Introduction: UNIX is an operating system which was first developed in the 1960s, and has been under constant development ever since. By operating system, we mean the suite of programs which make the computer work. It is a stable, multi-user, multi-tasking system for servers, desktops and laptops.

UNIX systems also have a graphical user interface (GUI) similar to Microsoft Windows which provides an easy to use environment. However, knowledge of UNIX is required for operations which aren't covered by a graphical program, or for when there is no windows interface available, for example, in a telnet session.

Types of Unix: There are many different versions of UNIX, although they share common similarities. The most popular varieties of UNIX are Sun Solaris, GNU/Linux, and MacOS X.

Here in the School, we use Solaris on our servers and workstations, and Fedora Linux on the servers and desktop PCs.

The UNIX operating system is made up of three parts; the kernel, the shell and the programs.

The kernel

The kernel of UNIX is the hub of the operating system: it allocates time and memory to programs and handles the file store and communications in response to system calls.

As an illustration of the way that the shell and the kernel work together, suppose a user types **`rm myfile`** (which has the effect of removing the file **myfile**). The shell searches the file store for the file containing the program **`rm`**, and then requests the kernel, through system calls, to execute the program **`rm`** on **myfile**. When the process **`rm myfile`** has finished running, the shell then returns the UNIX prompt % to the user, indicating that it is waiting for further commands.

The shell

The shell acts as an interface between the user and the kernel. When a user logs in, the login program checks the username and password, and then starts another program called the shell. The shell is a command line interpreter (CLI). It interprets the commands the user types in and arranges for them to be carried out. The commands are themselves programs: when they terminate, the shell gives the user another prompt (% on our systems).

The adept user can customise his/her own shell, and users can use different shells on the same machine. Staff and students in the school have the **tcsh shell** by default.

The tcsh shell has certain features to help the user inputting commands.

Filename Completion - By typing part of the name of a command, filename or directory and pressing the [**Tab**] key, the tcsh shell will complete the rest of the name automatically. If the shell finds more than one name beginning with those letters you have typed, it will beep, prompting you to type a few more letters before pressing the tab key again.

History - The shell keeps a list of the commands you have typed in. If you need to repeat a command, use the cursor keys to scroll up and down the list or type history for a list of previous commands.

The programs

One of the main features of UNIX is that it includes a variety of small programs to meet various needs. Typically, each of these programs does one thing and does it well. This modular design allows the functionality of small programs to be mixed and matched. As you become more familiar with UNIX, you will find that this design provides you great flexibility and power to accomplish almost any task. Typically these programs operate on top of the shell, but they may also interface directly with the kernel.

Linux is one of popular version of UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

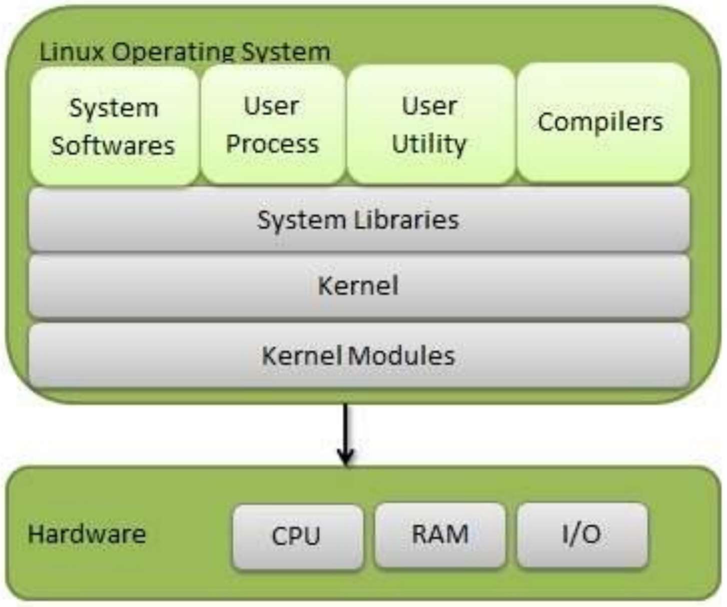
Components of Linux System

Linux Operating System has primarily three components

- **Kernel** − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

- **System Library** − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.

- **System Utility** − System Utility programs are responsible to do specialized, individual level tasks.



Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called **kernel mode** with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in **User Mode** which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Basic Features

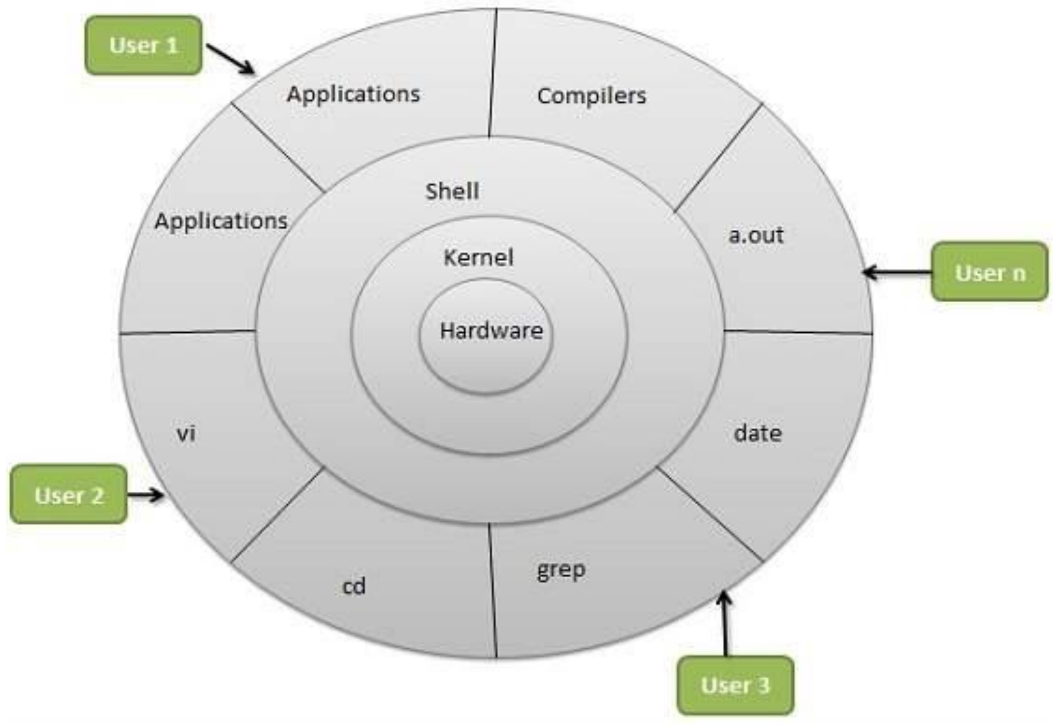
Following are some of the important features of Linux Operating System.

- **Portable** – Portability means software can works on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
- **Open Source** – Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** – Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.

- **Multiprogramming** – Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** – Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell** – Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
- **Security** – Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Architecture

The following illustration shows the architecture of a Linux system –



The architecture of a Linux System consists of the following layers –

- **Hardware layer** – Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** – It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** – An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.
- **Utilities** – Utility programs that provide the user most of the functionalities of an operating systems.

Role of System Administrator and Ordinary User

There are three types of accounts on a Unix system −

Root account

This is also called **super user** and would have complete control of the system. A super user can run any commands without any restriction. This user should be assumed as a system administrator.

System accounts

System accounts are those needed for the operation of system-specific components for example mail accounts. This account is usually needed for some specific function on your system, and any modifications to them could adversely affect the system.

User accounts

User accounts provide interactive access to the system for users and groups of users. General users are typically assigned to these accounts and usually have limited access to critical system files and directories.

Unix supports a concept of *Group Account* which logically groups a number of accounts. Every account would be a part of another group account. A Unix group plays important role in handling file permissions and process management.

Managing Users and Groups

There are four main user administration files −

- **/etc/passwd** − Keeps the user account and password information. This file holds the majority of information about accounts on the Unix system.
- **/etc/shadow** − Holds the encrypted password of the corresponding account. Not all the systems support this file.
- **/etc/group** − This file contains the group information for each account.
- **/etc/gshadow** − This file contains secure group account information.

Check all the above files using the **cat** command.

The following table lists out commands that are available on majority of Unix systems to create and manage accounts and groups −

S.No.	Command & Description
1	**useradd** Adds accounts to the system
2	**usermod** Modifies account attributes
3	**userdel** Deletes accounts from the system
4	**groupadd** Adds groups to the system
5	**groupmod** Modifies group attributes
6	**groupdel** Removes groups from the system

You can use Manpage Help to check complete syntax for each command mentioned here.

Create a Group

We will now understand how to create a group. For this, we need to create groups before creating any account otherwise, we can make use of the existing groups in our system. We have all the groups listed in ***/etc/groups*** file.

