BCA 306: Practical on Linux.

w.e.f. 2018-19
Total Lectures: 60

[Total Marks: 60 External + 40 Internal =100 Marks]

1. Access: Logging In. Linux Commands. Getting Help. Obtaining Information about Your System.
2. Starting and Stopping Linux: Shutting Down a Linux System, Booting a Linux System.
3. Demonstration of Linux commands with attributes: - pwd, cd, ls, more, less, echo, clear, kill, ps, man, cal, date, who, who am I, WC, mkdir, rmdir, rm, sort.
4. File and File Permission: Creation of Files, and changing their permission (Cat,vi, Chmod)
5. Archiving Files: Archiving Files with tar
6. Write a shell script to display first 20 terms of Fibonacci series.
7. Write a shell script to display current time of system and display the message according to the time.
8. Write a shell script to check the user is login or not and say hello.
9. Write a shell script to calculate factorial of a number
10. Using filters & redirections: create new processed files (Using Head, tail, cut, paste etc. create resultsheet/salarysheet)
11. Develop a C Program In Linux to find out 20 terms of Fibonacci series.
12. Develop a C Program In Linux to calculate factorial of a number

1. Access: Logging In. Linux Commands. Getting Help. Obtaining Information about Your System.

What is Linux ?

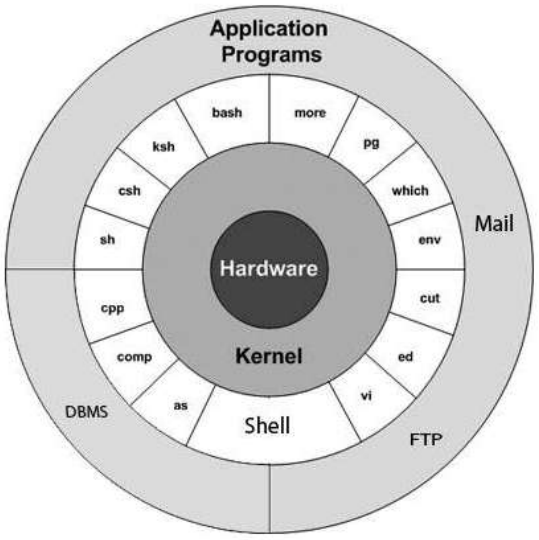
The Linux operating system is a set of programs that act as a link between the computer and the user. The computer programs that allocate the system resources and coordinate all the details of the computer's internals is called the **operating system** or the **kernel**.

Users communicate with the kernel through a program known as the **shell**. The shell is a command line interpreter; it translates commands entered by the user and converts them into a language that is understood by the kernel.

- Linux is also a flavor of Unix which is freely available.
- Several people can use a Linux computer at the same time; hence Linux is called a multiuser system.
- A user can also run multiple programs at the same time; hence Linux is a multitasking environment.

Linux Architecture

Here is a basic block diagram of a Linux system − The main concept that unites all the versions of Unix is the following four basics −



- **Kernel** − The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.
- **Shell** − The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.
- **Commands and Utilities** − There are various commands and utilities which you can make use of in your day to day activities. **cp**, **mv**, **cat** and **grep**, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.
- **Files and Directories** − All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the **filesystem**

System Bootup

If you have a computer which has the Unix operating system installed in it, then you simply need to turn on the system to make it live.

As soon as you turn on the system, it starts booting up and finally it prompts you to log into the system, which is an activity to log into the system and use it for your day-to-day activities.

Login Unix

When you first connect to a Unix system, you usually see a prompt such as the following −

```
login:
```

To log in

- Have your userid (user identification) and password ready. Contact your system administrator if you don't have these yet.
- Type your userid at the login prompt, then press **ENTER**. Your userid is **case-sensitive**, so be sure you type it exactly as your system administrator has instructed.
- Type your password at the password prompt, then press **ENTER**. Your password is also case-sensitive.

