UNIT-2

Unit 2. Overview

2.1 Logging in and Logging out

Unix session is a time period between logging and logged out.

There are two types of connection that a user of UNIX seeks to make with the system.

1. **Physical connection:**
   The *physical connection* implies that the terminal is adequately connected to the host using proper cables, cards and other necessary h/w. When the user gets the login prompt that signifies that there is a physical connection.
2. **Logical connection:**
   The *logical connection* is when user types the correct login and password; the host machine recognizes the user and allows him to enter into the system.

Logging in

putty

UNIX is a multiuser system. It has very strong security. User working with UNIX system generally works with terminal after the boot procedure is over, the kernel is loaded into memory and the following message appears at each user's terminal…

Login:

The user has to give his/her identification by enter the user name and password. The user name is also known as logging id. When password is entered, it is shown as ******. (In some systems to provide more security even ****** is not displayed.) If the password is correct the system allows you to work with it. If the password is wrong, it displays error message and gives the login prompt again.

login: tybcao1
password: ******(enter)

UNIX keeps track of all user name and password in a special file name */etc/passwd*.

We can *summarize* this process in **following steps:**

Step-1: Make contact with the system: Connecting with the system

Step-2: Wait for the system login prompt: Once we are connected with the system we need to wait for the login prompt that is displayed on user terminal.

Step-3: Type user-id: Enter the user name.

Step-4: Type password: Enter the password to enter in your system.

Once everything is correct then the users session starts and user can start entering commands on shell prompt.

$

Logging Out

When the user logs in, unless the user terminates, the session continues. It is very important that you log out when you are through with the system. The reasons for this are:

- It frees the resources allocated to the user
- And important reason is security, if the terminal is leaved logged anyone can start working in once login and may temper with important data.

To logout the system the user should perform any of the following commands depending upon the version of the UNIX he/she is using and even depending upon the shell.

- ***Logout:*** Once this command is typed the session gets over of particular user and at terminal we get the login prompt again.
- ***Exit****:* Typing exit at prompt
- ***Ctrl-d:*** Pressing control key and d from the keyboard.

2.2 UNIX Files

In UNIX a file is any source from which data can be read or any destination to which data can be written. Therefore the keyboard, monitor, printer documents stored on disk everything is considered to be a file.

There are very few **restrictions on how to make up filenames** in UNIX.

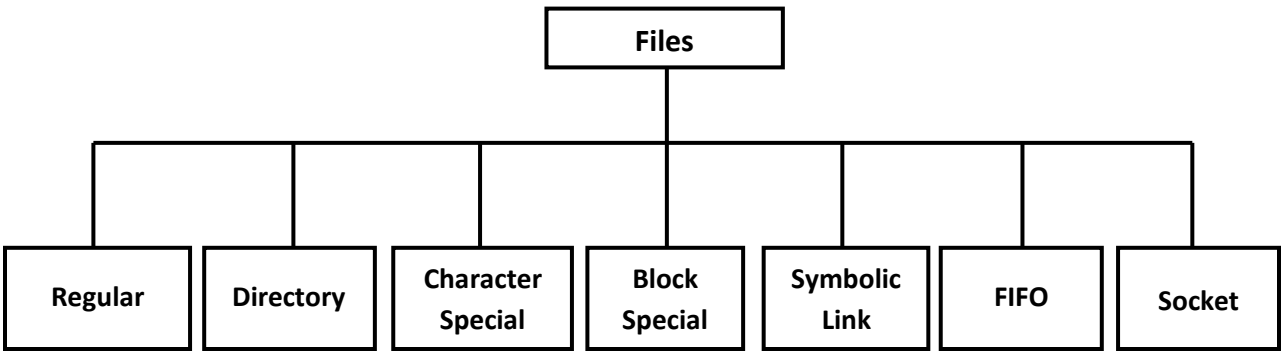
- Some implementations limit length of the name of the file to 14 characters some may allow up to 255 characters.
- A file name can be any sequence of ASCII characters. Some of the characters should be avoided as file names e.g. ">", "<" as they are used in redirection one should avoid its use in file name.
- Period(.) can be used anywhere in the file system as it has no meaning UNIX, like in other operating system, where period is used for differentiating the file name form extension.

There are some **rules** that are used to make meaningful file name in UNIX.

- Start the name with an alphabet character.
- Use dividers to separate parts of the name. Good dividers are the underscore.
- Use an extension at end of the filename, even though UNIX doesn't recognize extensions. Some applications, such as compliers require file extension. In addition, file extension helps user to classify files so that they can be easily indentified.
- Never start a file name with a period. File names that start with a period are hidden files in UNIX. Generally hidden files are created and used by the system. E.g. .profile file, .mailrc and .cshrc are some of the hidden files of UNIX.

