



Linux with Shell Programming SED, AWK And Many More



INDEX

Part-1: UNIX/LINUX Commands	4
Topic-1: Overview of UNIX/Linux Operating System	6
Topic-2: Linux File System	10
Topic-3: Linux Installation	16
Topic-4: ls, date and cal commands	18
Topic-5: Working with Directories	24
Topic-6: Working with Files	34
Topic-8: Comparing Files	43
Topic-9: Creation of Link Files	47
Topic-10: Word count command (wc command)	50
Topic-11: Sorting content of the file	52
Topic-12: Find unique content in the file by using uniq command	57
Topic-13: Input and Output of Commands and Redirection	59
Topic-14: Piping	64
Topic-15: How to use multiple commands in a single line	68
Topic-16: Regular Expressions and Wildcard Characters	69
Topic-17: Command Aliasing	73
Topic-18: Locate and Find Commands	76
Topic-19: Compression and Uncompression of files(tar, gzip, gunzip, bzip2, bunzip2)	84
Topic-20: grep command	86
Topic-21: cut command	99
Topic-22: File Permissions	103
Topic-23: Working with editors	114
Part-2: Shell Scripting / Shell Programming	119
Topic-24: Shell Scripting / Shell Programming	121
Topic-25: What is shell script, She-Bang and First Script	124
Topic-26: Shell Variables	128
Topic-27: Variable Substitution and Command Substitution	132
Topic-28: Command Line Arguments	134
Topic-29: How to read dynamic data from the user	137
Topic-30: Operators	142
Topic-31: Control Statements	146
Topic-32: Arrays	182
Topic-33: Shell Script Functions	185
Topic-34: Project on Shell Programming	194



UNIX/LINUX



Topic-35: Stream Editor (SED)	198
Topic-36: AWK Programming	202
Topic-37: Project on awk programming	212
Part-3: Linux Administration Basics	216
Topic-38: Job Scheduling with crontab	218
Topic-39: User Management	221
Topic-40: Working with putty, Mobaxterm and XShell	231
Topic-41: How to customize Open Source Software Code	233
Topic-42: Process Management	236
Topic-43: Communication Commands (write, wall, msg, mail)	240
How to use Gmail from the Ubuntu Terminal to send Emails	243
Topic-44: Package Management	246
Topic-45: Install and Working with MySQL Database	250
Topic-46: Working with Java	257
Topic-47: Job Control	259
Topic-48: nohup command	262
Topic-49: at command	263
Topic-50: Memory related commands	265
Topic-51: Networking commands	270
Topic-52: Working with winscp	273
Topic-53: Working with Filezilla	274
Topic-54: init command	275
Topic-55: Top Most important Linux Questions	276



Part-1



UNIX / LINUX Concepts & Commands



Agenda

Topic-1: Overview of UNIX/Linux Operating System

Topic-2: Linux File System

Topic-3: Linux Installation

Topic-4: ls, date and cal commands

Topic-5: Working with Directories

Topic-6: Working with Files

Topic-7: Comparing Files

Topic-8: Creation of Link Files

Topic-9: Word count command (wc command)

Topic-10: Sorting content of the file

Topic-11: Find unique content in the file by using uniq command

Topic-12: Input and Output of Commands and Redirection

Topic-13: Piping

Topic-14: How to use multiple commands in a single line

Topic-15: Regular Expressions and Wildcard Characters

Topic-16: Command Aliasing

Topic-17: Locate and Find Commands

Topic-18: Compression and Uncompression of files (tar, gzip, gunzip, bzip2, bunzip2)

Topic-19: grep command

Topic-20: cut command

Topic-21: File Permissions

Topic-22: Working with editors



Topic-1: Overview of UNIX/Linux Operating System

- What is UNIX?
- Flavours of UNIX
- Components of UNIX
- Online Terminal Demo with some basic commands

What is UNIX?

- * It is an operating system, by using that, users/applications can communicate with hardware components.
- * It was developed/created in 1960s.
- * With lot of extensions and improvements to base version, several flavours introduced by organization/companies (flavours like Redhat linux, ubuntu, CentOS etc)

Features of UNIX:

- 1) It is FOSS (Freeware and Open Source Software)
- 2) UNIX can be used by multiple users simultaneously and hence it is Multi User operating System.
- 3) Several tasks can be executed simultaneously and hence it is multi tasking operating system.
- 4) It is user friendly and provides both CUI and GUI Support.
- 5) When compared with windows, UNIX is more secured.

Flavours of UNIX:

As UNIX is open source, multiple flavours are available with lot of extensions and improvements.

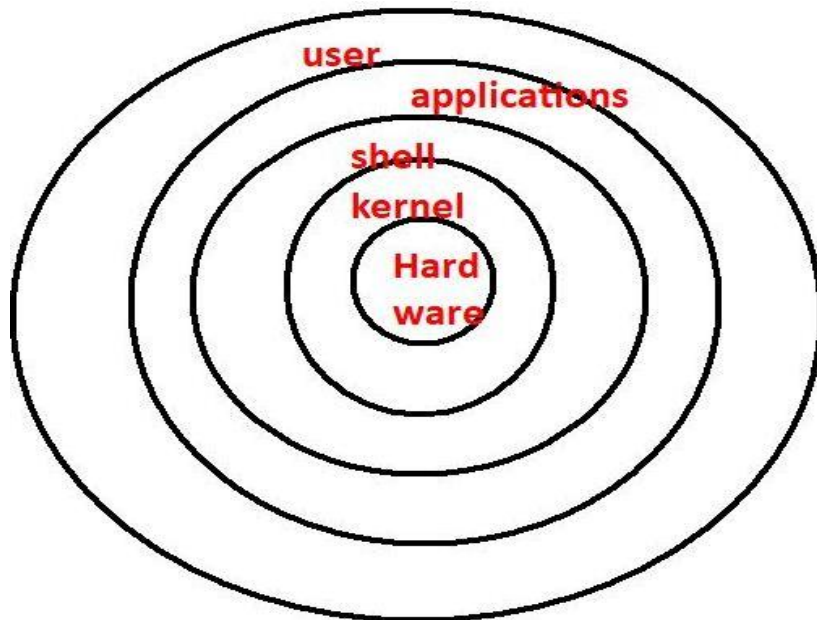
- Ubuntu
- RedHat
- Centos
- Fedora
- Slackware
- open solaris
- Suse Linux Enterprise server (SLES)
- Open Suse



All these flavours have lot of similarity. Hence if we are perfect with one flavour, we can work on any other flavour very easily.

Note: We can view a detailed list of Linux flavours in www.distrowatch.com

Components of UNIX



Shell:

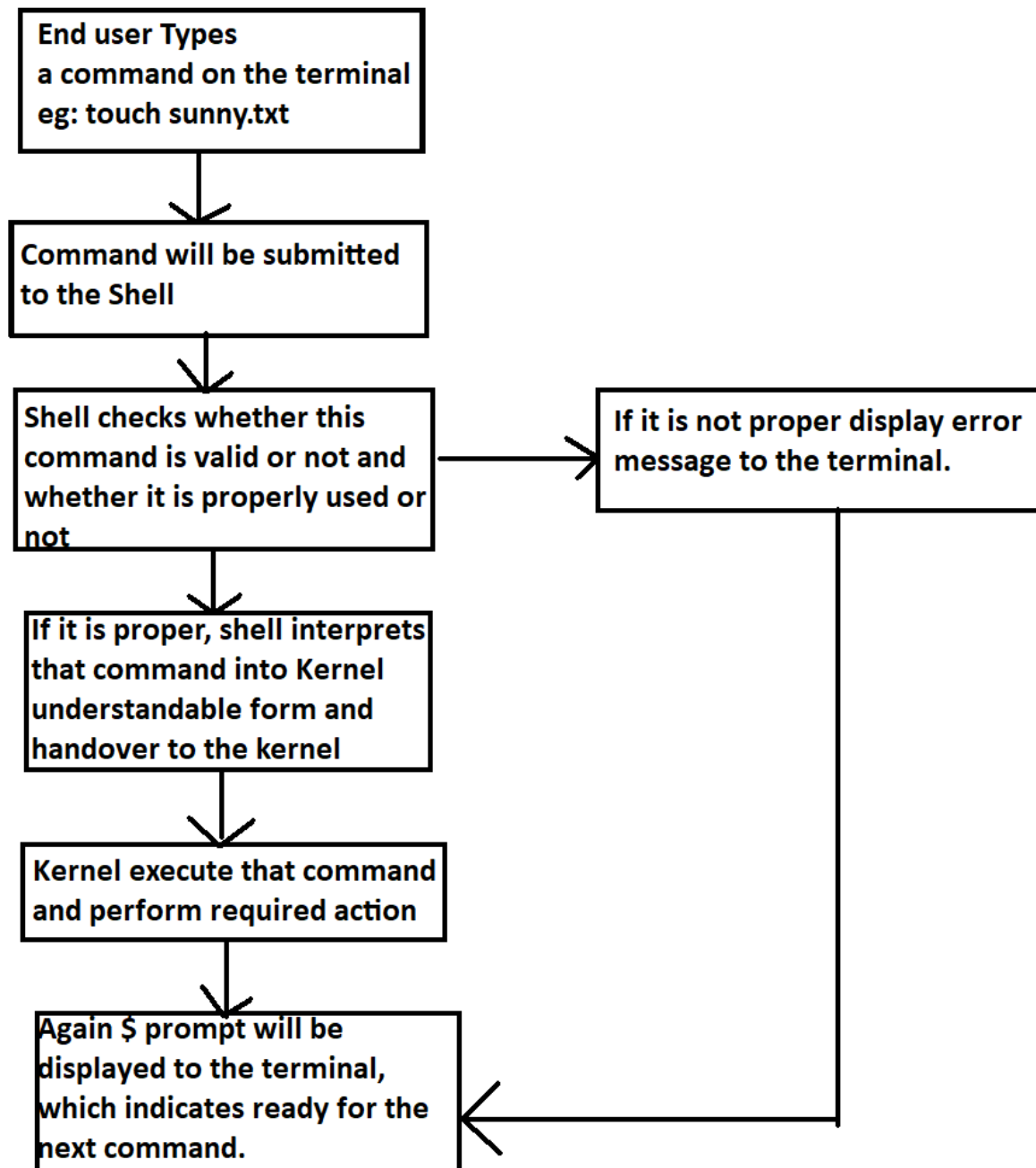
- * It is the outer layer of UNIX operating System.
- * It reads our command, verify syntax and check whether the corresponding command related application is available or not.
- * If everything is proper, then shell interprets our command into kernal understandable form and handover to the kernal.
- * Shell acts as interface between user and kernal.

