UNIT-1

UNIX

Unix (officially trademarked as UNIX, sometimes also written as Unix with small caps) is a computer operating system originally **developed in 1969** by a group of **AT&T** employees **at Bell Labs**, including Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, and Joe Ossanna.

History of UNIX

In late 1960's at AT&T Bell Laboratories the developers were working on the project known as MULTICS (**Mult**iplexed **I**nformation and **C**omputing **S**ervice).MULTICS was an operating system designed on GE-645 mainframe computers. The purpose of developing the operating was to have an operating system that can be portable and can support multiuser. It was developed with multiuser capability still it was considered to be inadequate, the project was dropped but the researchers working the system, Ken Thompson, Dennis Ritchie, M. D. McIlroy, and J. F. Ossanna, decided to redo the work on a much smaller scale. Ken Thompson still had access to the Multics environment; he wrote simulations for the new file and paging system on it. A team of Bell Labs researchers led by Thompson and Ritchie, including Rudd Canaday, developed a hierarchical file system, the notions of computer processes and device files, a command-line interpreter, and some small utility programs.

In the 1970s Brian Kernighan coined the project name *Unics* (**Un**iplexed**I**nformation and **C**omputing **S**ervice) as a play on *Multics*. Unics could eventually support multiple simultaneous users, and it was renamed *UNIX*. UNIX code was written in the assembly language of PDP-7 and was, consequently machine dependent. Ritchie and Thompson worked quietly on UNIX for several years. Ken Thompson then developed a new programming "B" which was subsequently modified by Dennis Ritchie and renamed the "C" language. The whole UNIX system was rewritten in "C" language by 1973 which made is portable language.

The AT&T Company distributed the UNIX to academic and research institutions at a nominal fee. The University of California, Berkeley (UCB), created a UNIX of its own. It was called BDS UNIX (Berkeley Software Distribution). These versions became quite popular worldwide, especially in university and engineering circles.

Berkeleyrewrites the whole operating system in the way they wanted. They created the Standard editor of the UNIX system (vi) and a popular shell(C shell). They created a better file system and they also offered with their standard distribution networkingprotocolsoftware (TCP/IP) that made the Internet possible.

Some other Organizations also developed the other versions of UNIX as given in table.

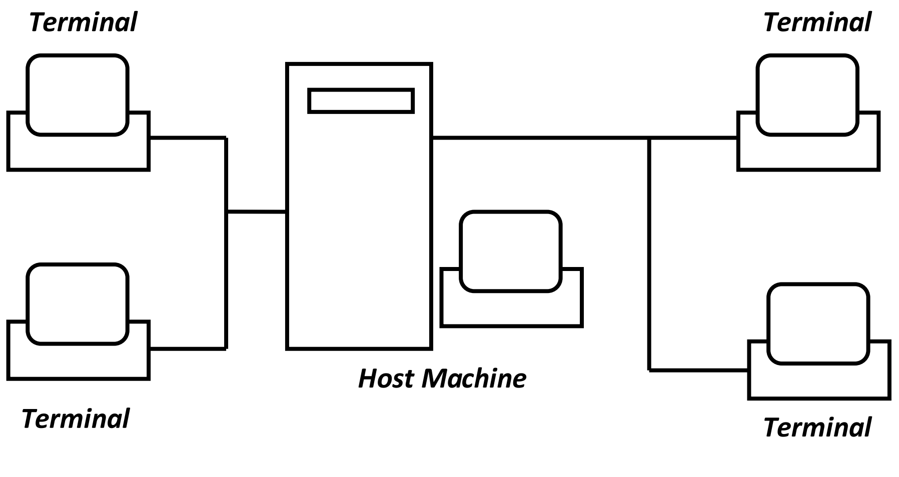
Organization	**Version of UNIX**
Sun Systems	Solaris
IBM	AIX
HP	HP-UNIX
DEC	Digital UNIX now Tru64
MIT	Xwindow

etc.

Features of UNIX

UNIX provides many features as given below:

- **Multiuser Capability:**
  UNIX is a multiuser system. In a multiuser system the **same computers resources**-hard disk, memory etc. ***are accessible*** to many users. All the users are given different terminal. All the **terminals are connected to the main computer** whose resources are availed by all users. So, a user at any of the terminals can use **not only** the computer, **but also** any peripherals that may be attached. e.g.: Printer. The following figure depicts the multiuser system.



The number of terminals connected to the host machine depends on the number of ports that are present in its controller card. There are several types of terminals that can be connected to the host machine.

