



Introduction

UNIX AND SHELL PROGRAMMING

UNIT-I

Overview

- ▶ It is designed to let a no. of programmers to access the computers at the same time and share its resources.
- ▶ It is mainly developed by programmers for programmers.
- ▶ OS coordinates the use of computer resources.
- ▶ Many user can log on at the same time and process their work.
- ▶ Initially it was designed for mini computers and later on it moved to larger, more powerful mainframe computers.

History

- ▶ In 1969 Ken Thompson comes up with the idea to make a general purpose operating system and with the help of Denis Ritchie he implemented his idea and created single user system in Bell Laboratories.
- ▶ In 1973, Thompson and Ritchie rewrote the UNIX operating system in C, breaking away from the traditional system software language, Assembly.
- ▶ Around 1974, it was licensed to universities for educational purposes and a few years later UNIX was made commercially available.

Uniqueness of UNIX

The features that made UNIX a hit from the start are:

- ▶ Multitasking Capability
- ▶ Multi-user Capability
- ▶ Unix Programs
- ▶ Library of application software.

Multi-User Operating System

Multi-user operating system is expected to perform various functions, which can be broadly categorized as :

- ▶ Command interpretation
- ▶ Peripheral Management
- ▶ Memory Management- is the extremely important job of allocation memory for various jobs being performed on the system and even disposing of useless data in memory after it has been processed and used.
- ▶ Process Management


Benefits of UNIX

- ▶ Portability
 - ▶ Machine Independent – the system hides the machine architecture from the user, making it easier to write the applications that can run on micros, minis and mainframes.
- ▶ Multi-user Operations
 - ▶ Hierarchical file system- UNIX uses a hierarchical structure to store information. This structure has the maximum flexibility in grouping information in a way that reflects its natural state. It allows for easy maintenance and efficient implementation.
- ▶ Unix Shell
 - ▶ Pipes and Filters- UNIX has facilities called pipes and filters, which permit the user to create complex programs from simpler programs.
- ▶ Utilities
- ▶ Background Processing
- ▶ Software Development tools
- ▶ Maturity – Unix is a time-tested operating system. It offers a bug free environment and high level of reliability

How UNIX is organized

Kernel –

- ▶ The kernel takes responsibility for deciding at any time which of the many running programs should be allocated to the processor or processors
- ▶ The kernel is responsible for deciding which memory each process can use, and determining what to do when not enough is available.
- ▶ The kernel allocates requests from applications to perform I/O to an appropriate device (or subsection of a device, in the case of files on a disk or windows on a display) and provides convenient methods for using the device.
- ▶ Kernels also usually provide methods for synchronization and communication between processes (called inter-process communication or IPC).



Operating system tasks are done differently by different kernels, depending on their design and implementation.

- ▶ Monolithic kernels execute all the operating system code in the same address space to increase the performance of the system,
- ▶ Micro kernels run most of the operating system services in user space as servers, aiming to improve maintainability and modularity of the operating system.

A range of possibilities exists between these above two extremes.

- ▶ If we don't have kernel then we need to design the application which would be able to communicate with the hardware not to use abstraction layer. But this will increase the complexity of the system.

Kernel Working

- ▶ The boot loader starts executing the kernel in supervisor mode. The kernel then initializes itself and starts the first process. After this, the kernel does not typically execute directly, only in response to external events (e.g., via system calls used by applications to request services from the kernel, or via interrupts used by the hardware to notify the kernel of events). Additionally, the kernel typically provides a loop that is executed whenever no processes are available to run; this is often called the idle process.