Kernal:

- * It is the core component of UNIX operating system.
- * It is responsible to execute our commands.
- * It is responsible to interact with hardware components.
- * Memory allocation and processor allocation will takes care by kernal.



Command Execution Flow



- 1) User types the command in the terminal.
touch durga.txt
- 2) Shell reads that command. It will check whether that command is valid or not and whether it is used properly or not. If everything is proper, then shell interprets/translate that command into kernel understandable form.
- 3) Shell handovers that interpreted command to the kernel.
- 4) Kernel executes that command and perform required activity.



- 5) Once command execution completed, then shell returns unix prompt (\$ OR # OR %).
- 6) \$ OR # OR % represents it is ready for the next command.

Online UNIX Terminal:

<http://www.masswerk.at/jsuix>

- * It is free terminal and written in JavaScript.
- * We can access by using any browser.
- * We can use this terminal to check very basic commands functionality.

The most commonly used basic Commands:

- 1) pwd → Print working directory
- 2) ls → List our all files and directories
- 3) mkdir → Create directory
- 4) cd → Change directory
- 5) touch → To create a file
- 6) rmdir → Remove directory
- 7) rm → To remove file
- 8) cal → Display Monthly Calander
- 9) date → Display current date and time.
- 10) help → To display list of commands.
- 11) hello → To display brief system information.
- 12) clear → To clear terminal.
- 13) exit → To logout session.



Topic-2: Linux File System

Types of Files in Linux:

In Linux everything is treated as File.

All files are divided into 3 types

1) Normal or Ordinary files:

These files contain data. It can be either text files (like abc.txt) OR binary files (like images, videos etc).

2) Directory Files:

- These files represent directories.
- In windows, we can use folder terminology where as in linux we can use directory terminology.
- Directory can contains files and sub directories.

3) Device Files:

In Linux, every device is represented as a file. By using this file we can communicate with that device.

Note: short-cut commands to open and close terminal

ctrl+alt+t → To open terminal

ctrl+d → To close terminal

How to check File Type:

In Ubuntu, blue color files represents directories and all remaining are considered as normal files. This color conventions are varied from flavour to flavour. Hence it is not standard way to check file type.

We have to use 'ls -l' command.



```
drwxr-xr-x 102 durgasoft durgasoft 4096 Nov 17 21:19 magic
-rw-r--r-- 1 durgasoft durgasoft 0 Nov 17 21:24 sunny.txt
```

The first character represents the type of file.

d → Directory File

- → Normal File

l → Link File

c → Character Special File

b → Block Special File

s → Socket File

Note: c, b, s are representing system files and mostly used by super user (also known as root user or admin user)

File System Navigation Commands:

1. Every directory implicitly contains 2 directories . and ..

. represents current directory

.. represents parent directory

2. \$ cd .

changes to current directory (useless)

3. \$ cd ..

Changes to parent directory

4. \$ cd

If we are not passing any argument, then changes to user home directory.

5. \$ cd ~

~ means user home directory.

It will changes to user home directory.

6. \$ cd -

- means previous working directory.

It will changes to previous working directory.

```
durgasoft@durgasoft:~/Downloads/coreutils-8.31/src$ pwd
/home/durgasoft/Downloads/coreutils-8.31/src
durgasoft@durgasoft:~/Downloads/coreutils-8.31/src$ cd ~
durgasoft@durgasoft:~$ pwd
/home/durgasoft
durgasoft@durgasoft:~$ cd -
/home/durgasoft/Downloads/coreutils-8.31/src
durgasoft@durgasoft:~/Downloads/coreutils-8.31/src$ pwd
/home/durgasoft/Downloads/coreutils-8.31/src
```



```
durgasoft@durgasoft:~/Downloads/coreutils-8.31/src$ cd ../../..  
durgasoft@durgasoft:~$ pwd  
/home/durgasoft
```

Linux File System Hierarchy:

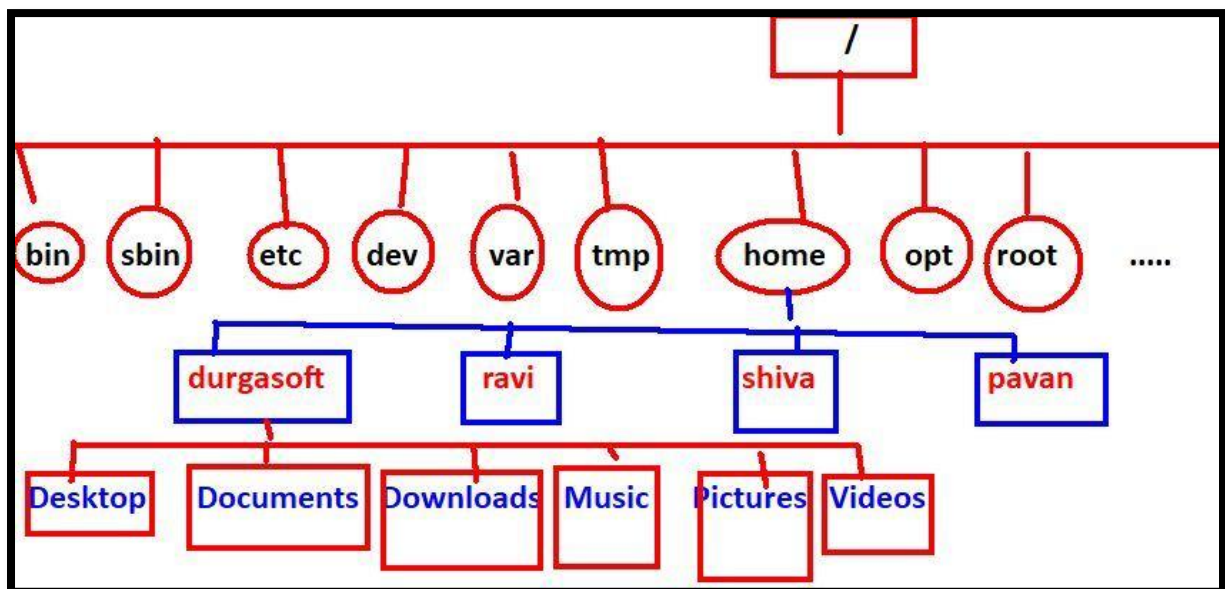
Linux file system has Tree Like Structure.

It starts with root(/).

/ is the topmost directory

This root directory contains the following important sub directories.

bin,sbin,lib,etc,dev,opt,home,usr,tmp,media etc



1) bin Directory:

bin means binary. This directory contains all binary executables related to our linux commands.

2) sbin Directory:

sbin means systembin. It contains all binary executables related to high end admin (super user OR root) commands.

Eg: Disk partitioning, network management etc

Q. What is the difference between bin and sbin?

bin contains binary executables related to commands used by normal user.

sbin contains binary executables related to commands used by superuser.



3) etc Directory:

- This directory contains all system configuration files. These configurations can be used to customize behaviour of linux os.
- All users information available in /etc/passwd file.
- All groups information available in /etc/group file.
- Hosts information (ip address and dns names) available in /etc/hosts file.

4) tmp Directory:

- tmp means temporary. It contains all temporary files created in the current session.
- If any file is required only for the current session, then create that file inside tmp directory. These files will be deleted automatically at the time of system shutdown.
- If any file which is required permanently, then it is not recommended to create inside tmp directory.

5) dev Directory:

- dev means device.
- In Linux, everything is treated as a file including devices also. i.e every device is represented as a file. By using these files, we can communicate with the devices.
- All device related files will be stored inside dev directory.

Eg:

tty → Terminal related File

fd → Floppy Drive related File

hd → Hard Disk related File

ram → RAM related File

stdin → standard Input Device File (keyboard)

stdout → Standard Output Device File (Terminal/Monitor)

stderr → Standard Error Device File (Terminal/Monitor)

6) mnt Directory:

- mnt means mounting.
- We have to attach external file system files from Pen drive, CD, external hard disk etc to the Linux File System. Then only we can use those external files. This attachment process is called mounting.
- In the old operating systems, we have to perform mounting manually. But in recent operating systems, mounting is performing automatically and we are not required to perform manually.
- The files of manual mounting will be placed inside mnt directory.

7) media Directory:

The files of automatic mounting will be placed inside media directory.



Q. What is the difference between mnt and media?

mnt → Contains manual mounting files.

media → Contains automatic mounting files.

8) opt Directory:

- opt means optional.
- This directory contains all 3rd party software installation files.

Eg:

If we are installing any software explicitly like google chrome, then the corresponding installation files will be stored inside opt directory.

9) lib Directory:

lib means library. It contains Linux os libraries which are required by our commands and applications.

10) var Directory:

- var means variable data. If any data which is keep on changing, such type of data will be stored inside var directory.
- log files will be stored inside var.

11) home Directory:

- As linux is multi user operating system, for every user a separate directory will be created to hold his specific data like videos, images, documents etc. All these user directories will be stored inside home directory.
- \$ ls /home
- demo demo1 demo2 durga durga1 durga2 durga5 durgasoft

Note:

/home/durgasoft → Is called durgasoft user home directory. It contains multiple sub directories like Desktop, Downloads, Movies, Pictures etc.

12) proc Directory:

- proc means processes.
- In Linux, multiple processes are running simultaneously. For every process a unique id will be there, which is also known as PID (Process ID).
- The data related to current running processes will be stored inside proc directory. For every process a separate directory will be created inside proc to maintain that data. The name of this directory is same as PID.

Note: We can find all running processes information by using ps command.

ps means process status.



```
durgasoft@durgasoft:/$ ps -ef
UID          PID    PPID  C STIME TTY          TIME CMD
root           1        0  0 11:05 ?          00:00:03 /sbin/init
root           2        0  0 11:05 ?          00:00:00 [kthreadd]
root           3        2  0 11:05 ?          00:00:00 [rcu_gp]
root           4        2  0 11:05 ?          00:00:00 [rcu_par_g
....
```

13) root Directory:

It is the home directory of super user.

Note:

/home/durgasoft → Durgasoft User Home Directory

/root → Super User Home Directory

Q. What is the difference between / and root directories?

/ acts as root for Linux file system. It is the topmost directory of linux file system.

root is subdirectory of /, which acts as home directory for the super user.

14) boot Directory:

This directory contains the files which are required to boot linux os.

15) usr Directory:

usr means user. This directory contains all user related softwares.

Note:

- 1) The main advantage of Linux File System is, operating system can locate required files very easily.
- 2) For every File System, a separate name will be assigned.
- 3) ext2,ext3,ext4,XFS are names of Linux File Systems.
- 4) NTFS, FAT are names of Windows File Systems.