File Types

UNIX provides seven types of files as given in the figure.



Regular Files (Ordinary Files): Regular files contain user data that need to be available for future processing. Sometimes called ordinary files, regular files are the most common files found in a system. Regular files are divided by the physical format used to store the data as text or binary. The physical format is controlled by the application program or utility that processes it.

- **Text Files**:

  A text file is a file of characters drawn from the computer's character set. UNIX computer use the ASCII character set. Because the UNIX shells treat data almost universally as strings of characters, the text file is most common UNIX file.

- **Binary Files:**

  A binary file is a collection of data stored in the internal format of the computer. In general, there are **two types** of binary files: **data files** and **program files.**

  - Data files contain application data.
  - Program files contain instructions that make a program work.

  If one tries to process binary file with a text-processing utility, the output will look very strange because it is not in a format that can be read by people.

Directory Files:A directory is a file that contains the names and locations of all files stored on physical device. The file system begins with the root directory. The root directory is denoted a forwardslash (/). There are several other directories called bin, lib, usr, tmp, etc. For each directory there is a file, by the same name as the directory. It contains information about the files and subdirectories under the directory. A directory file contains two fields for each file i.e name of each files or subdirectories under the directory and their I-node numbers. A directory file is automatically created by unix when a directory is created. Directory file cannot be modified by a user but is instead modified automatically by the system when a new file or directory is created under that directory.

In Unix, there are four special directories that play significant role in the directory structure such directories are root, home, current and parent directory.

The root directory is at the top level in the directory hierarchy. It is the root of whole file structure. It does not have parent directory. The abbreviation of root directory is the slash (/). In a UNIX environment the root directory always has several levels of

subdirectories. The root directory is the home directory of system administrator who can control the root directory.

The user is placed in **home** directory when he or she first logs in to the Unix system. It contains any files created during the session. It also contains personal system files search as user profile file and, the command history. Each user has home directory created by the system administrator at the time of creation of users account. The name of the home directory is the user login ID or user ID. The abbreviation of users home directory is the tilde (~) When we use the tilde the shell uses home directory path name implicitly by the system.

The current directory, also known as working directory of the user. A user at any point of time is located in only one directory during the session known as working directory. When the user logs into the system initially he is placed in the home directory which is the working directory of that user. The abbreviation of the working directory is dot(.)

The parent directory is immediately above the working directory. The abbreviation for the parent directory is two dots(..). All the directors in the Unix system have parent directory except the root directory.

<table>
<tr><td>f1</td><td>Dir1</td></tr>
<tr><td>Content
This is a file
It is the file for test</td><td><table>
<tr><td>f1</td><td>1234 (inode block)</td></tr>
<tr><td>f2</td><td>2345</td></tr>
<tr><td>dir2</td><td>23456</td></tr>
</table></td></tr>
</table>

Other files types (Special File Types):

- **Device files:**

  In UNIX devices are also considered to be file and saved under /dev directory. The advantage of treating the devices as files is that some of the commands used to access regular files also work with device files. I-nodes of such files do not reference to data instead they contains two numbers known as **major** and **minor**

numbers. The **major number** indicates a **device type** such as terminal and the **minor** number indicates the **unit number (instance of)** of the device.

The device files can be categorized in two types as given below.

- **Character Special File:** A character special file represents a physical device, such as terminals and network media that reads or writes one character at a time. These files are named as /dev/rdsk
- **Block Special Files:** A block special file represents a physical device, such as a disk and tapes, that reads or write data a block at time. These files are named as /dev/dsk

One of the important things about device is that they are no file sizes. They are included with major device number and minor device number. The major number represents the device driver that is actually the type of device being referred to all. The hard disk have same major number if they are attached to same controller. The minor number is parameter that the kernel passes to the device driver. In other words is it indicates the possible instances of the same device or it indicate the special characteristics of device. For example fd135ds18 and fd196ds15 represents two devices attached to a particular controller (for FD device). So both of them have the same major number but different minor numbers.

- **Symbolic Link Files:**

  A symbolic link is a logical file that defines the location of another file somewhere else in the system.