All the default groups are system account specific groups and it is not recommended to use them for ordinary accounts. So, following is the syntax to create a new group account −

```
groupadd [-g gid [-o]] [-r] [-f] groupname
```

The following table lists out the parameters −

S.No.	Option & Description
1	**-g GID** The numerical value of the group's ID
2	**-o** This option permits to add group with non-unique GID
3	**-r** This flag instructs **groupadd** to add a system account
4	**-f** This option causes to just exit with success status, if the specified group already exists. With -g, if the specified GID already exists, other (unique) GID is chosen
5	**groupname** Actual group name to be created

If you do not specify any parameter, then the system makes use of the default values.

Following example creates a *developers* group with default values, which is very much acceptable for most of the administrators.

```
$ groupadd developers
```

Modify a Group

To modify a group, use the **groupmod** syntax −

```
$ groupmod -n new_modified_group_name old_group_name
```

To change the developers_2 group name to developer, type −

```
$ groupmod -n developer developer_2
```

Here is how you will change the financial GID to 545 −

```
$ groupmod -g 545 developer
```

Delete a Group

We will now understand how to delete a group. To delete an existing group, all you need is the **groupdel command** and the **group name**. To delete the financial group, the command is −

```
$ groupdel developer
```

This removes only the group, not the files associated with that group. The files are still accessible by their owners.

Create an Account

Let us see how to create a new account on your Unix system. Following is the syntax to create a user's account −

```
useradd -d homedir -g groupname -m -s shell -u userid accountname
```

The following table lists out the parameters −

S.No.	Option & Description
1	**-d homedir** Specifies home directory for the account
2	**-g groupname** Specifies a group account for this account
3	**-m** Creates the home directory if it doesn't exist
4	**-s shell** Specifies the default shell for this account
5	**-u userid** You can specify a user id for this account
6	**accountname**

	Actual account name to be created

If you do not specify any parameter, then the system makes use of the default values. The **useradd** command modifies the **/etc/passwd**, **/etc/shadow**, and **/etc/group** files and creates a home directory.

Following is the example that creates an account ***mcmohd***, setting its home directory to ***/home/mcmohd*** and the group as ***developers***. This user would have Korn Shell assigned to it.

```
$ useradd -d /home/mcmohd -g developers -s /bin/ksh mcmohd
```

Before issuing the above command, make sure you already have the *developers* group created using the ***groupadd*** command.

Once an account is created you can set its password using the **passwd** command as follows −

```
$ passwd mcmohd20
Changing password for user mcmohd20.
New UNIX password:
Retype new UNIX password:
passwd: all authentication tokens updated successfully.
```

When you type ***passwd accountname***, it gives you an option to change the password, provided you are a superuser. Otherwise, you can change just your password using the same command but without specifying your account name.

Modify an Account

The **usermod** command enables you to make changes to an existing account from the command line. It uses the same arguments as the **useradd** command, plus the -l argument, which allows you to change the account name.

For example, to change the account name ***mcmohd*** to ***mcmohd20*** and to change home directory accordingly, you will need to issue the following command −

```
$ usermod -d /home/mcmohd20 -m -l mcmohd mcmohd20
```

Delete an Account

The **userdel** command can be used to delete an existing user. This is a very dangerous command if not used with caution.

There is only one argument or option available for the command **.r**, for removing the account's home directory and mail file.

For example, to remove account *mcmohd20*, issue the following command −

```
$ userdel -r mcmohd20
```

If you want to keep the home directory for backup purposes, omit the **-r** option. You can remove the home directory as needed at a later time.