Topic-3: Linux Installation

- 1) Oracle Virtual Box Installation
- 2) Virtual Machine Installation with Ubuntu OS

1) Oracle Virtual Box Installation:

To run virtual computers in our system without effecting original computer, we should go for virtual box.

It can extends the capability of our existing computer so that we can run multiple operating systems simultaneously.

Download virtual box from:

virtualbox.org → Downloads → Windows hosts

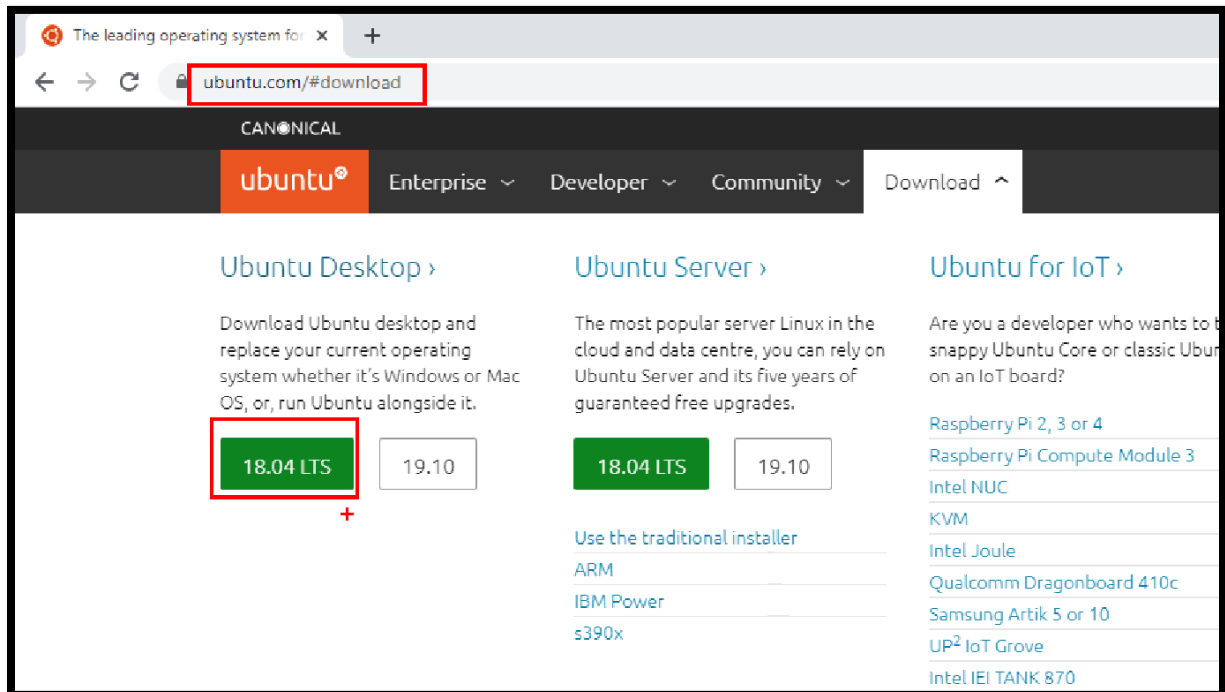
VirtualBox-6.0.14-133895-Win.exe





2) Creation of Virtual Machine with Ubuntu OS:

We have to download ubuntu software from
ubuntu.com → Download → Ubuntu Desktop → Ubuntu 18.04.3 LTS → Download
ubuntu-18.04.3-desktop-amd64.iso



Note: Download softwares from: bit.ly/36evx8y

Various Utilities:

- 1) To make Full Screen: Devices → Insert Guest Additions CD Image
- 2) To open terminal: **ctrl+alt+t**
- 3) To close terminal: **ctrl+d**
- 4) To increase font in terminal: **ctrl+shift+plus symbol**
- 5) To decrease font in terminal: **ctrl+ minus symbol**
- 6) To copy and paste from windows to ubuntu and from ubuntu to windows:
Devices → Shared Clipboard → Bidirectional
- 7) To drag and drop files from windows to ubuntu and from ubuntu to windows:
Devices → Drag and Drop → Bidirectional



Topic-4: ls, date and cal Commands

1) ls Command:

We can use ls command to list out all files and directories present in the given directory.
We can get manual documentation for any command by using man.

man ls

It provides complete information about ls command.

Various options of ls Command:

1) ls

It will display all files and directories according to alphabetical order of names.

2) ls -r

It will display all files and directories in reverse of alphabetical order.

3) ls | more

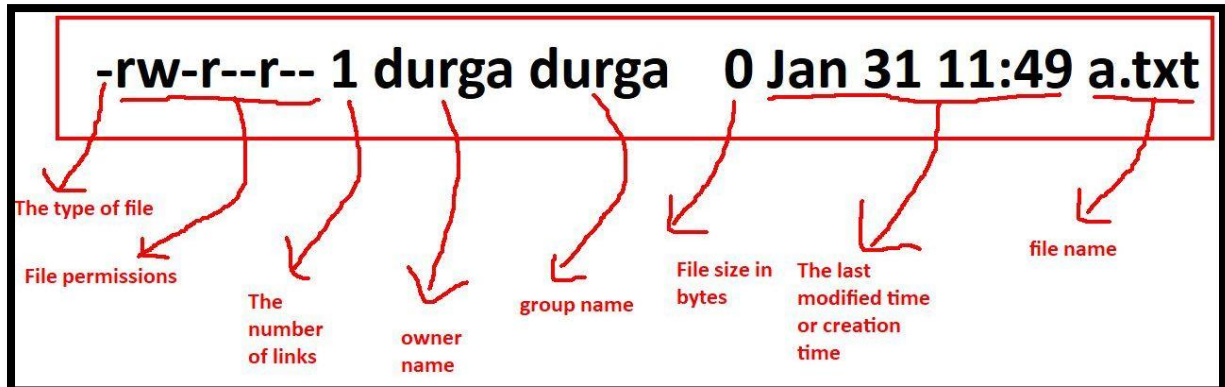
To display content line by line
(To come out we have to use q)

4) ls | pg

To display content page by page.
Each page contains 20 lines of content.
(To come out we have to use q)

5) ls -l

To display long listing of files



6) ls -t

To display all files based on last modified date and time. Most recent is at top and old are at bottom.

7) ls -rt

To display all files based on reverse of last modified date and time. Old files are at top and recent files are at bottom.

8) ls -a

a means all

To display all files including hidden files. Here . and .. also will be displayed.

9) ls -A

A means almost all

To display all files including hidden files except . and ..

```
durga@durga-VirtualBox:~$ ls -a
.          msmtprc
..         .msmtprc
.bash_history Music
.bash_logout .mysql_history
.bashrc      Pictures
.cache      .profile
.config     Public
.dbus       script1.sh
durga@durga-VirtualBox:~$ ls -A
.bash_history .msmtprc
.bash_logout Music
.bashrc      .mysql_history
.cache      Pictures
.config     .profile
.dbus       Public
```



10) ls -F

To display all files by type.

directory → /
executable file → *
link file → @

Eg:

initctl@ → Link File

pts/ → Directory

ls* → Executable File

11) ls -f

To disable colors

12) ls -l

- To display all files including inode number.
- i-node is the address of location, where file attributes are stored.
- i-node is the address of the location, where file attributes are stored.
- The following are various file attributes
 - 1) The size of the file
 - 2) The number of links
 - 3) The owner
 - 4) The group
 - 5) The creation time
 - 6) The last modified time
 - 7) The last accessed timeetc

13) ls -R

- R means Recursive.
- It will list all files and directories including sub directory contents also. By default ls will display only direct contents but not sub directory contents.

14) ls -s

The number of blocks used by file will be displayed.

1 Block = 1Kb

Note: In ubuntu each block is of 1KB but not 4KB.

15) ls -h

display in human readable format



Note:

If the number of files are very huge, then we can use less and more commands with ls to display page by page.

```
$ ls /dev | less
$ ls /dev | more
```

If we want only fixed number of files either from top or from bottom we have to use head and tail commands with ls commands.

```
$ ls /dev | head -5 → display only top 5 lines
$ ls /dev | tail -5 → display only bottom 5 lines
```

Note: We can use these options simultaneously. When ever using options simultaneously then the order is not important.

Eg: All the following commands are equal

```
$ ls -l -t -r
$ ls -t -r -l
$ ls -l -r -t
$ ls -ltr
$ ls -trl
```

Q1. Write the command to display all files including hidden files in last modification time order. Oldest should be first and recent should be last. It should include inode number and the number of blocks used by that file. The output should be in long listing form?

```
$ ls -attrls
131279 4 -rw-r--r-- 1 durga durga 807 Jan 3 12:57 .profile
131277 4 -rw-r--r-- 1 durga durga 3771 Jan 3 12:57 .bashrc
162011 4 -rw-r--r-- 1 durga durga 220 Jan 3 12:57 .bash_logout
132496 4 drwx----- 3 durga durga 4096 Jan 3 13:03 .gnupg
132517 4 drwx----- 3 durga durga 4096 Jan 3 13:03 .local
404481 4 drwxr-xr-x 2 durga durga 4096 Jan 3 13:03 Templates
```

Q2. Which command will lists all files including hidden files along with their inode numbers?

```
ls -ai
```

Q3. Which command will make a long listing of all the files in our system including hidden files, sorted by modification date(oldest first)?

```
ls -latr
```

Q4. Is -r will list the files sorted by modification date (Oldest first)?

False.

It lists the files based on reverse of alphabetical order of names.



ls -rt → It will list the files sorted by modification date (Oldest first)

Q5. ls -la will not produce the same result as ls -al

False

2) date Command:

We can use date command to display date and time of system.

Various Options:

1) date +%D

To display only date in the form: mm/dd/yy

2) date +%T

To display only time in the form: hh:mm:ss

3) date +%d

To display only day value

4) date +%m

To display only month value

5) date +%y

To display only year value in yy form

6) date +%Y

To display only year value in yyyy form.

7) date +%H

To display only Hours value (in 24 hours scale format)

8) date +%M

To display only Minutes value

9) date +%S

To display only Seconds value

Eg 1: To display current system date in dd-mm-yyyy format.

default format: mm/dd/yy

date +%d-%m-%Y



Eg 2: Create an empty file where file name contains current system date.

```
touch "durgajobs$(date +%d%m%Y).log"
```

```
durgajobs31102019.log  
durgajobs01112019.log  
durgajobs02112019.log  
durgajobs03112019.log  
durgajobs04112019.log
```

Eg 3: Create an empty file where file name contains current system date and time

```
touch "durgajobs$(date +%d%m%Y%H%M%S).log"
```

```
durgajobs31102019205834.log
```

Note:

If the file name contains date and time then that file is said to be timestamped file (file with timestamp)

cal Command:

```
$ cal → To display current month calendar.  
$ cal 2020 → To display total year calendar.  
$ cal 1 → To display 1st year calendar.  
$ cal 9999 → To display 9999th year calendar.  
$ cal 10000 → cal: year '10000' not in range 1..9999  
$ cal 08 2019 → To display august 2019th calendar
```

Note: cal command can provide support only for the years 1 to 9999.