- If you provide the correct userid and password, then you will be allowed to enter into the system. Read the information and messages that comes up on the screen, which is as follows.

```
login : student
student's password:________ [#student or zbpc@123 ]
Last login: Thu Aug 30 08:32:32 2018 from 192.168.1.100
$
```

<u>You will be provided with a command prompt (sometime called the $ prompt) where you type all your commands.</u>

Logging Out

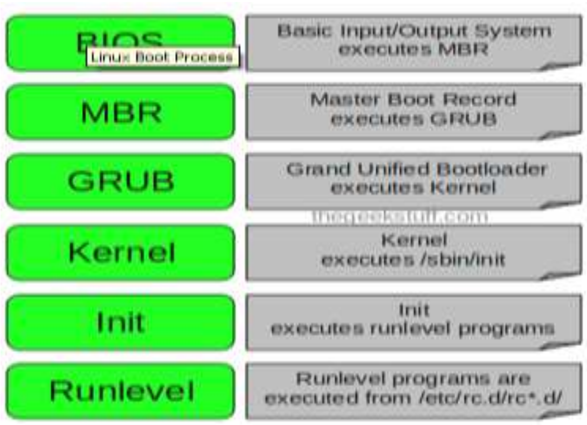
When you finish your session, you need to log out of the system. This is to ensure that nobody else accesses your files.

To log out

- Just type the **logout** command at the command prompt, and the system will clean up everything and break the connection.

2. **Starting and Stopping Linux:** Booting a Linux System , Shutting Down a Linux System.

Linux System Booting (Startup Sequence)


BIOS
Linux Boot Process
Basic Input/Output System executes MBR
MBR
Master Boot Record executes GRUB
GRUB
Grand Unified Bootloader executes Kernel
thegeekstuff.com
Kernel
Kernel executes /sbin/init
Init
Init executes runlevel programs
Runlevel
Runlevel programs are executed from /etc/rc.d/rc*.d/

Press the power button on your system, and after few moments you see the Linux login prompt.

Have you ever wondered what happens behind the scenes from the time you press the power button until the Linux login prompt appears?

The following are the 6 high level stages of a typical Linux boot process.

1. BIOS

- BIOS stands for Basic Input/Output System
- Performs some system integrity checks
- Searches, loads, and executes the boot loader program.
- It looks for boot loader in floppy, cd-rom, or hard drive. You can press a key (typically F12 of F2, but it depends on your system) during the BIOS startup to change the boot sequence.
- Once the boot loader program is detected and loaded into the memory, BIOS gives the control to it.
- So, in simple terms BIOS loads and executes the MBR boot loader.

2. MBR

- MBR stands for Master Boot Record.
- It is located in the 1st sector of the bootable disk. Typically /dev/hda, or /dev/sda
- MBR is less than 512 bytes in size. This has three components 1) primary boot loader info in 1st 446 bytes 2) partition table info in next 64 bytes 3) mbr validation check in last 2 bytes.
- It contains information about GRUB (or LILO in old systems).
- So, in simple terms MBR loads and executes the GRUB boot loader.

3. GRUB

- GRUB stands for Grand Unified Bootloader.
- If you have multiple kernel images installed on your system, you can choose which one to be executed.
- GRUB displays a splash screen, waits for few seconds, if you don't enter anything, it loads the default kernel image as specified in the grub configuration file.
- GRUB has the knowledge of the filesystem (the older Linux loader LILO didn't understand filesystem).
- Grub configuration file is /boot/grub/grub.conf (/etc/grub.conf is a link to this). The following is sample grub.conf of CentOS.

```
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
hiddenmenu
title CentOS (2.6.18-194.el5PAE)
        root (hd0,0)
        kernel /boot/vmlinuz-2.6.18-194.el5PAE ro root=LABEL=/
        initrd /boot/initrd-2.6.18-194.el5PAE.img
```

- As you notice from the above info, it contains kernel and initrd image.
- So, in simple terms GRUB just loads and executes Kernel and initrd images.

4. Kernel

- Mounts the root file system as specified in the "root=" in grub.conf
- Kernel executes the /sbin/init program
- Since init was the 1st program to be executed by Linux Kernel, it has the process id (PID) of 1. Do a 'ps -ef | grep init' and check the pid.
- initrd stands for Initial RAM Disk.
- initrd is used by kernel as temporary root file system until kernel is booted and the real root file system is mounted. It also contains necessary drivers compiled inside, which helps it to access the hard drive partitions, and other hardware.