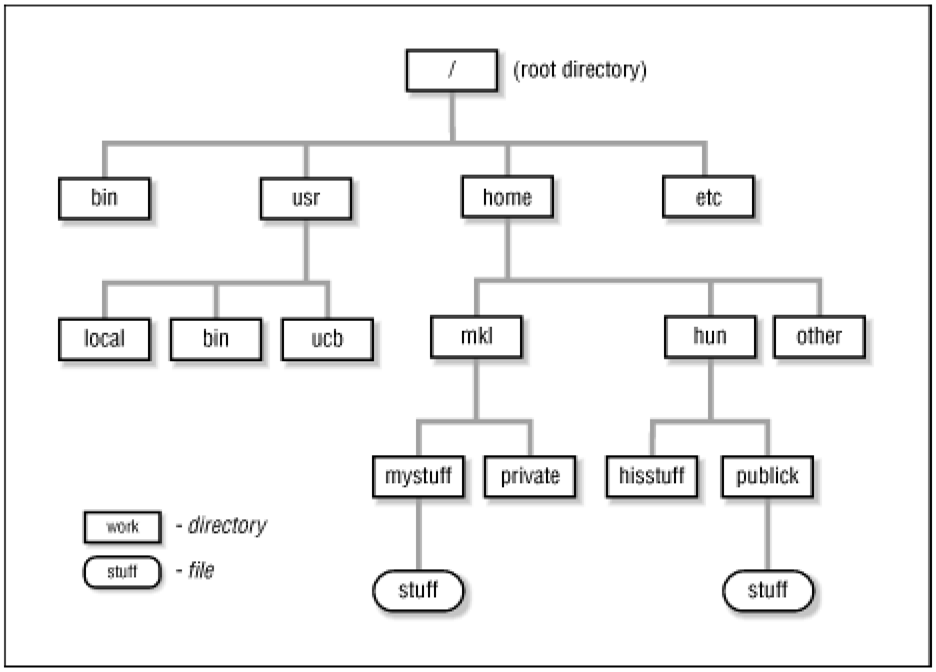
A file system is a logical collection of files on a partition or disk. A partition is a container for information and can span an entire hard drive if desired.

Your hard drive can have various partitions which usually contain only one file system, such as one file system housing the **/file system** or another containing the **/home file system**.

One file system per partition allows for the logical maintenance and management of differing file systems.

Everything in Unix is considered to be a file, including physical devices such as DVD-ROMs, USB devices, and floppy drives.

Tree Structure:



Directory Structure

Unix uses a hierarchical file system structure, much like an upside-down tree, with root (/) at the base of the file system and all other directories spreading from there.

A Unix filesystem is a collection of files and directories that has the following properties −

- It has a root directory (**/**) that contains other files and directories.
- Each file or directory is uniquely identified by its name, the directory in which it resides, and a unique identifier, typically called an **inode**.
- By convention, the root directory has an **inode** number of **2** and the **lost+found** directory has an **inode** number of **3**. Inode numbers **0** and **1** are not used. File inode numbers can be seen by specifying the **-i option** to **ls command**.
- It is self-contained. There are no dependencies between one filesystem and another.

The directories have specific purposes and generally hold the same types of information for easily locating files. Following are the directories that exist on the major versions of Unix −

S.No.	Directory & Description
1	**/** This is the root directory which should contain only the directories needed at the top level of the file structure
2	**/bin** This is where the executable files are located. These files are available to all users
3	**/dev** These are device drivers
4	**/etc** Supervisor directory commands, configuration files, disk configuration files, valid user lists, groups, ethernet, hosts, where to send critical messages

5	**/lib** Contains shared library files and sometimes other kernel-related files
6	**/boot** Contains files for booting the system
7	**/home** Contains the home directory for users and other accounts
8	**/mnt** Used to mount other temporary file systems, such as **cdrom** and **floppy** for the **CD-ROM** drive and **floppy diskette drive**, respectively
9	**/proc** Contains all processes marked as a file by **process number** or other information that is dynamic to the system
10	**/tmp** Holds temporary files used between system boots
11	**/usr** Used for miscellaneous purposes, and can be used by many users. Includes administrative commands, shared files, library files, and others
12	**/var** Typically contains variable-length files such as log and print files and any other type of file that may contain a variable amount of data
13	**/sbin** Contains binary (executable) files, usually for system administration. For

	example, ***fdisk*** and ***ifconfig*** utlities
14	**/kernel** Contains kernel files

Navigating the File System

Now that you understand the basics of the file system, you can begin navigating to the files you need. The following commands are used to navigate the system −

S.No.	Command & Description
1	**cat filename** Displays a filename
2	**cd dirname** Moves you to the identified directory
3	**cp file1 file2** Copies one file/directory to the specified location
4	**file filename** Identifies the file type (binary, text, etc)
5	**find filename dir** Finds a file/directory
6	**head filename** Shows the beginning of a file
7	**less filename**

	Browses through a file from the end or the beginning
8	**ls dirname** Shows the contents of the directory specified
9	**mkdir dirname** Creates the specified directory
10	**more filename** Browses through a file from the beginning to the end
11	**mv file1 file2** Moves the location of, or renames a file/directory
12	**pwd** Shows the current directory the user is in
13	**rm filename** Removes a file
14	**rmdir dirname** Removes a directory
15	**tail filename** Shows the end of a file
16	**touch filename** Creates a blank file or modifies an existing file or its attributes
17	**whereis filename**

	Shows the location of a file
18	**which filename** Shows the location of a file if it is in your PATH

You can use Manpage Help to check complete syntax for each command mentioned here.

