**Operating Systems**

The structure of a Computer system can be visualized as having four basic components:  
1. Hardware – Provides basic computing resources - CPU, memory, I/O devices  
2. Operating system – Controls and coordinates use of hardware among various applications and users  
3. Application programs – Define the ways in which the system resources are used to solve the computing problems of the users  
4. Users – Anybody who desires work to be done by a computer system. People, machines, other computers



**Types of OS**

OS are classified into different types depending on their capability of processing.

**Single User and Single Tasking OS:**

These are simple operating system designed to manage one task at a time ,for use by a single user for a standalone single computer for performing a single task .

**Single User and Multitasking OS:**

These OS allow execution of more than one task or process concurrently by dividing the processor time amongst different tasks.

**Multi-programming OS:**

These OS allow more than one programs to run at the same time .

**Real Time OS:**

These are designed to respond to an event within a predetermined time. These operating systems are used to control processes

**Embedded OS:**

Embedded in a device in the ROM. They are specific to a device and are less resource intensive.

**Function Of OS**

The OS performs basic tasks such as controlling and allocating memory, prioritizing system requests, controlling input and output devices, facilitating networking, and managing files

**UNIX Introduction**

* Unix is an OS for Programmers as shell(the command interpreter)provides the programming facility.
* It provides an in-built security mechanism through the user name and password, combined with the access rights associated with files
* Developed by Ken Thompson and Ritchie originally in assembly, and later in C, thus making it portable to other machines

 Supports C, Fortran, Basic, Pascal, COBOL, Lisp, Prolog, Java, Ada compilers

**Features of Unix Operating System**

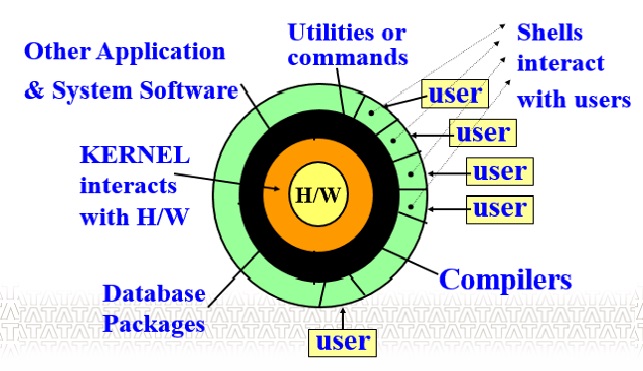
• Multi-user and Multitasking   
• Everything is a file   
• Configuration data stored in text   
• Small, single purpose programs   
• Ability to chain programs together to perform complex task   
• Facility of background processing

**Architecture of the UNIX System**

The UNIX operating system has a layered architecture having three main components   
• Kernel

• Shell

• Utilities



**The Unix Kernel**

Kernel is a collection of programs mostly written in C which allocate the system resources and coordinate all the details of the computer's internals.

**Functions of Kernel:**

* It allocates time and memory to programs and handles the file store and communications
* Interacts directly with the hardware through device drivers
* Provides sets of services to programs
* Manages memory, controls access, maintains file system, handles interrupts, allocates resources of the computer

**System calls**

The system calls are functions used in the kernel itself. UNIX system calls are used to manage the file system, control processes, and to provide interpocess communication.

System calls can be categorized as:

* File structure related calls -For example create, open,read, write,lseek,dup etc.
* Process Related calls  -For example fork,exec,wait,exit etc
* Inter process related calls - For example pipe,msgget,msgsnd etc.

**The Unix File System:**

It is a hierarchical collection of 3 types of files:

* ordinary
* directory
* special files (device, pipe, fifo, socket).

A UNIX file is featureless because it is simply an array of bytes.   
Dominant file type in UNIX is the text file.  
System related files are also stored in text form.  
Separate device can be added by creating a file for it.

Root is the supremo and is represented by the ‘/’. Every subdirectory must have  a parent.