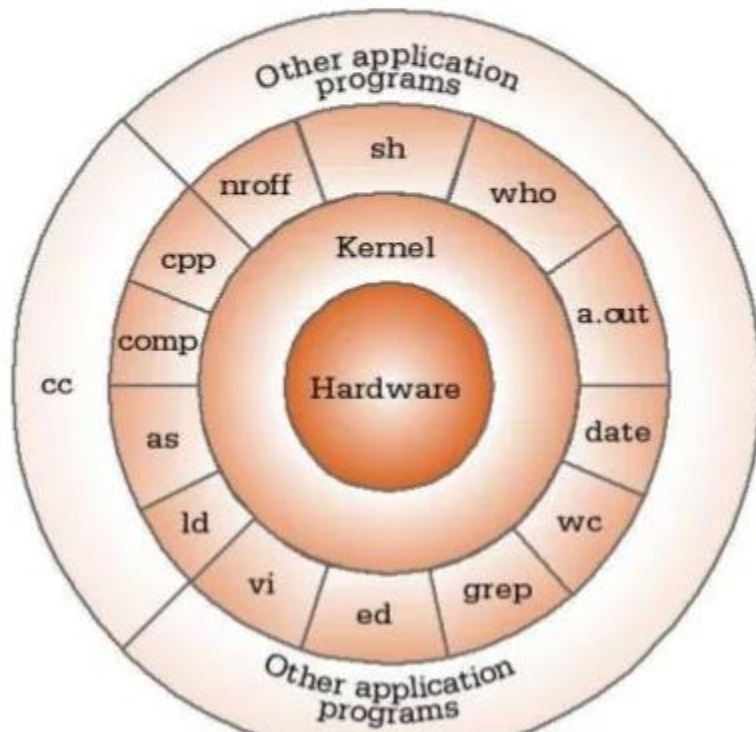
Shell

- ▶ It acts as an interpreter between the user and the computer.
- ▶ It also provides the functionality of "pipes".
- ▶ Pipes- a number of commands can be linked together by a user permitting the output of one program to become the input of another program or command.

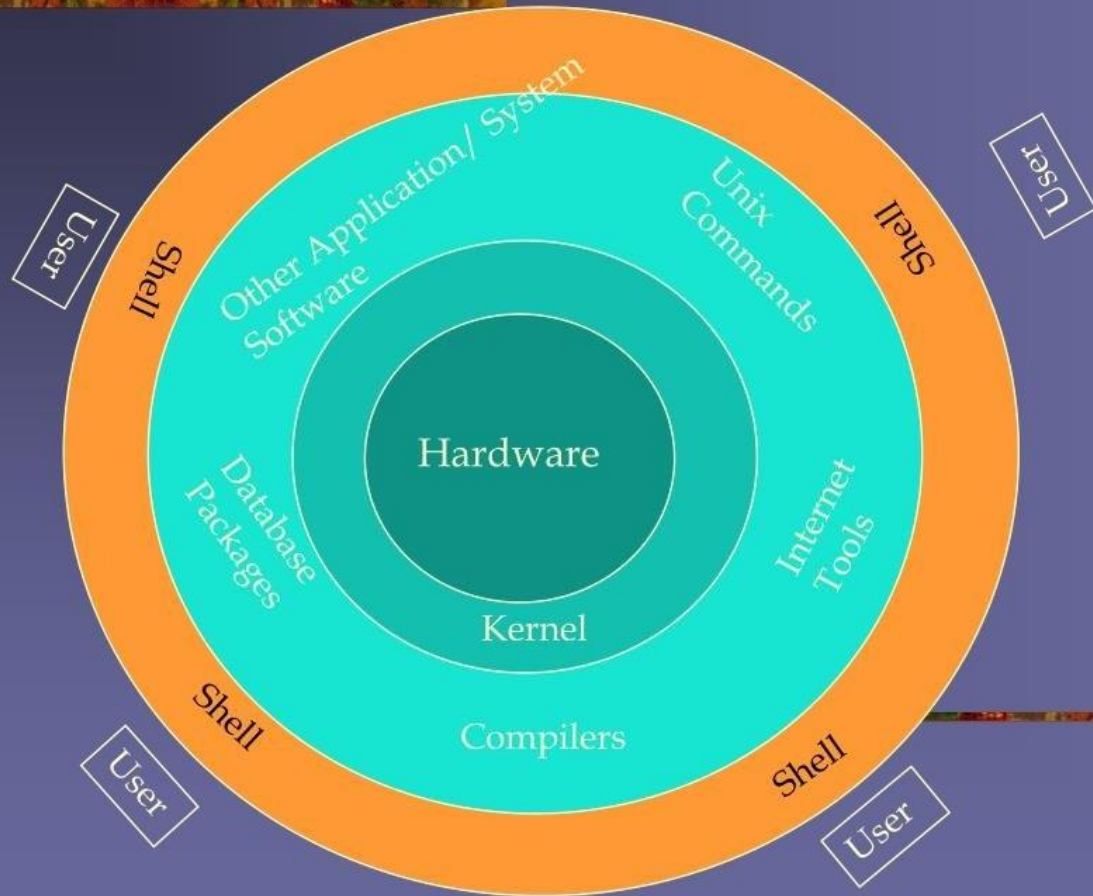
The Shell

- The UNIX user interface is called the *shell*.
- The shell does 4 jobs repeatedly:





Kernel-Shell Relationship



Your Account

- Each user has their own space called their *account*.
 - Type your login ID and password to enter your account.
 - Only if the login ID and password match will you be let in.
-

Login to your Account

login: `ad`

You type your ID and **RETURN**.

