every program a slice of time and switch from process to process so quickly that it will appear to the user as if these processes were being executed simultaneously. The kernel uses scheduling algorithms to determine which process is running next and how much time it will be given. The algorithm sets priority among the processes.

7. System calls and Interrupt Handling:

A system call is a mechanism that is used by the application program to request a service from the operating system. System call includes close, open, read, write and wait. To access the service provided by the kernel we need to invoke the related kernel

functions. Most kernels provide a C library or API, which in turn invoke the related kernel functions.

There are few methods by which the respective kernel function can be invoked using software simulated interrupt or using a gate call or by using special system call Instruction and by using a memory based queue.

8. Security or Protection Management:

Kernel also provides protection from faults (error control) and from malicious behaviour (security) one approach towards this can be language based protection system, in which the kernel will only allow code to execute which has been produced by a trusted language compiler.