Topic-5: Working with Directories

1) Creation of Directories:

We can create directories by using mkdir command.

1) mkdir dir1

To create a directory

2) mkdir dir1 dir2 dir3

To create multiple directories

3) mkdir dir1/dir2/dir3

To create dir3. But make sure dir1 and in that dir2 should be available already.

4) mkdir -p dir1/dir2/dir3

- -p means path of directories.
- All directories in the specified path will be created.
- First dir1 will be created and in that dir2 will be created and within that dir3 will be created.

Case Study-1: Film Heroine's Manager

heroines

 sunny

 jan2020, feb2020, mar2020,dec2022

 katrina

 jan2020, feb2020, mar2020,dec2022

 kareena

 jan2020, feb2020, mar2020,dec2022

jan2020

 schedule_1.txt

 schedule_2.txt

 ...

 schedule_31.txt

\$ mkdir heroines

\$ cd heroines



```
$ mkdir sunny katrina kareena
$ mkdir
{sunny,katrina,kareena}/{jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec}_{2020,2021,2022}
```

```
$ touch
{sunny,katrina,kareena}/{jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec}_{2020,2021,2022}/schedule_{1..31}.txt
```

Case Study-2:

Create 5 directories named with dir6,dir7,dir8,dir9 and dir10. In these directories create empty files with a.txt,b.txt,c.txt and d.txt

```
$ mkdir dir{6..10}
$ touch dir{6..10}/{a..d}.txt
```

Note: *,[,]{ } etc are called wild characters. We can use wild card characters in every command.

2) How to remove Directories:

We can remove directories by using rmdir command.

1) \$ rmdir dir1

To remove empty directory dir1

2) 2. \$ rmdir dir1 dir2 dir3

To remove multiple empty directories

Note: rmdir command will work only for empty directories. If the directory is not empty then we will get error. We cannot use rmdir for files. Hence the most useless (waste) command in linux is rmdir.

```
durga@durga-VirtualBox:~$ rmdir heroines/
rmdir: failed to remove 'heroines/': Directory not empty
```

If the directory is not empty then to remove that directory we should use rm command. All internal content also will be removed. rm command can work for files also. Hence rm is recommended to use than rmdir.

```
durga@durga-VirtualBox:~$ rm heroines
rm: cannot remove 'heroines': Is a directory
```

Whenever we are using rm command for directories, we should use -r or -R option. Here case is not important.



```
$ rm -r heroines  
$ rm -R heroines
```

Note: In Linux operating system, there is no way to perform undo operation. Once we delete a file or directory, it is impossible to retrieve that. Hence while using rm command we have to take special care.

The following command is the most dangerous command in linux, because it removes total file system.

```
rm -r /
```

Various options with rm Command:

1) interactive Option(-i)

While removing files and directories, if we want confirmation then we have to use -i option.

```
durga@durga-VirtualBox:~$ rm -ri dir7  
rm: descend into directory 'dir7'? y  
rm: remove regular empty file 'dir7/c.txt'? y  
rm: remove regular empty file 'dir7/d.txt'? y  
rm: remove regular empty file 'dir7/a.txt'? y  
rm: remove regular empty file 'dir7/b.txt'? y  
rm: remove directory 'dir7'? y
```

2) force removal(-f):

While removing files and directories, if we don't want any error messages, then we should use -f option. It is opposite to -i option.

```
durga@durga-VirtualBox:~$ rm -r dir99  
rm: cannot remove 'dir99': No such file or directory  
durga@durga-VirtualBox:~$ rm -rf dir99  
durga@durga-VirtualBox:~$
```

Even dir99 is not available, we won't get any error message, because we used -f option.

3) verbose Option(-v):

If we want to know the sequence of removals on the screen we should go for -v option.

```
durga@durga-VirtualBox:~$ rm -r dir6  
durga@durga-VirtualBox:~$ rm -rv dir8  
removed 'dir8/c.txt'  
removed 'dir8/d.txt'  
removed 'dir8/a.txt'  
removed 'dir8/b.txt'
```



removed directory 'dir8'
durga@durga-VirtualBox:~\$

Q1) What is the difference between the following 2 Commands?

\$ mkdir dir1/dir2/dir3
\$ mkdir -p dir1/dir2/dir3

mkdir dir1/dir2/dir3

Only dir3 will be created and compulsory dir1 and in that dir2 should be available already. If dir1 or dir2 not available then this command won't work.

mkdir -p dir1/dir2/dir3

-p means complete path
All 3 directories will be created.

Q2) What is the Advantage of using rm Command over rmdir Command while removing Directories?

rmdir command will work only for empty directories.
rm command will work for both empty and non-empty directories. Even we can rm command for files also.

Q3) Assume that dir1 is an Empty Directory. Which of the following Commands will remove dir1?

- rm dir1
- remove dir1
- rmdir dir1
- del dir1

- Ans: C

Q4) Assume that dir1 is non-empty Directory. Which of the following Commands will remove dir1?

- rmdir dir1
- rm -R dir1
- rm -i dir1
- rm -f dir1
- rm -v dir1

- Ans: B



Q5) Assume that dir1 is non empty Directory and text1 is just a Text File. Which of the following Command will remove both dir1 and text1 successfully?

- rm text1 dir1
- rm -v text1 dir1
- rm -R text1 dir1

Ans: C

Q6) How to Create a Directory called pythonclasses in the Videos Directory within User Home Directory?

\$ mkdir ~/Videos/pythonclasses

Q7) How to Create a Directory named A and in that a Directory B and inside that a Directory C?

-\$ mkdir -p A/B/C

Q8) How many Directories will be created after running the following Command?

\$ mkdir {a..c}{1..3}

9 Directories named with a1, a2, a3, b1, b2, b3, c1, c2, c3

Q9) To Create a Directory named with Java Classes, is the following Command valid?

\$ mkdir java classes

This command will create two directories java and classes.

To create a single directory we have to use:

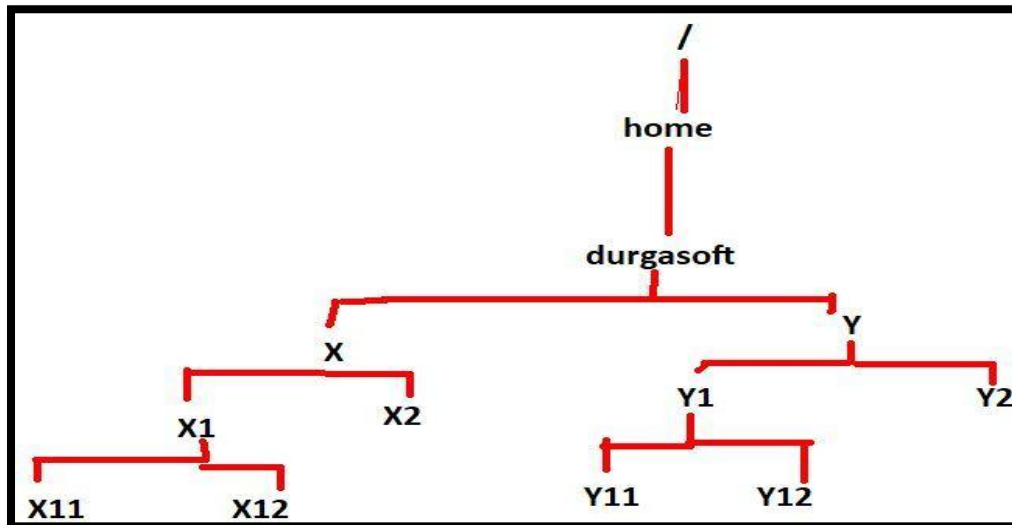
\$ mkdir "java classes"

Note: In file or directory names, it is not recommended to use space. Instead of that we have to use _ symbol like java_classes.



Case Study:

Write commands to create the following directory structure



1st way:

```
durga@durga-VirtualBox:~$ pwd
/home/durga
durga@durga-VirtualBox:~$ mkdir x y
durga@durga-VirtualBox:~$ cd x
durga@durga-VirtualBox:~/x$ mkdir x1 x2
durga@durga-VirtualBox:~/x$ cd x1
durga@durga-VirtualBox:~/x/x1$ mkdir x11 x12
durga@durga-VirtualBox:~/x/x1$ cd ..
durga@durga-VirtualBox:~/x$ cd ..
durga@durga-VirtualBox:~$ cd y
durga@durga-VirtualBox:~/y$ mkdir y1 y2
durga@durga-VirtualBox:~/y$ cd y1
durga@durga-VirtualBox:~/y/y1$ mkdir y11 y12
```

2nd way:

```
$ mkdir x x/x1 x/x2 x/x1/x11 x/x1/x12 y y/y1 y/y2 y/y1/y11 y/y1/y12
```

3rd way:

```
$ mkdir -p x/x1/x11 x/x1/x12 x/x2 y/y1/y11 y/y1/y12 y/y2
```

4th way:

```
$ mkdir -p x/x{1,2} x/x1/x1{1,2} y/y{1,2} y/y1/y1{1,2}
```



Q10) To Remove Directories dir1, dir2, dir3.... dir10

If Directories are Empty → `$ rmdir dir{1..10}`

If Directories are non Empty → `$ rm -R dir{1..10}`

Q11) To Remove 3 Empty Directories dir2, dir4, dir6

`$ rmdir dir{2,4,6}`

Q12) To Remove Directories where Name contains 2 OR 4 → \$ rmdir *[24]*

Q13) To Remove Directories where Name Starts with 'd' → \$ rmdir d*

Q14) To Remove Directories where Name Starts with 'd' and Ends with 'n'

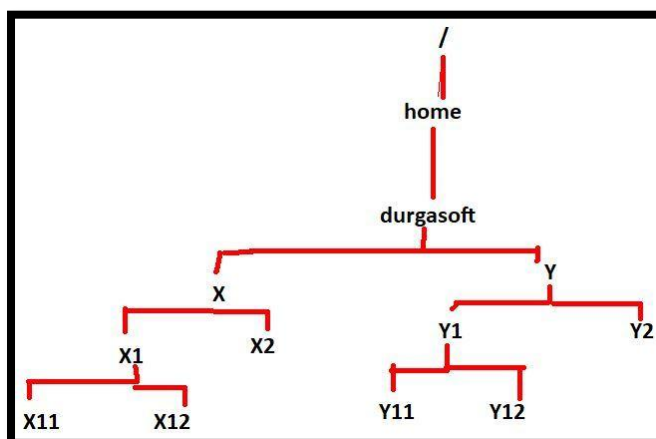
`$ rmdir d*n`

Q15) To Remove Directories where Name Starts with 'd' OR 'x' → \$ rmdir [dx]*

Absolute Path vs Relative Path:

Absolute Path: It is the path from root(/) to destination. ie it is complete path.