Password:

You type your password and RETURN. It does not appear.

\$

The UNIX prompt (or similar).
You can now enter

commands.

Logout from your Account

logout

or

^D

Press CONTROL and D
together

or

exit

On-line Help

- `man` Manual pages
Spacebar to go on; `^C` to stop

```
man gnuchess  
man man
```

- `apropos` *topic* Lists commands
related to `topic`

```
apropos game  
apropos help
```

UNIX Books

- **The Unix Programming Environment**, Brian W. Kernighan and Rob Pike.
Prentice Hall, Inc., 1984.
- Sumitabha Das, "Unix : Concepts and Applications"
- *A Student's Guide to UNIX*, Harley Hahn, McGraw-Hill, 1993
- *A Practical Guide to the UNIX System*, Mark G. Sobell, Benjamin-Cummings, 3rd Edition, 1995

Typing Commands

- Try these:

```
date
```

```
cal 3 2005
```

```
who
```

```
ls -a
```

```
man cal
```

```
clear
```

Changing your Password

- The command is:

```
passwd
```

- It will ask you for the new password twice.

Date Commands

■ `date` Gives time and date

■ `cal` Calendar

```
cal 1997
```

```
cal 3
```

```
cal 7 1962
```

```
cal 9 1752
```

You and the System

- `uptime` Machine's 'up' time
- `hostname` Name of the machine
- `whoami` Your name
- `who`

Calculators

- `expr e`

Simple arithmetic

`expr 3 + 5 + 7`

- `bc`

Programmable
Calculator

Some General Purpose Commands

date	locate
cal	more
who	passwd
ls	echo
man	banner
clear	tty
uptime	uname
hostname	tput
quota	spell
whoami	ispell
apropos	cat
whatis	sort
which	pwd

Redirection, pipes , processes

- Output can be *redirected* to a file with '`>`':

```
ls > dir.txt
```

```
cal 2004 > year2004
```

- Output can be *appended* to a file with '`>>`'

```
cal 2004 > years
```

```
cal 2005 >> years
```

- Pipes : sending the output of one program to the input of the other

```
ls | sort
```

```
who | sort
```

- Processes : Running two commands sequentially

```
locate mj > xxx; date
```

```
locate usr > xxx &
```

File System

- ▶ The unix file system includes directories containing files and directories, each directory of which can contain yet more file and files and directories.
- ▶ Managing file system is one of the important task of the system administrator.

File Types

There are four types of files in unix-

- ▶ Ordinary files :
 - ▶ An ordinary file may contain text, a program or other data in either ASCII form or in Binary form.
- ▶ Directory files
 - ▶ Suppose directory x contains a,b,c and that b is a directory, and b contains u and v files.
- ▶ Device files
- ▶ Link files

Unix Directory Structure

There are some directories in unix which will install with the unix operating system as a dependency under the root(main) directory are:

- ▶ Etc : Contains all system configuration files and the files which maintain information about the user and group.
- ▶ Bin : Contains all binary executable
- ▶ Usr : default directory provided by UNIX OS to create users home directories and contains manual pages
- ▶ Tmp: System or users temporary files which will be removed when the system reboots.
- ▶ Dev: Contains all device files i.e. logical names to physical devices.
- ▶ Devices : Contains all the device files. i.e. physical names to physical devices.
- ▶ Home : default directory allocated for the home directories of normal users when the administrator don't specify any other directory.
- ▶ Var : contains all the system log files and message files.
- ▶ Sbin : Contains all the system administrator executable files.

Internal vs External Commands

▶ Internal Command

- ▶ Internal commands are something which is built into the shell. For the shell built in commands, the execution speed is really high. It is because no process needs to be spawned for executing it. For example, when using the "cd" command, no process is created. The current directory simply gets changed on executing it.

▶ External Command

- ▶ External commands are not built into the shell. These are executable present in a separate file. When an external command has to be executed, a new process has to be spawned and the command gets executed. For example, when you execute the "cat" command, which usually is at /usr/bin, the executable /usr/bin/cat gets executed.
- ▶ Use keyword "type" to know whether the command is Internal or External.

Major task of system administrator are

- ▶ Making files available to users.
- ▶ Managing and monitoring the system disk resource.
- ▶ Protecting against file corruption, hardware failures, user errors through backup.
- ▶ Security of these filesystems, what users need and have access to which files.
- ▶ Adding more disks, tape drives, etc when needed.