- **Dumb Terminals:** These terminals consist of the keyboard and display unit with no memory or disk of its own. These can never acts independent machines If they are to be used they have to be connected to the host machine.
- **Terminal Emulation/Smart Terminal:**PC has its own microprocessor, memory & storage device by attaching this PC to the host machine through a cable & running s/w from this machine, we can emulate to work as it is dumb terminal. However memory & disk are not in use & machine can’t carry out any processes on its own.
- **Dial-In terminals:**
  These terminals use telephone lines to connect with host machine. To communicate over telephone line, it is necessary *to attach* a unit called modem to the terminal as well as to the host machine.

➢ **Multitasking Capability:**
UNIX is capable of carrying out multiple jobs at the same time,i.e. a single user can type a program in its editor as well as simultaneously executes some other command you might have given earlier. The latter job is performed in the background while in the foreground he uses the editor. This is managed by dividing the CPU time intelligently between all processes being carried out. Depending upon the priority the operating system **allots** small ***time slots*** to all the process running in foreground and background. MS-DOS also provides multitasking facility which is known as serial Multiprocessing. In DOS time slicing is done i.e. at a time only one job will run rest of the jobs are stopped temporarily. Where as inUNIXtime slicing is not given, in UNIX***priorities*** are **given** and the **processes that have same priority** are ***scheduled*** on a **round-robin base.**

➢ **Inter Process Communication:**
UNIX provides excellent feature that allows **users to communicate with fellow users**. The communication can be within the network of a single main computer or between two or more networks. The users can **easily exchange mail, dataand programs** through such networks.

➢ **Security:**

UNIX provides outstanding facility for security. It provides security in 3 levels.

- The first level security is provided by providing passwords (also called **Login Level** Security). It can be considered as a **System Level Security**. Username and **passwords are assigned** to all the users to assure that no other user can access his work.
- The second level security is at the **file level**. There are read, write and execute permissions to each file that decide who can access a particular file, who can modify it and who can execute it.
- The third level security is given by **file encryption**. This utility encodes user's file into an unreadable format, so that even if the user succeeds in opening the file he will not be able to read it. When user wants to see the contents of the file he can decrypt the file.

➢ **System Portability:**
UNIX is written in High level language "C", hence it can **easily run on any machine** with or without small changes. The code can be changed and compiled on a new machine.

➢ **Open System:**
UNIX has an **open architecture** one can add to the toolkit by simply writing a program & Starting that executable in a separate area in the file. A separate device can also be added by creating file for it. Modification of system is easy because the **source code** is ***always available.***

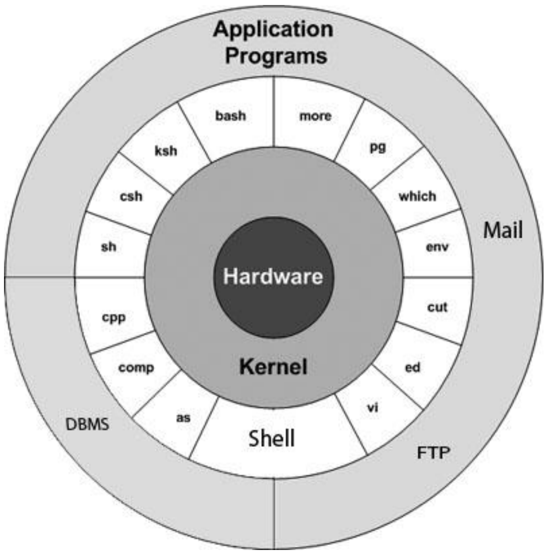
➢ **Programming facility:**
UNIX shell is also a programming language; it is **designed for a programmer**, not a casual end user. It has all the necessary ingredients, like **control structures loops** and **variables** that establish it as a powerful programming language in its own right. These features are used to design shell scripts.