5. Init

- Looks at the /etc/inittab file to decide the Linux run level.
- Following are the available run levels
  - 0 – halt
  - 1 – Single user mode
  - 2 – Multiuser, without NFS
  - 3 – Full multiuser mode
  - 4 – unused
  - 5 – X11
  - 6 – reboot
- Init identifies the default initlevel from /etc/inittab and uses that to load all appropriate program.
- Execute 'grep initdefault /etc/inittab' on your system to identify the default run level
- If you want to get into trouble, you can set the default run level to 0 or 6. Since you know what 0 and 6 means, probably you might not do that.
- Typically you would set the default run level to either 3 or 5.

6. Runlevel programs

- When the Linux system is booting up, you might see various services getting started. For example, it might say "starting sendmail …. OK". Those are the runlevel programs, executed from the run level directory as defined by your run level.
- Depending on your default init level setting, the system will execute the programs from one of the following directories.
  - Run level 0 – /etc/rc.d/rc0.d/
  - Run level 1 – /etc/rc.d/rc1.d/
  - Run level 2 – /etc/rc.d/rc2.d/
  - Run level 3 – /etc/rc.d/rc3.d/
  - Run level 4 – /etc/rc.d/rc4.d/
  - Run level 5 – /etc/rc.d/rc5.d/
  - Run level 6 – /etc/rc.d/rc6.d/
- Please note that there are also symbolic links available for these directory under /etc directly. So, /etc/rc0.d is linked to /etc/rc.d/rc0.d.
- Under the /etc/rc.d/rc*.d/ directories, you would see programs that start with S and K.
- Programs starts with S are used during startup. S for startup.
- Programs starts with K are used during shutdown. K for kill.
- There are numbers right next to S and K in the program names. Those are the sequence number in which the programs should be started or killed.
- For example, S12syslog is to start the syslog deamon, which has the sequence number of 12. S80sendmail is to start the sendmail daemon, which has the sequence number of 80. So, syslog program will be started before sendmail.

System Shutdown

The most consistent way to shut down a Unix system properly via the command line is to use one of the following commands −

Sr.No.	Command & Description
1	**Halt :** Brings the system down immediately
2	**init 0 :** Powers off the system using predefined scripts to synchronize and clean up the system prior to shutting down
3	**init 6 :** Reboots the system by shutting it down completely and then restarting it
4	**Poweroff :** Shuts down the system by powering off
5	**Reboot :** Reboots the system
6	**Shutdown :** Shuts down the system

You typically need to be the super user or root (the most privileged account on a Unix system) to shut down the system. However, on some standalone or personally-owned Unix boxes, an administrative user and sometimes regular users can do so.

3. **File and File Permission**: Creation of Files, and changing their permission (Cat,vi, Chmod)

chmod -

NAME

chmod - To change access permissions, change mode.

SYNOPSIS

```
chmod [Options]... Mode [,Mode]... file...
chmod [Options]... Numeric_Mode file...
chmod [Options]... --reference=RFile file...
```

DESCRIPTION

chmod changes the permissions of each given file according to mode, where mode describes the permissions to modify. Mode can be specified with octal numbers or with letters.

OPTIONS

Tag	Description
-f, --silent, --quiet	suppress most error messages
-v, --verbose	output a diagnostic for every file processed
-c, --changes	like verbose but report only when a change is made
-c, --reference=RFile	use RFile's mode instead of MODE values
-R, --recursive	change files and directories recursively
--help	display help and exit
--version	output version information and exit

Numeric mode

The format of a numberic mode is 'augo'

A numeric mode is from one to four octal digits (0-7), derived by adding up the bits with values 4, 2, and 1. Any omitted digits are assumed to be leading zeros. The first digit selects the set user ID (4) and set group ID (2) and sticky (1) attributes. The second digit selects permissions for the user who owns the file: read (4), write (2), and execute (1); the third selects permissions for other users in the file's group, with the same values; and the fourth for other users not in the file's group, with the same values.