Introduction with UNIX Basic Commands

Directory Handling

- ▶ `mkdir` : used to create a new directory.
- ▶ `rmdir` : used to remove the directory.
- ▶ `pwd` : to find out the path of the directory.
- ▶ `cd` : it is used to change the directory.

Path Variables

- ▶ **Absolute paths**

- ▶ `cd /home/nishant`
- ▶ `cd /home/nishant/my`

- ▶ **Relative paths**

- ▶ `cd ../nishant/my`
- ▶ `cd ~nishant/my`

File Operations

- ▶ **cat**

- ▶ `cat > newfile:` will create a new file.
- ▶ `cat newfile:` will show all the data of the newfile.

- ▶ **cp**

- ▶ `cp file1 file2 :` will create a new file which is the copy of file1 with the name of file2.

- ▶ **rm**

- ▶ `rm file1:` will delete the file1.
- ▶ `rm -r mydir:` will delete the directory named mydir.

- ▶ **mv**

...

▶ **ls**

- ▶ ls : will list all the files and directories currently present in the directory.
- ▶ ls -a: will list all the files including hidden files.
- ▶ ls -l : will list all the files currently into the directory and also all the details of it.

▶ **Ln**

- ▶ ln x y: will create a shortcut of the file x with the new name y.

▶ **chown**

- ▶ chown username filename: will change the file owner.

▶ **chmod**

- ▶ chmod options filename: will change the permissions of the file.

Process Operations

- ▶ **ps**

- ▶ ps -aux: will show all the process running into the system currently.
- ▶ ps -u username: will show all the process related to that user only.

- ▶ **Kill**

- ▶ kill process_id: will kill the specific process.


General Purpose Utilities

- ▶ **Cal**
- ▶ **Date**
- ▶ **Echo**
- ▶ **printf**- Formatted output.
- ▶ **Bc**- echo "2+3" | bc. Evaluates the expression.
- ▶ **Script**- creates a log file for all the commands between script and exit command.
- ▶ **Passwd**
- ▶ **Who**- displays list of users currently logged in.
- ▶ **Uname**- it writes the operating system characteristics.

Understanding UNIX / Linux filesystem

Inodes

- ▶ The **inode (index node)** is a fundamental concept in the Linux and UNIX filesystem. Each object in the filesystem is represented by an inode. But what are the objects? Let us try to understand it in simple words. Each and every file under Linux (and UNIX) has following attributes:
- ▶
 - => File type (executable, block special etc)
 - => Permissions (read, write etc)
 - => Owner
 - => Group
 - => File Size
 - => File access, change and modification time (remember UNIX or Linux never stores file creation time, this is favourite question asked in UNIX/Linux system admin job interview)
 - => File deletion time
 - => Number of links (soft/hard)
 - => Extended attribute such as append only or no one can delete file including root user (immutability)
 - => Access Control List (ACLs)



All the above information stored in an inode. In short the inode identifies the file and its attributes (as above) . Each inode is identified by a unique inode number within the file system. Inode is also know as index number.

Inode Definition

An inode is a data structure on a traditional Unix-style file system such as UFS or ext3. An inode stores basic information about a regular file, directory, or other file system object.

\$ ls -li /etc/passwd- shows the inode number

Useful Commands

Creating Files

- ▶ `cat > filename:` to create file.
- ▶ `vi filename`

Copying and moving files

- ▶ `cp filename /destinationfolder/filename`
- ▶ `mv filename newfilename`

Creating links

- ▶ `ln filex filey:` will create a shortcut of the filex with the new name filey.

Relative and Absolute Pathnames

► Relative Path Names

The use of the ".." notation allows us to navigate the directory tree structure. The ".." symbol means "parent directory." Names with ".." in them are relative names because their meaning depends on where they are issued (the present working directory). we can string together several ".." symbols, separated by the / symbol and other directory names, to change directories.

► Absolute Path Names

If we string together the unique name of all the intervening subdirectories in the file system to a particular subdirectory, we have created the absolute pathname for that directory. The absolute pathname allows us to switch to a directory no matter what my present working directory is. Absolute pathnames always start with a "/". we can navigate the file system by using absolute pathnames.

Relative and Absolute Pathnames Examples

▶ Relative Path Names Example

```
$ pwd
/users/john/portfolio
$ cd ../../mary
$ pwd
/users/mary
```

▶ Absolute Path Names Example

```
$ pwd
/users/john
$ cd /users/mary
$ pwd
/users/mary
$ cd /tmp
$ pwd
```