Relative Path: It is the path from current working directory to destination directory. It is always wrt current location.



`$ mkdir -p x/x{1,2} x/x1/x1{1,2} y/y{1,2} y/y1/y1{1,2}`

Eg 1: Assume we are in x11 directory. To change to y2 directory

Absolute Path:

`$ cd /home/durga/y/y2`

`$ cd ~/y/y2`

Relative Path:

`$ cd ../../../y/y2`



Eg 2: Assume we are in y2 directory. To change to x11

Absolute Path:

```
$ cd /home/durga/x/x1/x11
```

```
$ cd ~/x/x1/x11
```

Relative Path:

```
$ cd ../../x/x1/x11
```

Eg 3: Assume we are in x11 directory. To create y21 directory inside y2 without enter into y2 directory.

Absolute Path:

```
$ mkdir /home/durga/y/y2/y21
```

```
$ mkdir ~/y/y2/y21
```

Relative Path:

```
$ mkdir ../../y/y2/y21
```

4) Copy Command (cp)

1) To Copy from File1 to File2 (File to File)

- \$ cp source_file destination_file
- \$ cp file1 file2
- Total content of file1 will be copied to file2.
- If file2 is not already available, then this command will create that file.
- If file2 is already available and contains some data, then this data will be over write with file1 content.

2) To Copy File to Directory:

- \$ cp file1 file2 output
- file1 and file2 will be copied to output directory.
- Here we can specify any number of files, but last argument should be directory.
- output directory should be available already.

3) To Copy all Files of One Directory to another Directory:

- \$ cp dir1/* dir2
- All files of dir1 will be copied to dir2
- But dir2 should be available already.

4) To Copy Total Directory to another Directory:

- \$ cp dir1 dir2
- cp: -r not specified; omitting directory 'dir1'



- Whenever we are copying one directory to another directory, compulsory we should use -r option.
- `$ cp -r dir1 dir2`
- total dir1 will be copied to dir2

Note:

If the destination directory (dir2) already available then total dir1 will be copied to dir2.
If the destination directory (dir2) not already available, then destination directory will be created and all files of source directory will be copied to destination directory but source directory won't be copied.

5) To Copy Multiple Directories into a Directories:

- `$ cp -r dir1 dir2 dir3 dir4 dir5`
- dir1,dir2,dir3 and dir4 will be copied to dir5

Q. Write command to copy data from a.txt,b.txt,c.txt to d.txt?

`$ cp a.txt b.txt c.txt d.txt` → It won't work.

We will discuss solution in the next classes.

Moving and Renaming Directories:

Both moving and renaming activities can be performed by using single command: mv

1) Renaming of files:

`$ mv oldname newname`

Eg: `$ file1.txt file2.txt`

file1.txt will be renamed to file2.txt

2) Renaming of Directories:

`$ mv dir1 dir2`

dir1 will be renamed to dir2

3) Moving files to directory:

`$ mv a.txt b.txt c.txt output`

a.txt,b.txt and c.txt will be moved to output directory.

4) Moving of all files from one directory to another directory:

`$ mv dir1/* dir2`

All files of dir1 will be moved to dir2. After executing this command dir1 will become empty.

5) Moving total directory to another directory:

`$ mv dir1 dir2`



Note: If dir2 is already available then dir1 will be moved to dir2. If dir1 is not already available then dir1 will be renamed to dir2.

Summary of Directory related Commands:

`mkdir dir1`

`mkdir dir1 dir2 dir3`

`mkdir dir1/dir2/dir3`

`mkdir -p dir1/dir2/dir3`

`mkdir dir{1..6}`

`rmdir dir1`

`rmdir dir1 dir2 dir3`

`rm -r dir1`

`rm -ri dir1`

`rm -rf dir1`

`rm -rv dir1`

`rm -r dir*`

`rm -r dir{2..6}`

`rm -r dir[2,4]`

`cp file1.txt file2.txt`

`cp file1.txt file2.txt file3.txt output`

`cp dir1/* dir2`

`cp -r dir1 dir2`

`mv file1.txt file2.txt`

`mv dir1 dir2 (rename b'z dir2 not available)`

`mv dir1/* dir2`

`mv dir1 dir2 (move dir1 to dir2 because dir2 available)`

`cd`

`cd ../../..`

`cd /`

`cd ~`

`cd -`



Topic-6: Working with Files

- 1) Creation of Files
- 2) Viewing of Files
- 3) Copying of Files
- 4) Comparison of Files
- 5) Renaming of files
- 6) Deleting Files
- 7) Creation of Hidden files and directories
- 8) Creation of Link Files
- 9) Editing of Files

1) Creation of Files:

In Linux, we can create files in the following ways:

- 1) By using touch command (to create empty file)
- 2) By using cat command
- 3) By using editors like gedit, vi, nano etc

cat Command:

```
cat > file1.txt
```

Eg:

```
$ cat > file1.txt
```

```
Hello Friends
```

```
Listen Carefully
```

```
Otherwise Linux will give Left and Right
```

```
ctrl+d → To save and exit
```

If file1.txt is not already available, then file1.txt will be created with our provided data.

If file1.txt is already available with some content, then old data will be over written with our provided new data.

Instead of overwriting, if we want append operation then we should use >> with cat command.

```
cat >> file1.txt
```

```
extra content
```

```
ctrl+d
```



Q1. What is the difference between touch and cat?

touch for creating empty file where as cat for creating a file with some content.

Q2. How we can perform overwriting and appending with cat command?

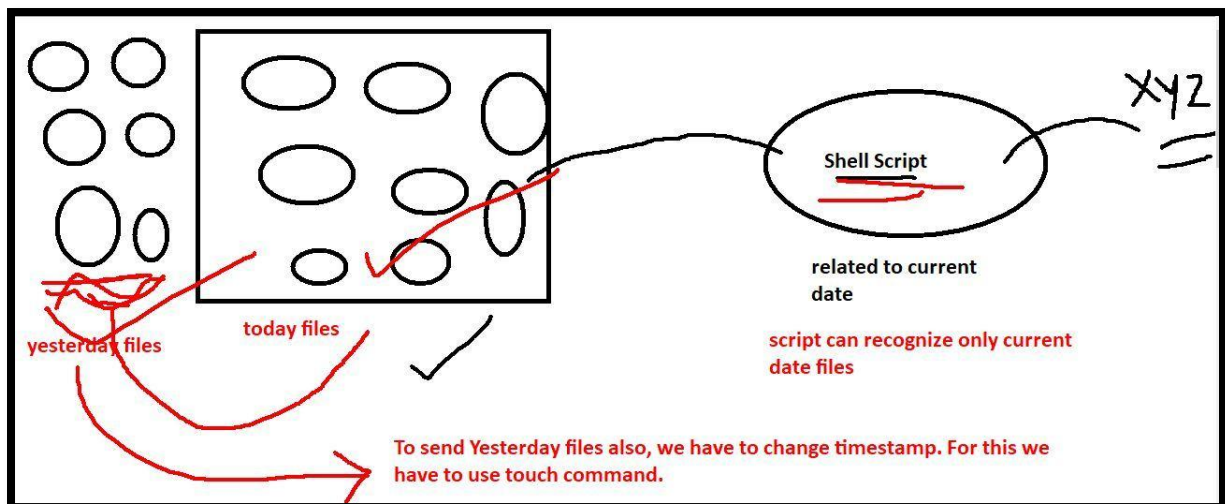
- > meant for overwriting
- >> meant for appending/concatenation

Q3. If we are using touch command, but the file is already available then what will happen?

The content of the file won't be changed. But last modified date and time (i.e timestamp) will be updated.

```
durga@durga-VirtualBox:~$ ls -l file1.txt
-rw-r--r-- 1 durga durga 77 Jan  9 12:24 file1.txt
durga@durga-VirtualBox:~$ touch file1.txt
durga@durga-VirtualBox:~$ ls -l file1.txt
-rw-r--r-- 1 durga durga 77 Jan  9 12:33 file1.txt
```

Use Case:Diagram_usecase_of_touch_command_for_existing_non_empty_file



Assume that we write one shell script. The job of this shell script is to send all current date files to remote server. Assume that this script won't be executed on sat and sun. But on Monday all 3 days files have to be send.

But the problem with this script is it can recognize only current date files. To change timestamp of sat and Sunday files, we have to use touch command.

Note: We can use touch command for the following two purposes:

- 1) To create an empty file.
- 2) To change timestamp of existing file.



2) View content of the files

We can view content of the file by using the following commands

- 1) cat
- 2) tac
- 3) rev
- 4) head
- 5) tail
- 6) less
- 7) more

1. View content of the file by using cat Command:

\$ cat < file1.txt OR \$ cat file1.txt
< is optional

```
durga@durga-VirtualBox:~$ cat < file1.txt
```

This is first line

This is second line

This is third line

This is extra line

```
durga@durga-VirtualBox:~$ cat file1.txt
```

This is first line

This is second line

This is third line

This is extra line

While viewing file content we can include line numbers by using -n option.

```
durga@durga-VirtualBox:~$ cat -n file1.txt
```

- 1 This is first line
- 2 This is second line
- 3 This is third line
- 4 This is extra line

While display file content we can skip blank lines by using -b option.

```
durga@durga-VirtualBox:~$ cat -b file1.txt
```

- 1 This is first line
- 2
- 3 This is second line
- 4
- 5 This is third line
- 6
- 7
- 8 This is extra line



```
durga@durga-VirtualBox:~$ cat -b file1.txt
```

- 1 This is first line
- 2 This is second line
- 3 This is third line
- 4 This is extra line

We can view multiple files content at a time by using cat command.

```
$ cat file1.txt file2.txt file3.txt
```

Note: The word cat derived from "con'cat'enation"

Various utilities of cat Command:

- 1) To create new file with some content

```
$ cat > filename
```

```
data  
ctrl+d
```

- 2) To append some extra data to existing file

```
$ cat >> filename
```

```
extra data  
ctrl+d
```

- 3) To view content of file

```
$ cat < filename or $ cat filename
```

- 4) Copy content of one file to another file

```
$ cat input.txt > output.txt
```

- 5) To copy content of multiple files to a single file

```
$ cat file1.txt file2.txt file3.txt > file4.txt
```

- 6) Merging/append of one file content to another file

```
$ cat file1.txt >> file2.txt
```

2. tac Command:

It is the reverse of cat.