EXAMPLES

Read by owner only
```
$ chmod 400 sample.txt
```
Read by group only
```
$ chmod 040 sample.txt
```
Read by anyone
```
$ chmod 004 sample.txt
```
Write by owner only
```
$ chmod 200 sample.txt
```
Write by group only
```
$ chmod 020 sample.txt
```
Write by anyone
```
$ chmod 002 sample.txt
```
Execute by owner only
```
$ chmod 100 sample.txt
```
Execute by group only
```
$ chmod 010 sample.txt
```
Execute by anyone
```
$ chmod 001 sample.txt
```
Allow read permission to owner and group and anyone.
```
$ chmod 444 sample.txt
```
Allow everyone to read, write, and execute file.
```
$ chmod 777 sample.txt
```

Symbolic mode

The format of a symbolic mode is '[ugoa...][[+-=][rwxXstugo...]...][,...]'. Multiple symbolic operations can be given, separated by commas. A combination of the letters 'ugoa' controls which users' access to the file will be changed: the user who owns it (**u**), other users in the file's group (**g**), other users not in the file's group (**o**), or all users (**a**). If none of these are given, the effect is as if 'a' were given, but bits that are set in the umask are not affected.

The operator '+' causes the permissions selected to be added to the existing permissions of each file; '-' causes them to be removed; and '=' causes them to be the only permissions that the file has.

The letters 'rwxXstugo' select the new permissions for the affected users: read (r), write (w), execute (or access for directories) (x), execute only if the file is a directory or already has execute permission for some user (X), set user or group ID on execution (s), sticky (t), the permissions granted to the user who owns the file (u), the permissions granted to other users who are members of the file's group (g), and the permissions granted to users that are in neither of the two preceding categories (o).

EXAMPLES

Allow read permission to everyone.

```
$ chmod a+r sample.txt
```

Make a file readable and writable by the group and others.

```
$ chmod go+rw sample.txt
```

Make Deny execute permission to everyone.

```
$ chmod a-x sample.txt
```

a shell script executable by the user/owner.

```
$ chmod u+x samplescript.sh
```

Allow everyone to read, write, and execute the file and turn on the set group-ID.

```
$ chmod =rwx,g+s samplescript.sh
```

Permission(s)	Owner	Group	OtherUser
Read	4	4	4
Write	2	2	2
Execute	1	1	1
Sum=> allow to all	**7**	**7**	**7**
Deny to all rwx	**0**	**0**	**0**

Permission's	Owner (u)	Group (g)	OtherUser (o)	All (a)
Read (r)				
Write (w)				
Execute (x)				
+ grant permission		- Deny permission		
Ex.1.	$ chmod u+w sample.txt			
Ex.2	$ chmod ug+rw sample.txt			
Ex.3	$ chmod ugo+r sample.txt			
Ex.4	$ chmod go+rw sample.txt			
Ex.5	$ chmod ugo+rwx sample.txt			
Ex.6	$ chmod a+rwx			