- **FIFO Files:**

  A first-in, first-out, also known as a named pipe, is a file that is used for inter-process communication.

  ls |wc

- **Socket:**

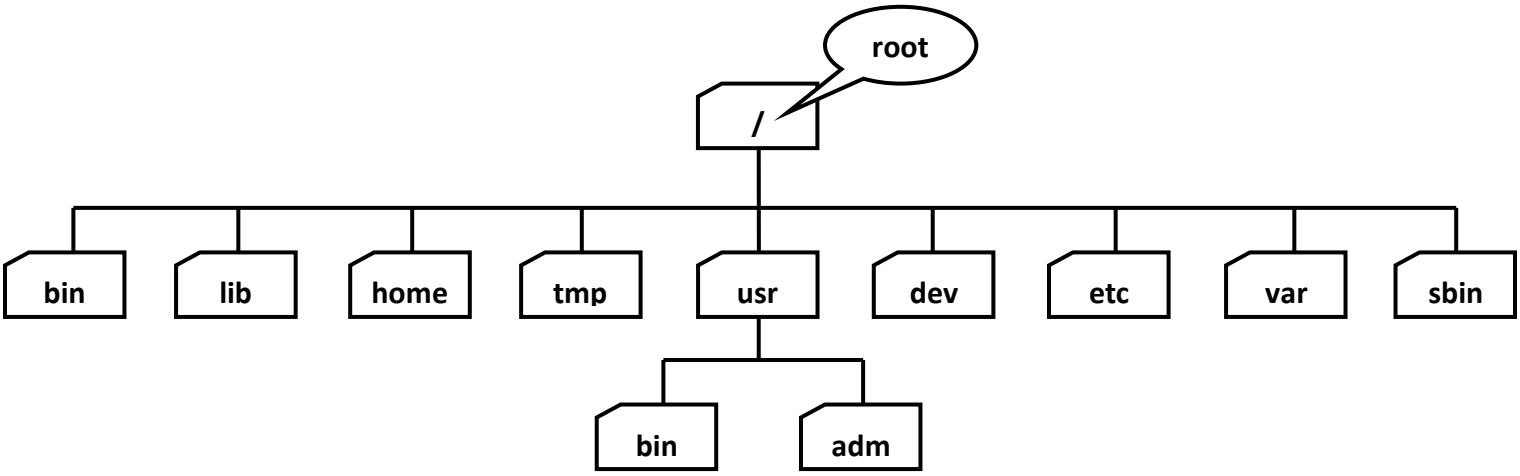
  A socket is a special file that is used for network communication. There are two types of socket file:

  1. Unix domain socket for IPC : They can be used for processes running locally on the same Unix system and communicate to each other.
  2. Unix internet socket : It supports protocol which allows you to connect processes between different Unix system.

The UNIX File System Tree Or Directory Structure of Unix File system.

The Unix file system has a hierarchical structure. The file system begins with the root directory. The root directory is denoted by a slash (/). There are several other directories called bin, lib, usr, tmp, dev, etc.

The root directory also contains a UNIX file which is UNIX kernel itself. These directories are called subdirectories, there parent being root directory. Following shows the basic structure of UNIX file system.



- **/bin directory:** It contains essential UNIX utilities that all users will have to use at some time. The word bin stands for the binary; hence this directory contains binary files. Binary files are executable files. The common majority of UNIX utilities are all binary files. Although some utilities are shell scripts. Shell scripts are also executable but they are not binary files. E.g. we have commands like , cat, who, date stored in bin directory
- **/dev directory:** It contains files for all devices that are connected with UNIX server. UNIX treats devices (such as printer, disk storage, terminals and even areas of computer memory) as a file. The files in dev directory are termed as special files neither directory nor ordinary files.I-nodes of such files do not reference to data instead they contain two numbers known as major and minor numbers. The major number indicates a device type such as terminal and the minor number indicates the unit number (instance of) of the device**. *(can write about device file type also, see topic Device Files under TYPES OF FILES)**
- **/etc directory:** etc directory contains various administration utilities together with other special system files that allow the host UNIX system (server) to startup properly at bootstrap time.

The files such as /etc/inittab, /etc/gettydef, /etc/profile,/etc/passwd and /etc/motd are found in the etc directory. Message of the file is displayed at the login time & is useful for displaying important messages to user.

It contains several system files which store the relevant information about the user of the system and the terminal and devices connected to the system.

- **/tmp directory:** tmp directory contains temporary files created by the UNIX. These files automatically get deleted when the system is shut down or restarted.
- **/usr directory:** In the usr directory the user work area is provided. There are several directories each associated with a particular user. The system administrator creates these directories when he creates account for different users. Each user is allowed to work with this directory known as home directory.
  - **/usr/bin directory**: Within the usr directory there is another directory which contains additional UNIX command files that are more important to end users.
  - **/usr/adm directory:** adm stands for administrator; hence adm directory contains administrative utilities used by system administrator. Although most users will not be able to access them.
- **/home directory:** On many systems are housed here. If any user wants to create his own home directory he create user directory in this directory.
- **/lib directory:** The lib directory contains all the library functions provided by UNIX for programmers. When programs written under UNIX need to make system calls, they can use libraries provided in this directory.
- **/sbin directory:** There are some commands which only the system administrator can execute, a user can not execute such commands are stored under this directory.
- **/var:** The variable part of the files system. Contains all the print jobs and outgoing and incoming mails.