It will display file content in reverse order of lines. i.e first line will become last line and last line will become first line.

This is vertical reversal.



```
durga@durga-VirtualBox:~$ cat abc.txt
CAT
RAT
MAT
durga@durga-VirtualBox:~$ tac abc.txt
MAT
RAT
CAT
```

3. rev Command:

rev means reverse.

Here each line content will be reversed.
It is horizontal reversal.

```
durga@durga-VirtualBox:~$ cat abc.txt
CAT
RAT
MAT
durga@durga-VirtualBox:~$ rev abc.txt
TAC
TAR
TAM
```

Note:

cat command will display total file content at a time. It is best suitable for small files. If the file contains huge lines then it is not recommended to use cat command. We should go for head, tail, less and more commands.

4. head Command:

We can use head command to view top few lines of content.

* head file1.txt

- It will display top 10 lines of file1.txt.
- 10 is the default value of number of lines.

* head -n 30 file1.txt OR head -30 file1.txt

- To display top 30 lines of the file.
- Instead of 30 we can specify any number.

* head -n -20 file1.txt

To display all lines of file1.txt except last 20 lines.

* head -c 100 file1.txt

To display first 100 bytes of file content.



5. tail Command:

- We can use tail command to view few lines from bottom of the file.
- It is opposite to head command.

* tail file1.txt

Last 10 lines will be displayed.

* tail -n 30 file1.txt OR tail -30 file1.txt OR tail -n -30 file1.txt

It will display last 30 lines.

* tail -c 200 file1.txt

It will display 200 bytes of content from bottom of the file.

6. more Command:

We can use more command to view file content page by page.

* more file1.txt

- It will display first page.
- Enter → To view next line
- Space Bar → To view next page
- q → To quit/exit

* more -d file1.txt

-d option meant for providing details like
--More--(5%)[Press space to continue, 'q' to quit.]

7. less Command:

- By using more command, we can view file content page by page only in forward direction.
- If we want to move either in forward direction or in backward direction then we should go for less command.

less file1.txt

It will display first page

d → To go to next page.(d means down)

b → To go to previous page. (b means backward)



Eg: Assume a file contains enough data. Write command to display from 3rd line to 7th line.

1. Katrina Kaif
2. Kareena Kapoor
3. Karishma Kapoor
4. Sunny Leone
5. Mallika Sharawath
6. Sonakshi Sinha
7. Alia Butt
8. Pooja
9. Anushka
10. Deepika

head -7 demo.txt

1. Katrina Kaif
2. Kareena Kapoor
3. Karishma Kapoor
4. Sunny Leone
5. Mallika Sharawath
6. Sonakshi Sinha
7. Alia Butt

tail -5 demo.txt

3. Karishma Kapoor
4. Sunny Leone
5. Mallika Sharawath
6. Sonakshi Sinha
7. Alia Butt

head -7 demo.txt | tail -5

3. Karishma Kapoor
4. Sunny Leone
5. Mallika Sharawath
6. Sonakshi Sinha
7. Alia Butt

From 21st line to 30th line:

head -30 demo.txt | tail -10



Creation of Hidden files and Directories:

- If any file starts with '.', such type of file is called hidden file.
- If we don't want to display the files then we have to go for hidden files.
- Hidden files meant for hiding data. All system files which are internally required by kernel are hidden files.
- We can create hidden files just like normal files, only difference is file name should start with dot.

```
touch .securefile1.txt
```

```
cat > .securefile1.txt
```

Even by using editors also we can create hidden files.

We can create hidden directories also just like normal directories.

```
mkdir .db_info
```

Note: By using hidden files and directories we may not get full security. To make more secure we have to use proper permissions. For this we should use 'chmod' command.

Interconversion of Normal Files and Hidden Files:

Based on our requirement, we can convert normal file as hidden file and viceversa.

```
mv a.txt .a.txt
```

We are converting normal file a.txt as hidden file.

```
mv .a.txt a.txt
```

Similarly directories also

```
mv dir1 .dir1
```

```
mv .dir1 dir1
```

Copying of Files:

```
cp file1.txt file2.txt
```

If file2.txt not available, then file2.txt will be created and the content will be copied.

If file2.txt is already available and contains some data then that data will be overwritten with file1.txt data.

Before overwriting if we want confirmation, then we should go for -i option.

i means interactive.

```
cp -i file1.txt file2.txt
```

```
$ cp -i a.txt b.txt
```



cp: overwrite 'b.txt'?

If we want verbose output then we can use -v option.

cp -v file1.txt file2.txt

\$ cp -v a.txt b.txt

'a.txt' -> 'b.txt'

cp file1.txt file2.txt file3.txt file4.txt output

Note: To copy multiple files content to the single file, we should not use cp command. we should use cat command.

cp a.txt b.txt c.txt d.txt → Invalid

cp: target 'd.txt' is not a Directory

cat a.txt b.txt c.txt > d.txt → Valid

Moving and Renaming Directories:

Both moving and renaming activities can be performed by using single command: mv

1) Renaming of Files:

- mv oldname newname
- mv file1.txt file2.txt
- file1.txt will be renamed to file2.txt

2) Renaming of Directories:

- mv dir1 dir2
- dir1 will be renamed to dir2

3) Moving of files from one directory to another directory:

- mv dir1/* dir2
- All files of dir1 will be moved to dir2. After executing this command, dir1 will come empty.

4) Move total directory to another directory:

- mv dir1 dir2
- dir1 will be moved to dir2
- In the case of overwriting, if we want confirmation alert then we can use -i option with mv command.
- \$ mv -i a.txt d.txt dir1
- mv: overwrite 'dir1/a.txt'?



Topic-8: Comparing Files

We can compare data of two files by using the following commands:

- 1) `cmp`
- 2) `diff`
- 3) `sdiff`
- 4) `vidiff`
- 5) `comm`

1) cmp Comamnd:

It will compare byte by byte.

```
cmp file1.txt file2.txt
```

If content is same then we won't get any output.

If the content is different, then it provides information about only first difference. byte number and line number will be provided.

```
$ cmp a.txt c.txt
```

```
a.txt c.txt differ: byte 7, line 2
```

Note: `cmp` command won't show all differences and show only first difference.

2) diff Comamnd:

It will show all differences in the content.

```
diff file1.txt file2.txt
```

If the content is the same then no output.

If the content is different then it will show all differences.

```
$ diff a.txt b.txt
```

```
$ diff a.txt c.txt
```

```
2,3c2,3
```

```
< Bunny
```

```
< Chinny
```

```
---
```

```
> bunny
```



```
> chinny  
b
```

For the diff command we can use the following options.

- q shows message when files are different.
- s shows message when files are same | identical
- y shows comparison line by line (parallel comparison)

```
$ diff -q a.txt c.txt  
Files a.txt and c.txt differ  
$ diff -s a.txt b.txt  
Files a.txt and b.txt are identical  
$ diff -y a.txt c.txt  
Sunny  
Bunny  
Chinny  
Vinny  
Pinny
```

```
Sunny  
bunny  
chinny  
Vinny  
Pinny
```

If we want to suppress common lines then we should use --suppress-common-lines option with -y option.

```
$ diff -y --suppress-common-lines a.txt c.txt  
Bunny  
Chinny
```

```
| bunny  
| chinny
```

3) sdiff Command:

We can use sdiff command for side by side comparison (parallel comparison)

```
$ sdiff a.txt b.txt  
Sunny  
Bunny  
Chinny  
Vinny  
Pinny  
$ sdiff a.txt c.txt  
Sunny  
Bunny  
Chinny  
Vinny  
Pinny
```

```
Sunny  
Bunny  
Chinny  
Vinny  
Pinny  
  
bunny  
chinny  
Vinny  
Pinny
```

Note: sdiff comamnd and diff command with -y option are same.



4) vimdiff Command:

- It will highlight differences in vim.
- To support this command, we have to install vim by using the following command.
- `sudo apt install vim`
- `vimdiff a.txt b.txt`
- `ctrl+w+w` → To go to next window
- `:q` → Close current window
- `:qa` → Close all windows
- `:qa!` → Close all windows forcibly.

5) comm Command:

By using this command we can compare data of two files.

```
comm file1.txt file2.txt
```

It display results in 3 columns

column-1: Data present only in file1.txt but not in file2.txt

column-2: Data present only in file2.txt but not in file1.txt

column-3: Common data of both files.

```
$ comm a.txt c.txt
      Sunny
bunny
Bunny
      chinny
Chinny
      Vinny
      Pinny
```

With comm command we can use the following options

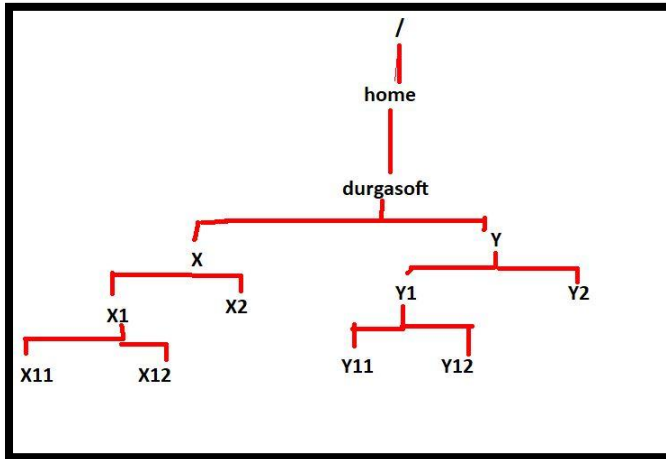
- 1 → If we don't want to display column-1
- 2 → If we don't want to display column-2
- 3 → If we don't want to display column-3
- 12 → If we don't want to display columns 1 and 2

Note: We can compare files from various builds by using our comparison commands (cmp, diff, sdiff, vimdiff, comm).