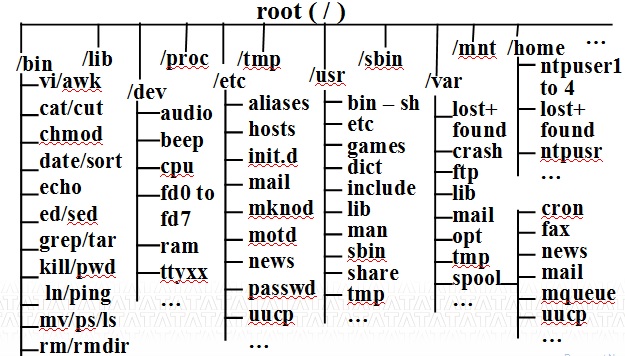
File names can be up to 14 characters long; can contain both upper and lower case alphabets, digits, a dot, hyphen (-), underscore (\_) anywhere; should not have a blank or tab; are case-sensitive.

Path names are a sequence of directory names separated by ‘/’. They are used to access files.

**Absolute pathname -** file location is determined with respect to the root.

**Relative pathname -**file location is determined with respect to the current directory.

Though the UFS looks hierarchical, it is actually a directed acyclic graph because files can be shared.



/home – It holds user’s home directories. In other UNIX systems, this can be /usr directory.

/bin – It holds many of the basic Linux programs; bin stands for binaries, files that are executable.

/usr – It holds many user-oriented directories:

/sbin – It holds system files that are usually run automatically.

/etc – It and its subdirectories hold many of Linux config files.

/dev – It holds device files. All info sent to /dev/null is thrown into trash. Your terminal is one of the /dev/tty files.

The UFS resides on a single logical disk. A logical disk is a disk partition comprising of a set of consecutive cylinders.

UFS further subdivides a partition into one or more cylinder groups and attempts to allocate inodes and related data blocks from the same cylinder group, thus minimizing the disk head movements.

At the beginning of the logical disk lies the boot block of UNIX operating system containing the bootstrap program.

It is followed by repetitive cylinder groups each one containing a super block, cylinder group block, inode list and the data area.

Each cylinder group contains a duplicate copy of the super block. The super block contains the size of file system, number of free blocks, index of next free block in free block list, size of inode list, number of free inodes, index of next free inode in free inode list.

The cylinder group block contains a number of inodes and corresponding data blocks for that cylinder group. The block size is a power of 2 (>=4096).

**Internal File Maintenance**

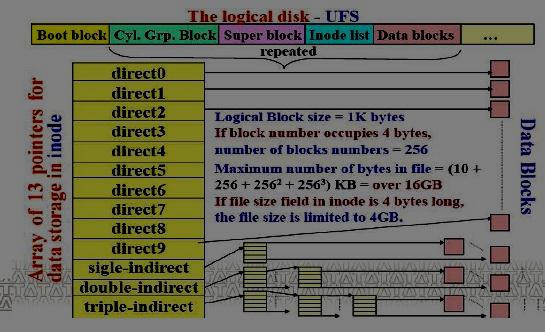
For each file created in the system, an inode is also created. Inode is a disk file record of 64 bytes that maintains the permanent attributes of a file.

An inode is permanent and it exists until the corresponding file is removed from the system.

Sample details of an inode –

* Owner and group identifiers
* File type and file size
* Number of links for this file
* Times of file creation, last file access and modification, and last inode modification
* List of access rights – read/write/execute permissions
* Reference count showing number of times file is opened
* Physical address of file on the disk: array of 13 pointers for data storage

Whenever a file is opened, its inode is brought into main memory. The active inode is kept there until the file is closed and is used to locate the beginning of an open file on disk and to verify that every I/O request comes from a valid user as per specified access permissions.



**Users and Access Rights**

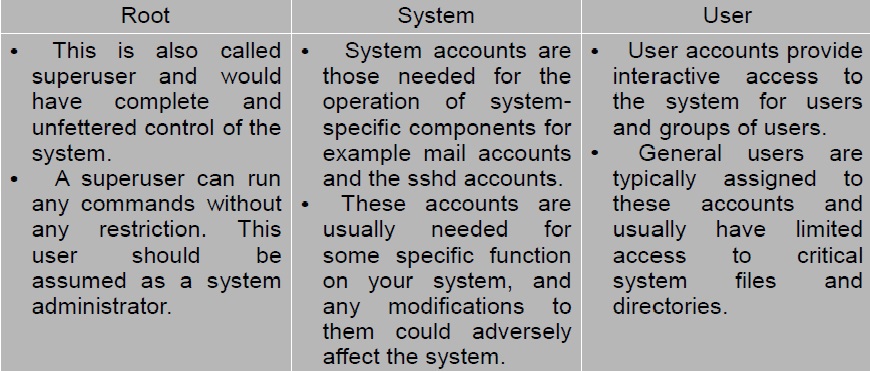
**Objective**  
This module is to understand about  different user types in UNIX and their access rights.