4. Demonstration of Linux commands with attributes: - pwd, cd, ls, more, less, echo, clear, kill, ps, man, cal, date, who, who am i, wc, mkdir, rmdir, rm, sort, tar,

```
[student@localhost student]$ pwd
        /home/student
[student@localhost student]$ cd sshah
[student@localhost sshah]$ pwd
        /home/student/sshah
[student@localhost sshah]$ echo "This is test"
        This is test
[student@localhost sshah]$ cal
                  September 2018
                Su Mo Tu We Th Fr Sa
                                  1
                 2  3  4  5  6  7  8
                 9 10 11 12 13 14 15
                16 17 18 19 20 21 22
                23 24 25 26 27 28 29
                30
[student@localhost sshah]$ date
        Wed Sep 19 09:29:52 IST 2018
[student@localhost sshah]$ wc sydata1819
           47     232    1332 sydata1819
[student@localhost sshah]$ who am i
        student  pts/12       Sep 19 08:48 (192.168.1.91)
[student@localhost sshah]$ who
        root     :0           Jan  1 00:05
        student  pts/1        Sep 19 07:50 (192.168.1.85)
        student  pts/6        Sep 19 08:13 (192.168.1.87)
        student  pts/3        Sep 19 08:13 (192.168.1.83)
        student  pts/7        Sep 19 08:15 (192.168.1.88)
        student  pts/10       Sep 19 08:21 (192.168.1.74)
        student  pts/0        Sep 19 08:28 (192.168.1.86)
        student  pts/12       Sep 19 08:48 (192.168.1.91)
        student  pts/5        Sep 19 08:58 (192.168.1.89)
        student  pts/8        Sep 19 09:08 (192.168.1.90)
        student  pts/4        Sep 19 09:09 (192.168.1.84)
        student  pts/9        Sep 19 09:09 (192.168.1.95)
        student  pts/11       Sep 19 09:29 (192.168.1.97)
        student  pts/13       Sep 19 09:31 (192.168.1.97)
 [student@localhost sshah]$ ps
          PID TTY          TIME CMD
        11354 pts/12   00:00:00 bash
        25898 pts/12   00:00:00 ps
[student@localhost sshah]$ sort sydata1819
        01 Ahire Bhagyashri Nandkumar SY
        02 Bacchav Sandip Sanjay SY
        03 Bagul Tushar Dnyaneshwar SY
        04 Bhamare Suyog Kishor SY
        05 Borase Pooja Pradip SY
        06 Borase Sandip Rajendra SY
        07 Chaudhari Bharat Ashok SY
        08 Chaudhari Kapil Nagindas SY
        09 Chaudhari Prem Vijay SY
        10 CHavan Dnyaneshwar Makram SY
        11 Chavan Sagar Dilip SY
        12 Chavan Sandip Papalal SY
        13 Chhetiya Heena Mohanlal SY
        14 Darji Vaibhav Mukesh SY
```

6. Write a **shell script** to calculate factorial of a number

[student@localhost sshah]$ cat fact

```
# [fact] Assignment Script
# Shell Script to find factorial of given number.

echo "|-----------------------|"
echo "|  Factorial Number :   |"
echo "|-----------------------|"
echo -n "| Enter Number - "
read n
echo "|-----------------------|"
f=1
d=$n
while [ $d -ge 1 ]
do
        f=`expr $f \* $d`
        d=`expr $d - 1`
done
echo " Factorial of $n is $f "
echo "| ----- That's All -----|"
echo "|-----------------------|"
```

[student@localhost sshah]$ sh fact

```
|-----------------------|
|  Factorial Number :   |
|-----------------------|
| Enter Number - 6
|-----------------------|
 Factorial of 6 is 720
| ----- That's All -----|
```

5. Write a **shell script** to display first 20 terms of Fibonacci series.

```
[student@localhost sshah]$ cat fibb
# [fibb] Assignment Script
# Shell Script to print fibbonnacci series upto given number of terms
                echo "|-------------------|"
                echo "| Fibbonacci Series |"
                echo "|-------------------|"
                echo "| Enter terms - "
                read d
                echo "|-------------------|"
                a=0
                b=1
                echo $a
                echo $b
                while [ $d -ge 0 ]
                do
                     c=`expr $a + $b`
                     echo $c
                     a=$b
                     b=$c
                     d=`expr $d - 1`
                done
                echo "| --- That's All ---|"
                echo "|-------------------|"
[student@localhost sshah]$ sh fibb
                |-------------------|
                | Fibbonacci Series |
                |-------------------|
                | Enter terms -
                10
                |-------------------|
                0
                1
                1
                2
                3
                5
                8
                13
                21
                34
                55
                89
                144
                | --- That's All ---|
                |-------------------|
[student@localhost sshah]$
```

7. **Archiving Files:** Archiving Files with tar

Compress an Entire Directory or a Single File

Use the following command to compress an entire directory or a single file on Linux. It'll also compress every other directory inside a directory you specify–in other words, it works recursively.

```
tar -czvf name-of-archive.tar.gz /path/to/directory-or-file
```

Here's what those switches actually mean:

- -c: **C**reate an archive.
- -z: Compress the archive with g**z**ip.
- -v: Display progress in the terminal while creating the archive, also known as "**v**erbose" mode. The v is always optional in these commands, but it's helpful.
- -f: Allows you to specify the **f**ilename of the archive.