RG → RA → D → C → Testing → Production Build-1 released Test.java
RG → RA → D → C → Testing → Production Build-2 released Test.java

Case Study:



```
mkdir -p x/x1/x1{1,2} x/x2 y/y1/y1{1,2} y/y2
```

```
durgasoft@durgasoft-VirtualBox:~$ cat > x/x1/x11/file1.txt
```

This is file1 content

This is file1 content

Requirement-1: assume file1.txt is available inside x11 directory and we are in user home directory. copy this file to y2 directory.

```
cp /home/durgasoft/x/x1/x11/file1.txt /home/durgasoft/y/y2
```

```
cp ~/x/x1/x11/file1.txt ~/y/y2
```

```
cp x/x1/x11/file1.txt y/y2
```

Requirement-2: assume file1.txt is available inside x11 directory and we are in user home directory. move this file to y11 directory.

```
mv /home/durgasoft/x/x1/x11/file1.txt /home/durgasoft/y/y1/y11
```

```
mv ~/x/x1/x11/file1.txt ~/y/y1/y11
```

```
mv x/x1/x11/file1.txt y/y1/y11
```

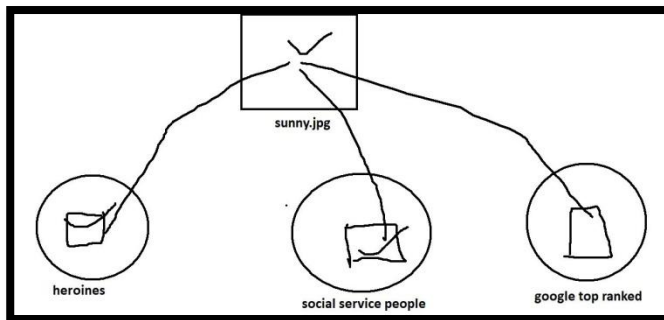


Topic-9: Creation of Link Files

There are 2 types of link files

- 1) Hard Link files
- 2) Soft Link files

1) Hard Link Files:



It is just another name of the same exact file.
We can create hard link file by using `ln` command.
`ln originalfile hardlinkfile`

Eg: `ln file1.txt file2.txt`

Here file1.txt is original file and file2.txt is hard link file.

Important conclusions about hard link file:

- 1) Both original file and hardlink file have same inode number, same size, same timestamp.
- 2) If we delete original file, then there is no effect on hardlink file.

2) Soft Link File:

- A softlink is a pointer to another file. It is just like windows shortcut.
- It is also known as symbolic link.
- We can create soft link file by using `ln` command but with `-s` option.
- `ln -s originalfile softlinkfile`

Eg: `ln -s file1.txt file2.txt`

Here file1.txt is original file and file2.txt is link file.



Important conclusions about softlink file:

- 1) Original file and softlink file have different inode numbers, different file sizes and different timestamps.
- 2) Usually softlink file has smaller file size than original file size.
- 3) If we delete original file then softlink files will become useless.

Link files for directories:

We cannot create hardlink for directories because it breaks Linux File System. Having two root directories is meaningless.

```
$ ln dir1 dir2
```

In: dir1: hard link not allowed for directory

We can create softlink for directories

```
$ ln -s dir1 dir2
```

Note: For files we can create both hard and soft links. But for directories we can create only softlinks but not hardlinks.

Case Study:

Assume dir1 contains dir2.

dir2 contains softlink dir3 pointing to dir1

dir1 → dir2 → dir3

```
$ mkdir -p dir1/dir2
```

```
$ cd dir1/dir2
```

```
$ ln -s ~/Desktop/dir1 dir3
```

It will form a loop.

Note: While creating link files there may be a chance of forming loops. Take a bit special care.

Q. Which of the following is valid about hard link?

- A) Inode number is different when compared to that of original file.
- B) File Size is different when compared to that of original file.
- C) It will become useless if we delete original file.
- D) Inode number, file size and timestamp are same when compared to that of original file.

Ans: D



Q. Which of the following is valid way to create softlink for sunny.jpg present in Pictures directory?

- A) In ~/Pictures/sunny.jpg newimg.jpg
- B) In -s ~/Pictures/sunny.jpg newimg.jpg
- C) In newimg.jpg ~/Pictures/sunny.jpg
- D) In -s newimg.jpg ~/Pictures/sunny.jpg

Ans: B

Q. We can create both hard and soft links to the directories. Is it valid?

- A) True
- B) False

Ans: B

Note: If we perform any change to the content of original file, then these changes will be reflected to the link file. Similarly, if we perform any change to the link file, then those changes will be reflected to the original file. This is true for both hard and soft links.



Topic-10: Word Count Command (wc Command)

We can use wc command to count number of lines, words and characters present in the given file.

```
wc filename  
no_of_lines no_of_words no_of_characters filename
```

Eg:

```
$ wc a.txt  
4 26 166 a.txt
```

4 → Number of Lines
26 → Number of words
166 → Number of characters (File size in bytes)

We can use the following options with wc Command

- l → To print only number of lines
- w → To print only number of words
- c → To print only number of characters
- lw → To print only number of lines and words
- lc → To print only number of lines and characters
- wc → To print only number of words and characters
- L → To print number of characters present in Longest Line.

```
$ wc -l a.txt  
4 a.txt  
$  
$ wc -w a.txt  
26 a.txt  
$ wc -c a.txt  
166 a.txt  
$ wc -lw a.txt  
4 26 a.txt  
$ wc -lc a.txt  
4 166 a.txt  
$ wc -wc a.txt  
26 166 a.txt
```



```
$ wc -L a.txt  
57 a.txt
```

We can use wc command for multiple files simultaneously.

```
$ wc a.txt b.txt c.txt  
 4 26 166 a.txt  
 3  4  27 b.txt  
 4  4 112 c.txt  
11 34 305 total
```



Topic-11: Sorting Content of the File

We can sort data of the file by using sort command.

`sort filename`

Here sorting is based on alphabetical order.

```
$ cat a.txt
```

```
Sunny  
Bunny  
Chinny  
Vinny  
Pinny
```

```
$ sort a.txt
```

```
Bunny  
Chinny  
Pinny  
Sunny  
Vinny
```

If we want to sort based on reverse of alphabetical order, then we should use `-r` option.

```
$ sort -r a.txt
```

```
Vinny  
Sunny  
Pinny  
Chinny  
Bunny
```

If the file contains alphanumeric data, then first numbers will be considered and then alphabet symbols.

```
$ cat a.txt
```

```
7  
Sunny  
8  
Bunny  
1
```



Chinny

6

Vinny

5

Pinny

\$ sort a.txt

1

5

6

7

8

Bunny

Chinny

Pinny

Sunny

Vinny

\$ sort -r a.txt

Vinny

Sunny

Pinny

Chinny

Bunny

8

7

6

5

1

If the file contains only numbers, then the sorting is not based on numeric value and it is just based on digits.

\$ cat > a.txt

11

2

7

2222222

9

\$ sort a.txt

11

2

2222222

7

9



If we want to sort based on numeric value then we have to use -n option.

-n means numeric value

```
$ sort -n a.txt
```

```
2
```

```
7
```

```
9
```

```
11
```

```
2222222
```

By default sort command will display duplicate lines. If we want only unique lines then we have to use -u option.

-u meant for unique lines.

```
$ cat a.txt
```

```
1
```

```
1
```

```
2
```

```
2
```

```
Sunny
```

```
Sunny
```

```
Bunny
```

```
$ sort a.txt
```

```
1
```

```
1
```

```
2
```

```
2
```

```
Bunny
```

```
Sunny
```

```
Sunny
```

```
$ sort -u a.txt
```

```
1
```

```
2
```

```
Bunny
```

```
Sunny
```

Note: We can use -ur and -un options also.

To play with unique data, there is special command available: uniq

Q. Assume a.txt contains the following data

```
Sunny
```

```
Bunny
```

```
Chinny
```

```
Vinny
```



Without using -r option with sort command, sort the content based on reverse of alphabetical order and store the result inside sorted.txt?

```
sort a.txt | tac > sorted.txt  
sort -r a.txt > sorted.txt
```

To remove duplicate lines also:

```
sort -u a.txt | tac > sorted.txt  
sort -ru a.txt > sorted.txt
```

Sorting Tabular Data by using -k option:

-k means KEYDEF (key definition). Based on which key (column) we have to sort.

```
$ ls -l /etc | head -10  
total 1068  
drwxr-xr-x 3 root root 4096 Aug 6 00:34 acpi  
-rw-r--r-- 1 root root 3028 Aug 6 00:28 adduser.conf  
drwxr-xr-x 2 root root 4096 Nov 7 19:47 alternatives  
-rw-r--r-- 1 root root 401 May 29 2017 anacrontab  
-rw-r--r-- 1 root root 433 Oct 2 2017 apg.conf  
drwxr-xr-x 6 root root 4096 Aug 6 00:30 apm  
drwxr-xr-x 3 root root 4096 Aug 6 00:33 apparmor  
drwxr-xr-x 8 root root 4096 Nov 7 06:32 apparmor.d  
drwxr-xr-x 4 root root 4096 Nov 7 06:33 apport
```

Sort based on file size in ascending order:

```
$ ls -l /etc | head -10 | sort -k 5n  
total 1068  
-rw-r--r-- 1 root root 401 May 29 2017 anacrontab  
-rw-r--r-- 1 root root 433 Oct 2 2017 apg.conf  
-rw-r--r-- 1 root root 3028 Aug 6 00:28 adduser.conf  
drwxr-xr-x 2 root root 4096 Nov 7 19:47 alternatives  
drwxr-xr-x 3 root root 4096 Aug 6 00:33 apparmor  
drwxr-xr-x 3 root root 4096 Aug 6 00:34 acpi  
drwxr-xr-x 4 root root 4096 Nov 7 06:33 apport  
drwxr-xr-x 6 root root 4096 Aug 6 00:30 apm  
drwxr-xr-x 8 root root 4096 Nov 7 06:32 apparmor.d
```



Sort based on Month:

6th column provides total date. If we want consider only month then we should use M.

```
$ ls -l /etc | head -10 | sort -k 6M
total 1068
-rw-r--r-- 1 root root 401 May 29 2017 anacrontab
drwxr-xr-x 3 root root 4096 Aug 6 00:33 apparmor
drwxr-xr-x 3 root root 4096 Aug 6 00:34 acpi
drwxr-xr-x 6 root root 4096 Aug 6 00:30 apm
-rw-r--r-- 1 root root 3028 Aug 6 00:28 adduser.conf
-rw-r--r-- 1 root root 433 Oct 2 2017 apg.conf
drwxr-xr-x 2 root root 4096 Nov 7 19:47 alternatives
drwxr-xr-x 4 root root 4096 Nov 7 06:33 apport
drwxr-xr-x 8 root root 4096 Nov 7 06:32 apparmor.d
```

Sort based on number of links in descending order:

```
$ ls -l /etc | head -10 | sort -k 2nr
total 1068
drwxr-xr-x 8 root root 4096 Nov 7 06:32 apparmor.d
drwxr-xr-x 6 root root 4096 Aug 6 00:30 apm
drwxr-xr-x 4 root root 4096 Nov 7 06:33 apport
drwxr-xr-x 3 root root 4096 Aug 6 00:33 apparmor
drwxr-xr-x 3 root root 4096 Aug 6 00:34 acpi
drwxr-xr-x 2 root root 4096 Nov 7 19:47 alternatives
-rw-r--r-- 1 root root 3028 Aug 6 00:28 adduser.conf
-rw-r--r-- 1 root root 401 May 29 2017 anacrontab
-rw-r--r-- 1 root root 433 Oct 2 2017 apg.conf
```




Topic-12: Find Unique Content in the File by using **uniq** Command

We can use **uniq** command to display unique content in the file.