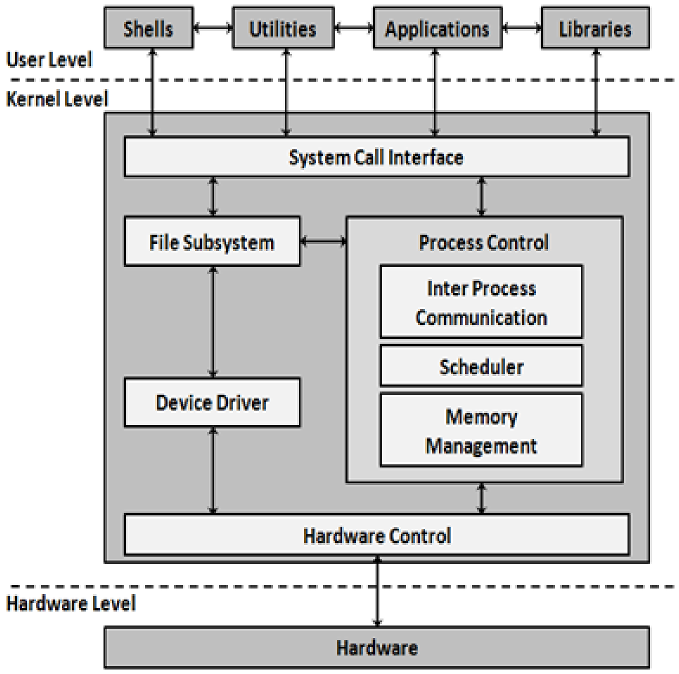
➢ **Documentation:**
UNIX provides an excellent **feature of online help.** The principle online help facility available is the ***man*** command, which is the most important reference for commands and their configuration. It **gives** detailed information about **all the commands**.

➢ **Pattern Matching:**
UNIX features very sophisticated pattern matching feature. There are **some special characters** (like *,?,[..]) through UNIX **provides very nice feature** of

pattern matching. There is also facility of **regular expression** that is framed with characters from this set.

System Structure





Kernel:

The kernel is the **core of the operating system**. Kernel is mostly written in C. It is **loaded into memory** when the **system is booted** and **communicates directly with the hardware**. User programs that need to access the hardware use the services of the ***kernel***, which **performs the job on the user's behalf**. The programs access kernel through set of functions called **system calls**. The kernel **also manages** the system memory, Schedules processes, decides their priorities and performs other task.

Shell:

The shell is the **command interpreter**. It acts as an **interface between the user and kernel**. When the **user types any command** it is interpreted by **shell** and then given to the **kernel.** There could be several copies of shell running on the system. For each user who logs in the system a different shell is created.

System Calls:

In UNIX there are thousand commands, all these commands **use a handful functions** called System calls **to communicate with the kernel**. The system calls are always same in all the versions of UNIX. In

UNIX there are many system calls to perform different tasks.**e.g.:** a typical UNIX command *writes* the file with the **write** system call, for *opening* a file UNIX uses **open** system call. The system call in UNIX works same for a file and device (as **device is also considered as a file** in UNIX). The **system calls are built into kernel**, and **interaction through them** represents **an efficient means of communication** with the system **i.e.** once the software has been developed on one UNIX system, it can easily be ported to another UNIX machine.

The Files and Processes:

Two simple entities support the UNIX system-the file and process. A <u>File</u> is just an array of bytes and can contain virtually anything. It is also related to another file by being part of a single hierarchical structure. UNIX does not care to know the type of file you are using. Directories and devices are considered as members of file system. UNIX provides a vast array of text manipulation tools that can edit the files without using an editor.

<u>Process</u> is a file in execution. Processes also belong to hierarchical tree structure. Processes are treated as living organism in UNIX as they have parents, children and grandchildren and they born and die also. UNIX provides the tools that allow us to control processes move them between foreground and background and even kill them.

Tools & Application:

The outermost layer of the UNIX operating system is its tools and applications. Tools vary from one implementation of UNIX to another. Some versions of UNIX are decked with more than 400 tools and applications. These tools can be invoked from the command line itself and help perform the day-to-day as well as complex tasks of the system. These are placed one level above shell and can be expanded and patched as required by the user. The tools are not mandatory, hence different implementations of UNIX have varying number of tools and applications available.

Shell & Its Features

Shell: Shell is command interpreter in UNIX. It accepts commands from the user and analyzes and interprets these commands. Shell requests the kernel to carry out the actual transfer of data which finally leads to the output that is displayed on the terminal. The shell hence acts an interface between the user and the kernel. When user logs on the system a separate copy of his shell is created. This means at a particular instance there may be several copies of shell running on the system with a minimum of one Shell per user. The shell program is stored in file called 'sh'.

There are **several different** shells available for UNIX:

You can use any one of these shells if they are available on your system. And you can switch between the different shells once you have found out if they are available.