The df Command

The first way to manage your partition space is with the **df (disk free)** command. The command **df -k (disk free)** displays the **disk space usage in kilobytes**, as shown below −

```
$df -k
Filesystem      1K-blocks      Used   Available Use% Mounted on
/dev/vzfs        10485760   7836644     2649116  75% /
/devices                0         0           0   0% /devices
$
```

Some of the directories, such as **/devices**, shows 0 in the kbytes, used, and avail columns as well as 0% for capacity. These are special (or virtual) file systems, and although they reside on the disk under /, by themselves they do not consume disk space.

The **df -k** output is generally the same on all Unix systems. Here's what it usually includes −

S.No.	Column & Description
1	**Filesystem** The physical file system name
2	**kbytes** Total kilobytes of space available on the storage medium
3	**used** Total kilobytes of space used (by files)
4	**avail**

	Total kilobytes available for use
5	**capacity** Percentage of total space used by files
6	**Mounted on** What the file system is mounted on

You can use the **-h (human readable) option** to display the output in a format that shows the size in easier-to-understand notation.

The du Command

The **du (disk usage) command** enables you to specify directories to show disk space usage on a particular directory.

This command is helpful if you want to determine how much space a particular directory is taking. The following command displays number of blocks consumed by each directory. A single block may take either 512 Bytes or 1 Kilo Byte depending on your system.

```
$du /etc
10     /etc/cron.d
126    /etc/default
6      /etc/dfs
...
$
```

The **-h** option makes the output easier to comprehend –

```
$du -h /etc

5k    /etc/cron.d

63k   /etc/default

3k    /etc/dfs

...

$
```

Mounting the File System

A file system must be mounted in order to be usable by the system. To see what is currently mounted (available for use) on your system, use the following command –

```
$ mount
/dev/vzfs on / type reiserfs (rw,usrquota,grpquota)
proc on /proc type proc (rw,nodiratime)
```

```
devpts on /dev/pts type devpts (rw)
$
```

The **/mnt** directory, by the Unix convention, is where temporary mounts (such as CDROM drives, remote network drives, and floppy drives) are located. If you need to mount a file system, you can use the mount command with the following syntax −

```
mount -t file_system_type device_to_mount directory_to_mount_to
```

For example, if you want to mount a **CD-ROM** to the directory **/mnt/cdrom**, you can type −

```
$ mount -t iso9660 /dev/cdrom /mnt/cdrom
```

This assumes that your CD-ROM device is called **/dev/cdrom** and that you want to mount it to **/mnt/cdrom**. Refer to the mount man page for more specific information or type mount **-h** at the command line for help information.

After mounting, you can use the cd command to navigate the newly available file system through the mount point you just made.

Unmounting the File System

To unmount (remove) the file system from your system, use the **umount** command by identifying the mount point or device.

For example, **to unmount cdrom**, use the following command −

```
$ umount /dev/cdrom
```

The **mount command** enables you to access your file systems, but on most modern Unix systems, the **automount function** makes this process invisible to the user and requires no intervention.

User and Group Quotas

The user and group quotas provide the mechanisms by which the amount of space used by a single user or all users within a specific group can be limited to a value defined by the administrator.

Quotas operate around two limits that allow the user to take some action if the amount of space or number of disk blocks start to exceed the administrator defined limits −

- **Soft Limit** − If the user exceeds the limit defined, there is a grace period that allows the user to free up some space.

- **Hard Limit** − When the hard limit is reached, regardless of the grace period, no further files or blocks can be allocated.

There are a number of commands to administer quotas −

S.No.	Command & Description
1	**quota** Displays disk usage and limits for a user of group
2	**edquota** This is a quota editor. Users or Groups quota can be edited using this command
3	**quotacheck** Scans a filesystem for disk usage, creates, checks and repairs quota files
4	**setquota** This is a command line quota editor
5	**quotaon** This announces to the system that disk quotas should be enabled on one or more filesystems
6	**quotaoff** This announces to the system that disk quotas should be disabled for one or more filesystems
7	**repquota** This prints a summary of the disc usage and quotas for the specified file systems