Course Content

* User types
* File access permissions
* Changing file access permissions using symbolic mode
* Changing file access permissions using octal mode

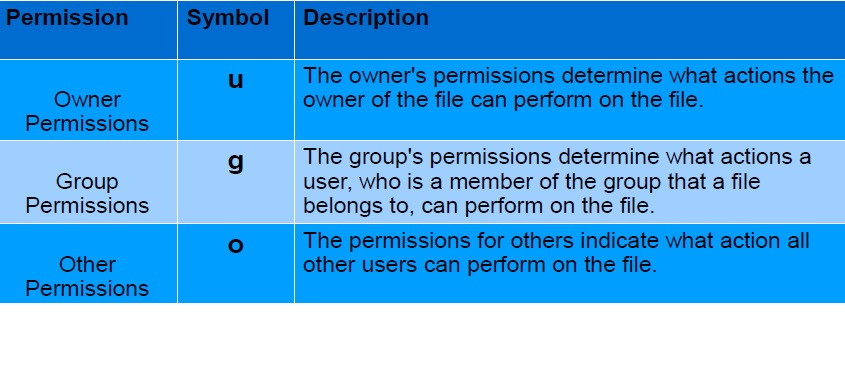
**User Types in Unix**

There are three types of accounts on a Unix system:  
- Root Account  
- System Account  
- User Account

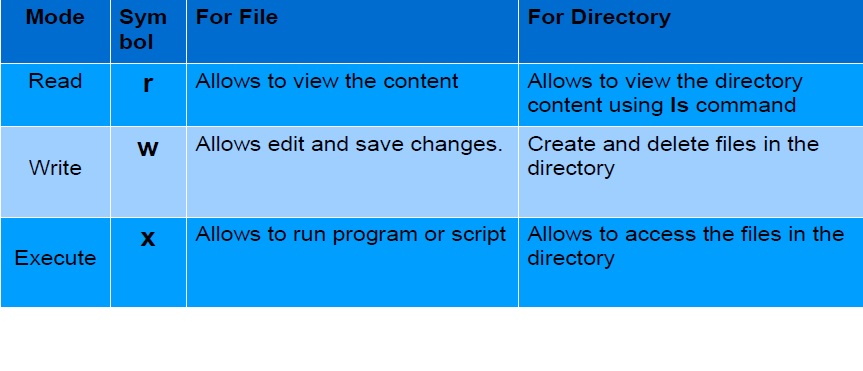


**Managing users and groups**  
Unix supports a concept of Group which logically groups a number of accounts.  
➢ Every account would be a part of any group.  
➢ Unix groups plays important role in handling file permissions and process management.  
➢ Grouping of users allows to grant and revoke file permissions collectively.

**File Permission: Ownerships level**  
File ownership is an important component of Unix that provides a secure method for storing files. Every file in Unix has the following attributes:

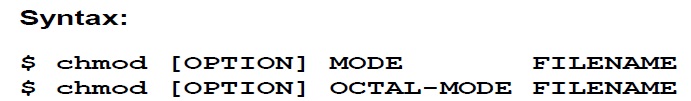


**File Permission Types (Mode)**  
Three type of permissions can be set for any ownership level:  
Read, Write and Execute Three modes or permissions have different meaning for file and directory;

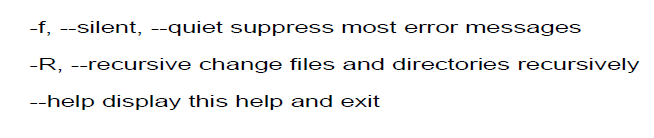


**Changing Permissions (chmod)**  
To change file or directory permissions, the **chmod** (change mode) command is used. There are two ways to use chmod:

* Symbolic mode
* Absolute mode

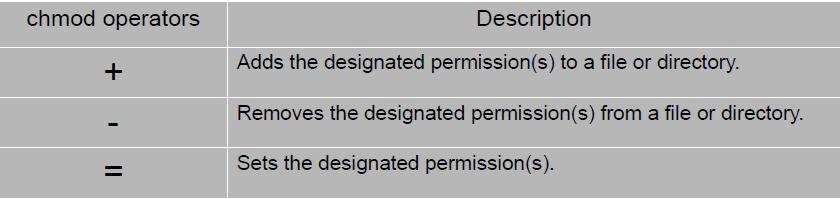


**chmod command options:**

****

**Using chmod in Symbolic Mode:**

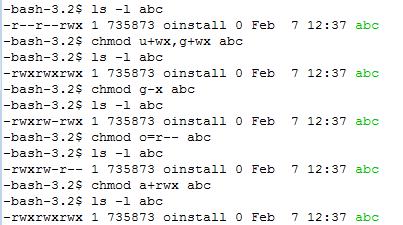
With symbolic representation, permission set can be added, deleted, or specified using the operators listed in the following table:



**Example:**

Let us assume there is a file name "abc" , whose initial permissions are only read permission to user and group and others. We will provide  full permissions to all three types of users.

Let us verify the existing file permissions using ls -l command as below and observe the next steps, like changing the permissions and verifying the changes.



In the above example,

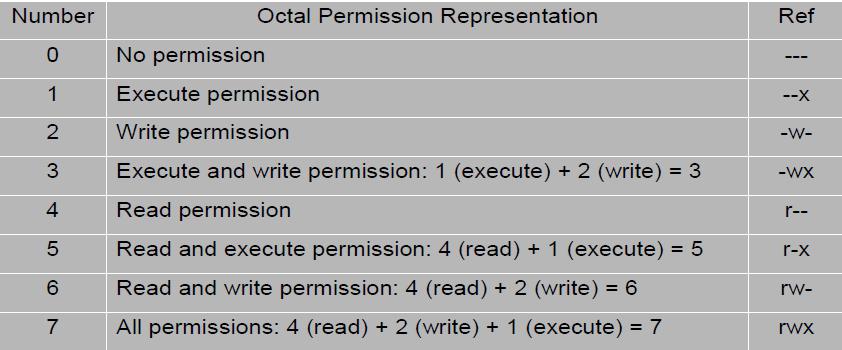
**chmod u+wx,g+wx abc** , adds write and execute permission to user and group to the file abc

**chmod g-x abc,**removes execute permission from group from the file abc.

**chmod o=r-- abc**, provides read permission and removes write and execute permission from others for file abc

**Using chmod in Absolute(octal) Mode**

The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file. Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set.



**Examples:** Observe the following sequence of command execution statements and analyze the output

1. Let us view the initial permissions of the file ***abc***  
    ls -l ***abc***

    -rw-r--r-- 1 735873 oinstall 0 Feb 7 12:37 abc

If we observe the output, initially owner have read and write permissions and group and others have only read permission

2. Let us remove all the permission from all users for ***abc***file, as below  
***chmod 0 abc***  
   Let us verify the changed permissions

   ls -l abc  
   ---------- 1 735873 oinstall 0 Feb 7 12:37 abc  
Yes all permissions are taken away from all users on the file abc

3. To provide all permissions to the user, read permission to group and write and execute permissions for Others,

    use the below command  
***chmod 743 abc***

Let us verify the changed permissions  
ls -l abc  
-rwxr---wx 1 735873 oinstall 0 Feb 7 12:37 abc  
From the output all permissions to user, read permission to group and write and execute  
permissions for Others, Use the below command

4. ***chmod 755 abc***  
    The above command provides all permissions to the user, read and execute permissions to group and others  
Let us verify ls -l abc  
-rwxr-xr-x 1 735873 oinstall 0 Feb 7 12:37 abc

5. ***chmod 777 abc***  
   The above command provides all permissions to all users . We can verify using below command  
   ls -l abc  
   -rwxrwxrwx 1 735873 oinstall 0 Feb 7 12:37 abc