- Bourne Shell (sh)
- C Shell (csh)
- TC Shell (tcsh)
- Korn Shell (ksh)
- Bourne Again Shell (bash)

Bourne Shell (sh):
This is the original UNIX shell written by **Steve Bourne** of Bell Labs in late **1970**'s. It is **available on all** UNIX systems. The ubiquitous "**dollar prompt**" on UNIX installation is trademark of Bourne Shell
This shell **does not have the interactive facilities** provided by modern shells such as the C shell and Korn shell. You are advised to use another shell which has these features.
The Bourne shell does provide an easy to use language with which you can write shell scripts.

C Shell (csh):
This shell was written at the **University of California**, Berkeley by **Bill Joy** who was a graduate student there. This is the default shell in the Berkeley versions of UNIX and is very popular with UNIX programmer and university researchers. It **provides a C-like language** with which **to write shell scripts** - hence its name.

It also provides some principal advantages over the Bourne Shell like,

- **<u>A History mechanism:</u>**
  The Shell remembers the commands that the users types and allows him to recall them without having them typing again. Hence that avoids lots of error as UNIX commands are too long that the user may miss spell.
- **<u>Aliasing:</u>** The C shell permits you to call frequently used commands by your own formulated abbreviations. This too proves very useful at the command line. This is a type of "macro" facility that is available at the command line.

TC Shell (tcsh)
This shell is available in the public domain. It provides all the features of the C shell together with ***emac*** style editing of the command line

Korn Shell (ksh)
This shell was written by **David Korn** of **Bell labs**. It is now provided as the standard shell on UNIX systems. It **provides all the features of the C and TC shells** together with a shell programming language similar to that of the original Bourne shell.
It is the most efficient shell.

Bourne Again Shell (bash)
This is a public domain shell written by the Free Software Foundation under their GNU initiative. Ultimately it is intended to be a full implementation of the IEEE Posix Shell and Tools specification. This shell is widely used within the academic community.
bash provides all the interactive features of the C shell (csh) and the Korn shell (ksh). Its programming language is compatible with the Bourne shell (sh).

Features of Shells:

- **Interactive Environment:**The shell allows the user to create a dialog (communication channel) between the user and the host UNIX system. This dialog terminates until the user ends the system session.
- **Shell scripts:** It is the shell that has the facility to be programmed; the shell contains commands that can be utilized by the user. Shell scripts are group of UNIX command string together and executed as individual files. The shell is itself a program; accept that is written in "C" language.
- **Input/Output Redirection:**Input/Output Redirection is a function of shell that redirects the o/p from program to a destination other than screen. This way you can save the o/p from a command into a file and redirect it to a printer, another terminal on the h/w or even another program. similarly, a shell can be a program that accepts i/p form other than keyboard by redirecting its i/p from another source.
- **Piping Mechanism:** Pipe facility allows the o/p of one command to be used as input to another UNIX command e.g. who **|** wc. The program that performs simple functions can easily be connected to perform more complex functions minimizing the need to develop new program.
- **Meta Character Facility:** Shell recognizes "*","?" or "[..]" as a special characters when reading the arguments from the command line, shell than perform file

name expansionthis list before executing the requested program. E.g. ls s* (enter) Displays the file or directories starting with s character.

- **Background Processing:** Multitasking facility allows the user to run command in the background. This allows the command to be processed while the user can proceed with other task when a background task is completed, the user is notified.
- **Customized Environment:** The shell is your working environment facilities are available by which the shell can be customized for your personal need.
- **Programming Language:** The shell includes features that allow it to be used as programming language. This feature can be used to build shell script that performscomplex operations.
- **Shell variables:** The user can control the behavior of the shell as well as other programs & utilities by storing data in variables.