Features of UNIX File System

- **Hierarchical file system:**

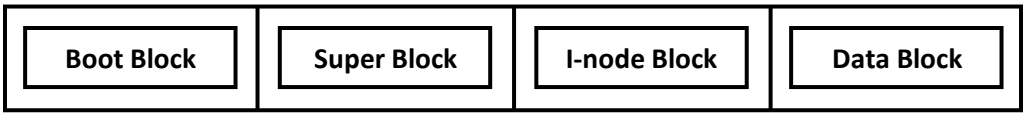
  The UNIX system organizes its files using an upside-down hierarchical tree structure. All files will have a parent file, apart from a directory called the root directory (/), which is the parent of all fileson the system. This hierarchical component also adds to the dynamic flexibility of the file system.
- **Access permissions:**

The files are protected using file ownership mechanism. This allows only a specific class of user to access certain files. Files have read, write & execute permissions. These permissions can be given to owner (or user), group and others.

- **All devices are implemented as files:**
  All the hardware devices such as printers, terminals, disk drives are considered to be files in UNIX operating system.
- **Files can grow dynamically:**
  The files system structure is dynamic i.e. its size is not determined by any rule other than amount of disk storage that it is available on the system. A file can be changed by the user at any time.
- **Structure less files:** (It contains sequence of bytes in a file)
  It has not specific structure. All the files are considered as collection of stream of bytes.

Physical File Structure of UNIX (UNIX File system Structure)/File system components

A file system is a group of files and relevant information regarding them. Disk space is allotted to UNIX file system is made up of blocks each of them is of 512 bytes. Some file system may have blocks of 1024 or 2048 bytes as well. All the blocks belonging to the file system are logically divided into four parts as given below:



BOOT BLOCK: Boot block (or Block-0) is the first block of the file system. This block is normally reserved for booting process. It is also known as MBR (Master Boot Record). The Boot block contains small boot strapping program. when the system is booted the system BIOS checks for existence of hard disk and loads the entire code of boot block in to the memory. It then handovers control to the boot strapping program. This program loads the kernel in to memory. Though all the files and directories have boot block, the bootstrap program of the boot block of root file system is loaded in to memory. For other file systems, this block is simply kept empty.

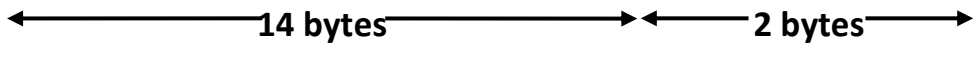
SUPER BLOCK: The super block (or Block-1) is the balance sheet of every UNIX file system. It contains the information about disk usage, availability of data block and I-nodes. It also contains the total size of the file system, logical blocks, the last time of updating, length of the file system, no. of free data block available and a partial list of immediately allocable free data blocks, no. of free I-nodes available and partial list of immediately usable I-nodes, the state of file system (clear or dirty). It also stores the information of bad blocks.

I-NODE BLOCK (Index/Information Node block): In UNIX all the entities are considered to be files. I-Nodes are used to describe the file characteristic & the location of data block, which store the contents of the file. The information related to all these files (not the contents) is stored in an 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 byte and contains the relevant details are:

- File Type (ordinary, directory, device file)
- Number of links (No. of alias)
- The numeric user-id of the owner.
- The numeric group-id (GUID - Global User Id) of the owner.
- File mode (access permission)
- Size of file.
- Date and time of last modification of file data.
- Date and time of last access of the file data.
- Date and time of last change of I-node.
- Address of blocks where the file is physically present.

I-node is accessed by a number known as I-node number. This number is unique for every file in the file system.