Let's say you have a directory named "stuff" in the current directory and you want to save it to a file named archive.tar.gz. You'd run the following command:

```
tar -czvf archive.tar.gz stuff
```

Or, let's say there's a directory at /usr/local/something on the current system and you want to compress it to a file named archive.tar.gz. You'd run the following command:

```
tar -czvf archive.tar.gz /usr/local/something
```

Extract an Archive

Once you have an archive, you can extract it with the tar command. The following command will extract the contents of archive.tar.gz to the current directory.

```
tar -xzvf archive.tar.gz
```

It's the same as the archive creation command we used above, except the `-x` switch replaces the `-c` switch. This specifies you want to e**x**tract an archive instead of create one.

You may want to extract the contents of the archive to a specific directory. You can do so by appending the `-C` switch to the end of the command. For example, the following command will extract the contents of the archive.tar.gz file to the /tmp directory.

```
tar -xzvf archive.tar.gz -C /tmp
```

This is the simplest possible usage of the tar command. The command includes a large number of additional options, so It is not possible to list them all here.

8. Introduction **Working in X Windows:** Gedit, Gcalc, Xclock, Choosing and Changing Desktops & properties etc

9. **Using filters & redirections:** create new processed files
(Using Head, tail, cut, paste etc. create resultsheet/salarysheet)

```
[student@localhost sshah]$ cat studdata
# [studdata] input data file for studscript file
        01 ABC 70 65 80 74
        02 PQR 68 72 66 78
        03 LMN 65 58 60 62
[student@localhost sshah]$ cat studscript
        # [studscript]
        # Shell script using input file studdata to calculate students result
        # and store output result to new file studresult

        count=`cat studdata|wc -l`
        lines=2
        while [ $lines -le $count ]
        do
             record=`cat studdata|tail +$lines|head -1`
             set $record
             tot=`expr $3 + $4 + $5 + $6`
             per=`expr $tot / 4`
             echo $record $tot $per  >> studresult
             lines=`expr $lines + 1`
        done
        # -----------------------------------------------------------------
[student@localhost sshah]$ sh studscript
[student@localhost sshah]$ cat studresult
        01 ABC 70 65 80 74 289 72
        02 PQR 68 72 66 78 284 71
        03 LMN 65 58 60 62 245 61
[student@localhost sshah]$
```

10. **Shell Script** to read username and check he is login or not and send Wel-Come message to him. (using write)

```
[student@localhost sshah]$ cat userLogin
        # [userLogin] - Shell script to check user login and sent mesg to him if
        # user is not login then repeat after after every 30 seconds.
        # echo Enter user name and terminal no.

        un=$1
        tn=$2
        while [ 1 ]
        do
        rec=`who|grep $tn|cut -d" " -f3`
        echo $un $tn " : " $rec

        if test -z $rec
        then
              echo User $un $tn is not logged inn.
              sleep 30
        else
              echo User $un with terminal no $tn is logs in and mesg sent to him
              echo "Hello $un! Wel-Come"|write $un $tn
              exit
        fi
        done
        # ----------------------------------------------------------------------
[student@localhost sshah]$ sh userLogin student pts/13
        student pts/13  :  pts/13
        User student with terminal no pts/13 is logs in and mesg sent to him
[student@localhost sshah]$

#----------------------------------- [Output On login: student terminal: pts/13]
Red Hat Linux release 9 (Shrike)
Kernel 2.4.20-8 on an i686
login: student
Password:
Last login: Wed Sep 19 09:01:51 from 192.168.1.84
[student@localhost student]$ who am i
        student  pts/13       Sep 19 09:02 (192.168.1.91)
[student@localhost student]$
        Message from student@localhost.localdomain on pts/12 at 09:03 ...
        Hello student! Wel-Come
        EOF

[student@localhost student]$
```