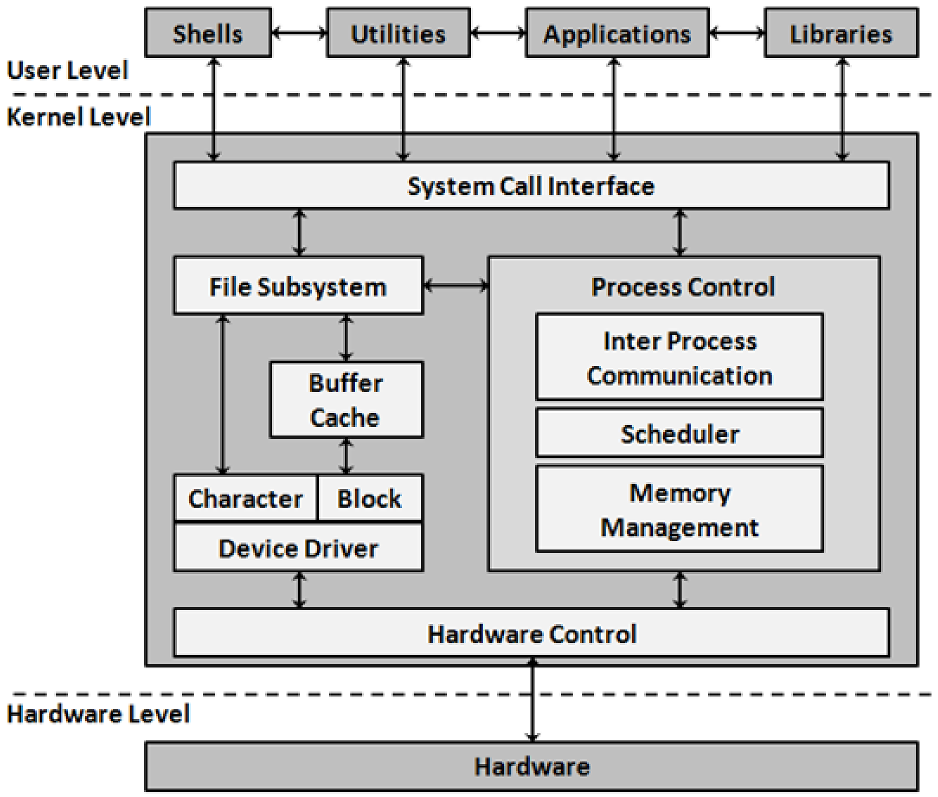
KERNEL

The kernel is the core of the operating system. Kernel is mostly written in C. It is loaded into memory when the system is booted and communicates directly with the hardware. User programs that need to access the hardware use the services of the kernel, which performs the job on the user's behalf. The programs access kernel through set of functions called system calls. The kernel program is usually stored in a file called "Unix".
Kernel manages many functions as given below:

- Manages files
- Carries out all the data transfer between the file system and hardware
- Manages memory
- Schedules various programs running in memory
- Handles interrupts etc...

Architecture of UNIX (Kernel Architecture) * (in exam write few sentences of kernel, shell also)

The UNIX architecture can be divided into ***three*** levels: User level, Kernel level, Hardware level.



The system call and library Interface represent the border between user programs and the kernel as shown in the figure. System calls are ordinary function calls in C programs and libraries map these functions calls to primitive needed to enter the operating system. Programs frequently use other libraries such as standard I/O library to provide more sophisticated use of the system calls. The libraries are linked with the programs at compile time.

The **system calls** are partitioned to the system calls that interact with the file sub system and the system calls that interact with the process control subsystem.

The **file subsystem manages** the files, allocates files space, administrating the free space, controlling access to files, and retrieving data for users. Processes interact with the file system through system calls. E.g. Open, close, read, write, etc. The files subsystem accesses the data using buffering mechanism that regulates dataflow between the kernel and secondary storage devices. The buffering mechanism interacts with block I/O devices drivers to initiate data transfer to and from the kernel.

Device drivers are the kernel modules that control the operation of peripheral devices. Block I/O devices are random access storage device to the rest of the system.

The **file subsystem also interacts** directly with the raw (Character devices) I/O device drivers without the intervention of a buffering mechanism.

The **process control subsystem** is responsible for process synchronization, inter-process communication, memory management, process scheduling. The system calls used with process control systems are fork (creating a new process), exec (overlay the image of a program onto the running process), Exit (finish executing a process), wait (Synchronize process execution with the exit of a previously forked process), brk (control the size of memory allocated to a process) and signal(control process response to extraordinary events).

The **memory management** module controls the allocation of memory. If at any time the system doesn't have enough physical memory for all processes, the kernel moves between main memory and secondary memory so that the all processes get a fair chance to execute.

The **scheduler module** allocates the CPU to processes. It schedules them to run in turn until they voluntarily relinquish the CPU while awaiting a resource or until the kernel preempts them when their recent run time exceeds a time quantum. The scheduler then chooses the highest priority eligible process to run; the original process will run when it is the highest priority eligible process available.

The **inter-process communication** provides message passing between processes. i.e.: it facilitates the communication between processes.

The**hardware control** is responsible for handling interrupts and for communicating with the machine. Devices such as disks or terminals may interrupt the CPU while a process is executing. The kernel executes the interrupt and then resumes the previously executing process. This way it provides access of hardware devices.