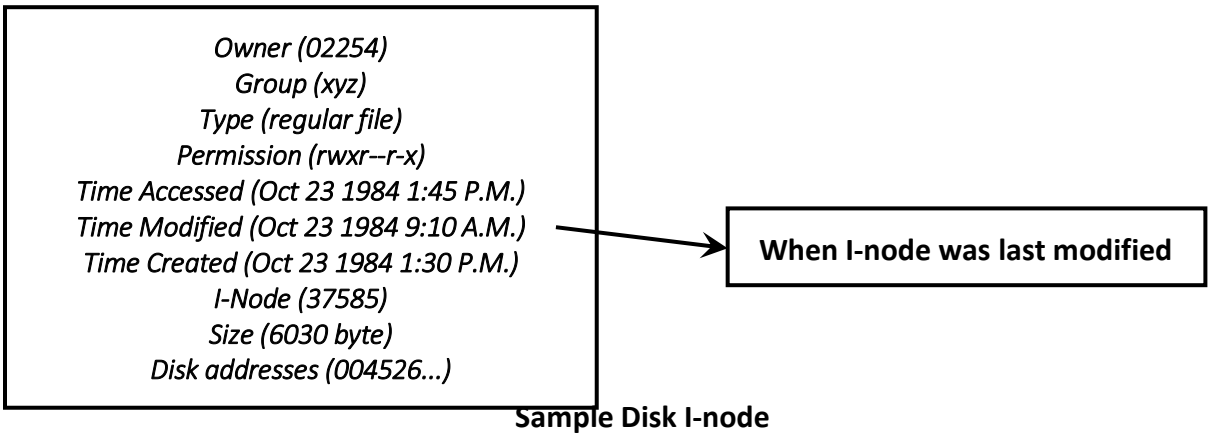
DATA BLOCK: The remaining space of the file system is taken up by data block or storage block. This blocks store the contents of the file in case of regular file.
For a directory file each data block contains 16 byte entries. Each such entry would have a file or subdirectory name up to 14 bytes and 2 bytes will be taken for I-node. Thus for directory the table of file-names & there I-nodes are available.
E.g. Directory name is /usr



File name 1	I-node 1
File name 2	I-node 2
‘	‘
‘	‘

I-Node

In UNIX all the entities are considered to be files. I-nodes are used to describe the file characteristic & the location of data block, which store the contents of the file. The information related to all these files (not the contents) is stored in an I-node table on the disk.



Sample Disk I-node

For each file there is an I-node entry in the table. Each entry is made up of 64 byte and contains the relevant details are:

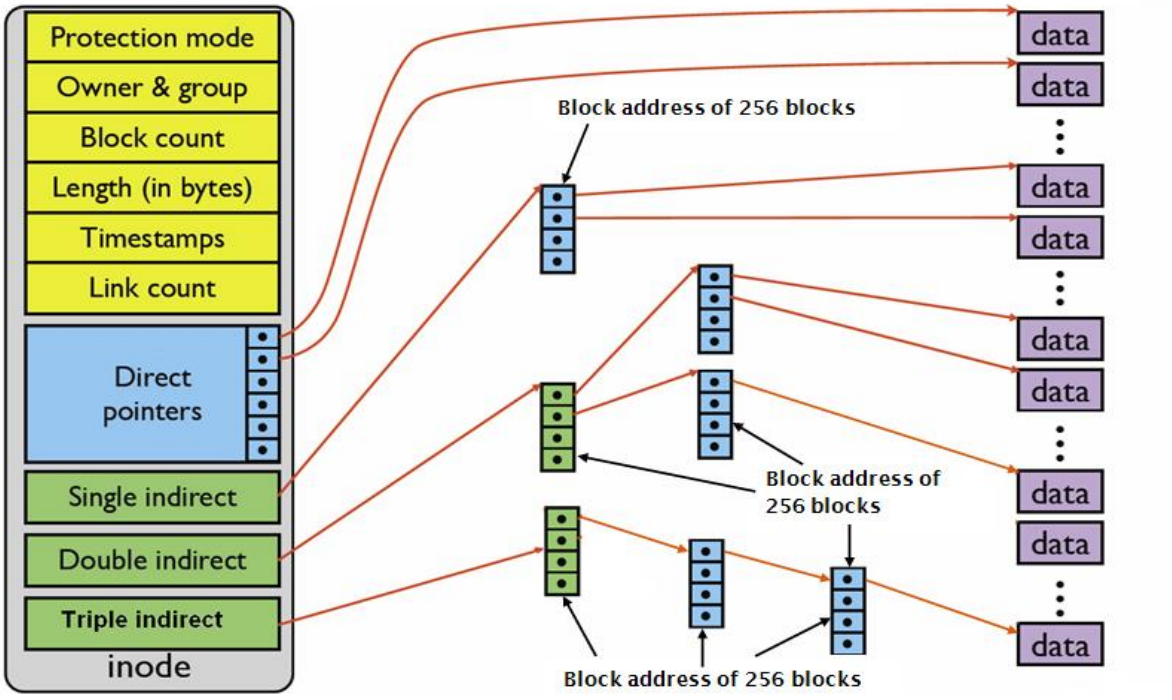
- File Type (ordinary, directory, device file)
- Number of links (No. of alias)
- The numeric user-id of the owner.
- The numeric group-id of the owner.
- File mode(access permission)
- Size of file.
- Date and time of last modification of file data.
- Date and time of last access of the file data.
- Date and time of last change of I-node.
- Address of blocks where the file is physically present.

I-node is accessed by a number known as I-node number. This number is unique for every file in the file system.I-node exists in a static form on disk and the kernel reads them into in-core I-node to manipulate them.

The content of the I-node changes when any modification is done in file (like contents of file, changing owner, group, permissions etc). ***[**The in-core(is I-node created by kernel-it is created when the file is create with disk I-node) I-node has some extra information like the status of I-node, logical device no. of file system that contains the file, the I-node number, pointers to other in-core I-nodes. **]* Topic just for information**

How I-node access data Block addressing scheme:

As per the **Figure**,…



- I-nodes have an array of 13 disk block addresses. I-nodes use these 13 pointers to Access data blocks related to file. The first **10 pointers are direct addresses** i.e. they point to the first 10 blocks of the file that particular I-node is representing.
- **Each logical block of UNIX has 1024 bytes (1KB)**, these 10 pointers allows access to files up to a maximum of 10,240 bytes (10 KB). So whenever a **file of less than 10 KB**

is stored on the file system, the I-node representing that **file uses its first 10 pointers.**

- When file is more than 10 KB then its **11th pointer** indirectly accesses 256 blocks, each of which addresses points to 1 KB. Thus, the maximum file size that can be address using 11th pointer entry is 256 KB. This is called **single indirection.**
- Similarly, **12th pointer** points to a block which contains the addresses of 256 blocks each of which has addresses of another 256 blocks thus the maximum file size that can be addressed using 12th pointer entry is 256 KB * 256 KB (=65536 KB or 65 MB). This is called **double indirection.**
- and the **13th pointer** contains address of block that contains addresses of 256th blocks each of which contains the address of 256 blocks & each of which contains addresses of 256 blocks. I.e.: 256 KB* 256 KB* 256 KB (approx. 16 GB). This is the maximum file size that can be addressed using 13 pointer is 16 GB. This is called **Triple Indirection.**
- Thus, the **maximum file size** that can be accessed by UNIX system is 10 KB + 256 KB + 65 MB + 16 GB = (approx.) 17 GB.

Booting Sequence (or Stages of System Startup or Booting Process)

- **ROM Diagnostics are run:** The hardware and memory tests are performed.
- **Boot Loader is loaded**: The ROM BIOS reads the boot loader from boot block and loads it in RAM
- **Kernel is loaded into main memory:** After the boot loader is loaded in memory it loads the kernel in memory. The kernel is the part of the UNIX operating system that remains running at all times when the system is up. The kernel is loaded into memory by bootstrap program. Now the control is handed over to kernel.
- **Kernel initialization takes place:** It involves performing memory tests, initializing the devices through device drivers, swapper scheduler process, the init process and etc
- **Init program is invoked:** The init program resides in /etc directory which is invoked by the kernel. This program takes over the system control.
- **The init program:** This process has the PID (Process Identification) 1 which is the second process of the system. It uses the inittab file to perform various tasks like, identifying run level, maintaininthg etc.
- **The getty process:** The init program also invokes the getty process ,it establishes the communication with terminals of the unix system. It prints the login prompt on terminal

and then moves to sleep mode and is activated when the user tries to login. It uses file /etc/gettydefs for instructions to provide login.

- **The Login program:** When the user types the login name and password the getty program transfers control to login program. The login program verifies the user name and password, if it is correct the shell prompt is displayed on terminal.

The process sequence after kernel invokes init can be shown as given below:

Init->getty->login->shell

When the user logs out the shell is killed and init wakes up again and invokes getty for next session.

Startup Processes summary

Function	Program	Files involved
It identifies run level (single,multiuser system),maintains files, **invokes getty**	Init (it reads inittab)	/etc/inittab
It provides login prompt ,accepts login name and password ,invokes login program	Getty	/etc/gettydefs
It compares login name and password validity, runs /etc/profile file program, .profile program	Login	/etc/passwd /etc/shadow .profile

Passwd file

- It is stored in /etc directory
- All the info except the encrypted password is stored in passwd file.
- The encrypted password is stored in /etc/shadow file
- It has entries regarding of the registered users
- There are seven fields in the file, each of this field is separated by colon(:)\

The structure of passwd file is as given below:

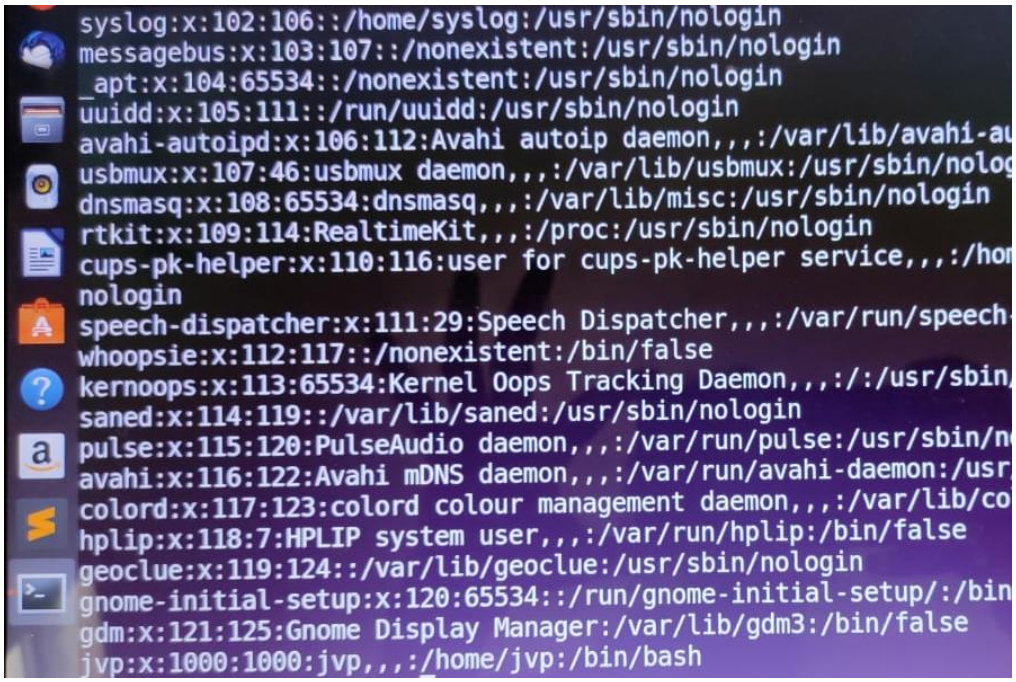
UserName: password: UID: GID: Comment :home directory :login shell

The information of these seven fields are:

1. User name-it's the login name of the user
2. Password-Encrypted password of the user, but it contains the "x". (The encrypted password is stored in shadow file)
3. UID: User Identification number (numeric UID)
4. GID: Group Identification number (numeric GID)
5. Comment : It stores detailed information about the user like, fullname of the user, address etc
6. Home directory: It contains the path of users home directory.
7. Shell: This is user's login shell(default shell)

e.g. **tybca01:x:1000:1000:tybca01 student:/home/tybca01:/bin/bash**

The following is the content of passwd file (Note: in exam only one or two examples should be given as in e.g., don't show as in the image it is just for more clarity)



Shadow file

- It is stored under /etc directory.
- The actual password is stored in this file.
- It is stored in an encrypted form.
- It contains 9 fields.

The structure of shadow file is as given below:

Username :pwd:last pwd change:minimum:maximum:warn:Inactive:Exprie:reserve

The information of these seven fields are:

Uername: It is the login name of the user

pwd : it contains encrypted password. It should be 6-8 characters long including special characters/digits

last pwd change : Days since Jan1, 1970 that password was last changed

minimum : The number of days left before the user is allowed to change the password or minimum days before password may be changed

e.g. if the value here is 0(zero) means user can change the password as and when he needs. But if the value is set to 2; he cannot change password till two days.

maximum : The maximum number of days the password is valid or days after which the password must be changed.

e.g. If the value is set to 2; the user is required to change the password after 2 days. To use the password for long time here we have to provide more number of days.

warn : The number of days before the password is to be expire ,the user is warned that the password must be changed.

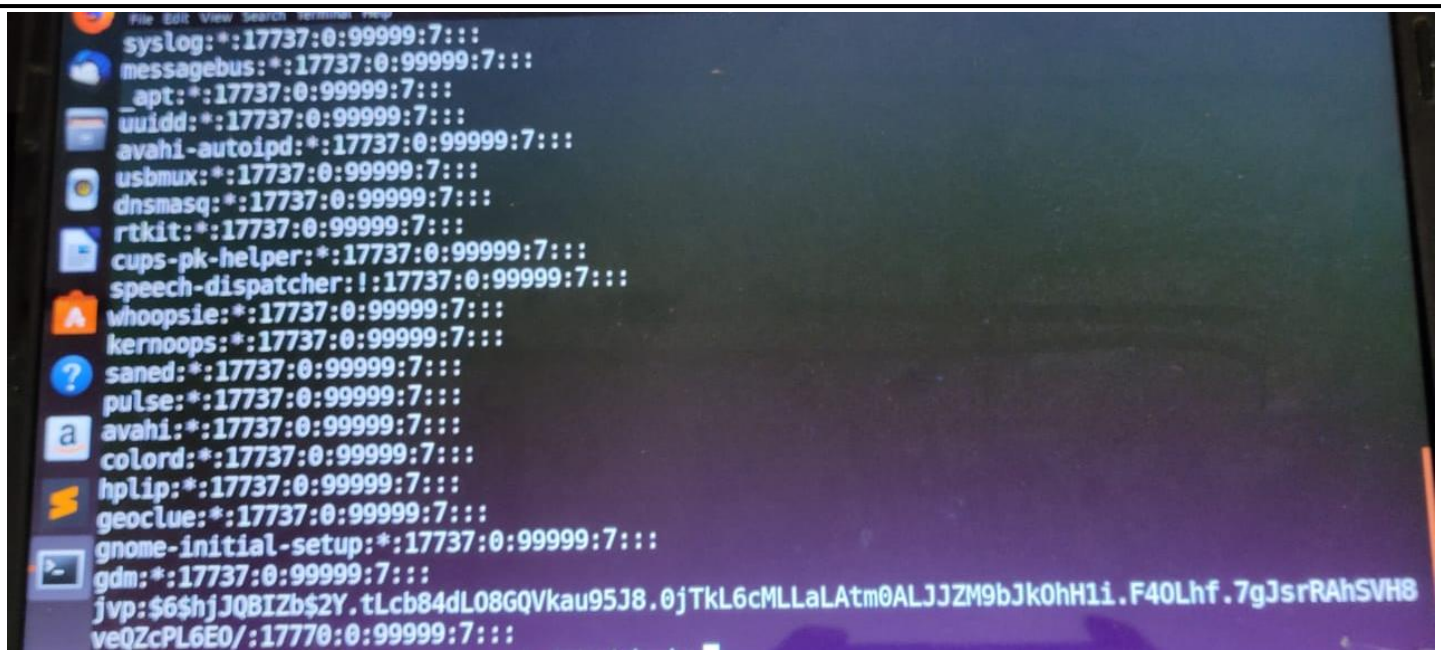
Inactive : The number of days after password expires that account is disable

Expire : days since 1st Jan 1970 that account is disabled i.e an absolute date specifying when the login may no longer be used

Reserve : It is a reserved field

e.g. **tybca01:abec.10.0abnd:17770:0:99999:7:::**

The following is the content of shadow file (Note: in exam only one or two examples should be given as in e.g., don't show as in the image it is just for more clarity)



Internal & External Commands

- A command built into the shell is known as ***Internal Commands.***

The shell doesn't create a separate process to run internal commands. **E.g.:** cd, echo, pwd, etc…

cd & **echo** doesn't generate process & execute directly by shell.

- A command stored in /bin directory known as ***External Commands.***

External commands require the shell to create a new process. **E.g.:** cat, date, ls, etc…

Sticky Bits

The most common use of the **sticky bit** today is on *directories*. When the sticky bit is set, only the item's owner, the directory's owner, or the super-user can rename or delete files. Without the sticky bit set, any user with write and execute permissions for the directory can rename or delete contained files, regardless of owner.

e.g.

The sticky bit can be set using the chmod command and can be set using its octal mode 1000 or by its symbol t (s is already used by the setuid bit).

For example, to add the bit on the directory /usr/local/tmp,

One would type **chmod +t /usr/local/tmp.**

Or, to make sure that directory has standard tmp permissions;

One could also type **chmod 1777 /usr/local/tmp.**

- In UNIX symbolic file system permission notation, the sticky bit is represented by the letter t in the final character-place.

The /tmp directory, which by default has the sticky-bit set, shows up as:

```
$ ls -ld /tmp
drwxrwxrwt   4 root     sys         485 Nov 10 06:01 /tmp
```

File Access Permission

In Unix all files and directories have permissions. There are three file level and directory level permissions as given below:

Read : r

Write : w

Execute : x

The file permissions are assigned numerical octal values from 0 to 7. The values assigned individually are:

Read: 4

Write: 2

Execute: 1

It can be said that the permissions can be assigned and displayed using symbolic and octal method.

e.g a file displaying permission string like, -rw-rw-rw- has read and write permission for all the three users (owner, group and others). In octal notation it can be written as 666.

Following table shows various permissions in symbolic as well as octal notation:

Permission	Symbolic code(r,w,x)	Octal value (0-7)

No permission	-	0
Execute	X	**1**
Write	W	2
Write & Execute	Wx	3 (2+1)
Read	r	4
Read ,Execute	Rx	5(4+1)
Read Write	Rw	6(4+2)
All (r,w,x)	Rwx	7