But to use **uniq** command, compulsory the file should be sorted, otherwise it won't work properly.

```
$ cat a.txt
```

```
Sunny  
sunny  
Bunny  
Chinny  
Sunny  
Bunny  
Chinny
```

```
$ uniq a.txt
```

```
Sunny  
sunny  
Bunny  
Chinny  
Sunny  
Bunny  
Chinny
```

```
$ sort a.txt | uniq
```

```
Bunny  
Chinny  
sunny  
Sunny
```

With **uniq** command we can use multiple options:

- d → To display only duplicate lines
- c → To display number of occurrences of each line
- i → Ignore case while comparing
- u → To display only unique lines i.e the lines which are not duplicated.



1. To display only duplicate lines:

```
$ sort a.txt | uniq -d
```

Bunny

Chinny

Sunny

2. To display number of occurrences of each line:

```
$ sort a.txt | uniq -c
```

2 Bunny

2 Chinny

1 sunny

2 Sunny

3. To ignore case while comparing:

```
$ sort a.txt | uniq -i
```

Bunny

Chinny

sunny

```
$ sort a.txt | uniq -ic
```

2 Bunny

2 Chinny

3 sunny

4. To display only unique lines i.e the lines which are not duplicated:

```
$ sort a.txt | uniq -u
```

sunny



Topic-13: Input and Output of Commands and Redirection



Commands can take input, perform required operation and produces some output. While executing command if anything goes wrong then we will get error message.

Command can take input either from standard Input or from command line arguments. Command will produce results to either Standard output or Standard Error.

Standard Input, Standard Output and Standard Error are Data Streams and can flow from one place to another place. Hence redirection and piping are possible.

Command Line arguments are static and these are not streams. Hence redirection and piping concepts are not applicable to command line arguments.

These data streams are associated with some numbers.

Standard Input associated with 0.

Standard Output associated with 1.

Standard Error associated with 2.

By default Standard input connected with keyboard, Standard output and Standard Error connected with Terminal. But we can redirect.

Standard Input from the keyboard and output to Standard Output

Device:

\$cat

read required input from the keyboard

this data will be displayed to the standard output.

ctrl+d

Note: For the cat command if we are not providing any arguments, then the input will be taken from standard input device (keyboard) and display the output to the standard output device (Terminal).



```
$ cat
```

This is data provided from Standard Input

This is data provided from Standard Input

Input from command line arguments and error messages to the Standard Error:

```
$ rm file100
```

```
rm: cannot remove 'file100': No such file or directory
```

We are providing filename as command line argument to the rm command. Specified file not available and hence this command will produce error message to the Standard Error device (Terminal).

Note: Some commands may accept Standard Input and Some commands may accept command line arguments.

- 1) rm command will always accept command line arguments only.
rm file1 file2
- 2) echo command will always accept command line arguments only.
echo "durgasoft"
- 3) cat command can accept input either from standard input or from command line arguments.

Redirection

As Standard Input, Standard Output and Standard Error are Data streams, we can redirect these streams.

Redirecting Standard Output:

We can redirect standard output by using > and >> symbols.

> will perform overwriting of existing data

>> will perform appending to existing data

Eg 1: To redirect the standard output of cat command from terminal to output.txt

```
$ cat 1> output.txt
```

```
sample data
```

```
ctrl+d
```

sample data won't be displayed to the terminal and will write to output.txt

Redirection symbol > is always associated with 1 by default. Hence we are not required to specify 1 explicitly.



```
$cat > output.txt  
sample data  
ctrl+d
```

Instead of overwriting if we want to perform appending then we should use >>.

Redirecting Standard Error:

We can redirect error messages from the terminal to our own file by using > and >> symbols.

```
$ cal 34 w3892384208342 2>> error.txt
```

Now error message won't be displayed to the console and will be written to error.txt.
For error redirection 2 is mandatory.

Redirecting Standard Input:

We can redirect standard input from keyboard to our required file.
We can perform input redirection by using < symbol.

```
$cat 0< a.txt 1>>output.txt 2>>error.txt
```

< symbol is always associated with 0 by default. Hence we can remove.

```
$cat < a.txt >>output.txt 2>>error.txt
```

***Note: To redirect both standard output and standard error to the same destination we can use shortcut as follows

```
$ cat < a.txt &> output.txt  
&> means both standard output and standard error.
```

Redirecting Standard output from one terminal to another terminal:

In unix every thing is treated as file even our terminal also.
We can find terminal related file by using tty command.

terminal - 2:

```
$ tty  
/dev/pts/1
```

terminal - 1:

```
$ls -l 1> /dev/pts/1
```

terminal-1 long listing output will be displayed to terminal-2



Bits

Q. In how many ways command can get input?

2 ways. Either from Standard Input or from command line arguments.

Q. Which of the following contains Data Streams?

- A) Standard Input
- B) Standard Output
- C) Standard Error
- D) Command Line arguments

Ans: A,B,C

Q. By default Standard Input connected to

- A) Terminal
- B) Keyboard
- C) A File

Ans: B

Q) By default standard Output connected to:

Terminal

Q) By default standard error connected to:

Terminal

Q) What number represents Standard Input Stream?

0

Q) What number represents Standard Output Stream?

1

Q) What number represents Standard Error?

2

Q) How we can redirect standard output of the ls command to a file called output.txt?

- A) ls 2> output.txt
- B) ls 0< output.txt
- C) ls 1< output.txt
- D) ls 1> output.txt

Ans: D



ls 2> output.txt → redirecting standard error from terminal to output.txt
ls 0< output.txt → redirecting standard input from keyboard to output.txt
ls 1< output.txt → Meaningless

Q) How we can redirect the standard output of the ls command to output.txt, but at the same time, redirect standard error to error.txt?

ls 1> output.txt 2> error.txt
ls > output.txt 2> error.txt

Q. Explain the difference between <,>,>> in redirection?

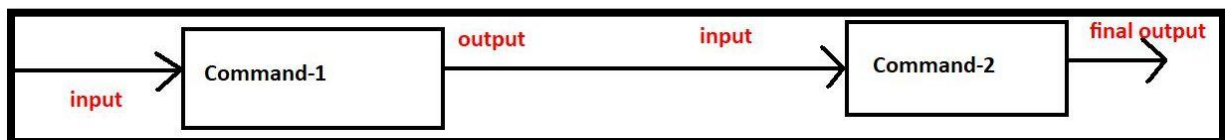
< symbol meant for input redirection
> symbol meant for output redirection where the existing data will be overwritten.
>> symbol meant for output redirection where the data will be appended instead of overwriting.



Topic-14: Piping

Sometimes we can use output of one command as input to another command. This concept is called piping.

By using piping, multiple commands will work together to fulfill our requirement.



We can implement piping by using vertical bar (|).

```
$ ls -l /etc | wc
215 1940 11872
```

First ls got executed and the output of this command will become input to wc command.

Eg 2: \$ ls -l /etc | more

Eg 3: \$ ls -l /etc | wc | wc -l
The output is: 1

Eg 4: \$ ls -l /etc | head -5

Note: instead of ls -l we can use ll command, most of linux flavours provides support.

tee Command:

Requirement:

The output of the ls command should be saved to output.txt and should be provided as input to wc command:

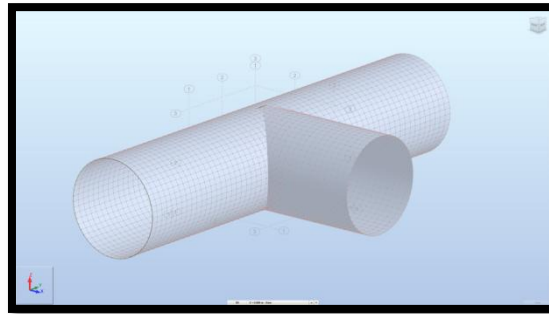
```
ls -l 1>output.txt | wc
```

This command won't work because if we are using redirection in the middle of piping, it will break piping concept.

In piping, if we want to save the output of one command to a file and if we want to pass that output as input to next command simultaneously, then we should go for tee command.



tee command is just like T-Junction or T-Pipe. It will take one input but provides two outputs.

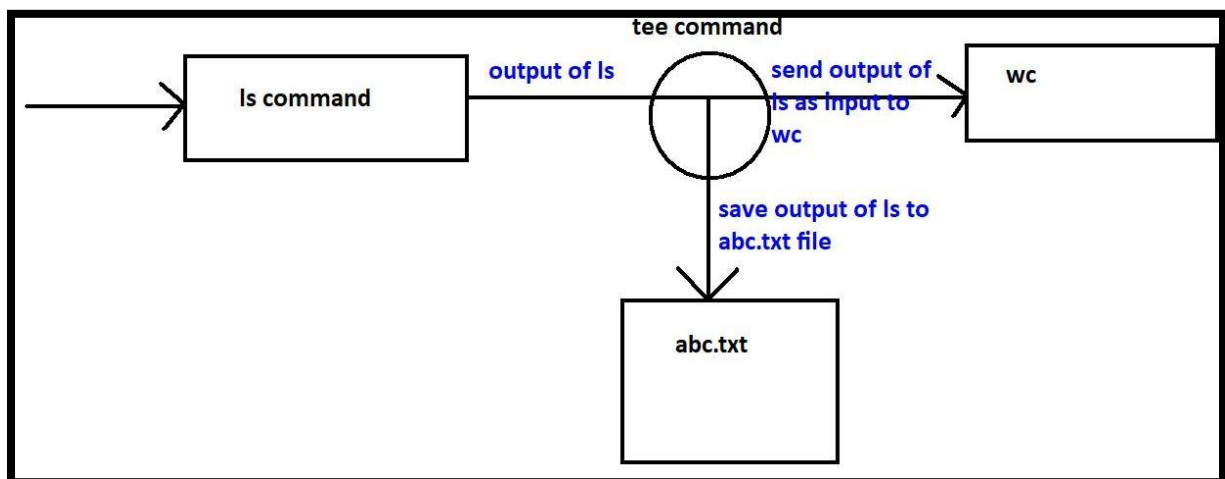


Eg 1: To save the output of ls command to a file and to display to the terminal simultaneously.

\$ ls -l → It will display to the terminal

\$ ls -l > abc.txt → It will save to the abc.txt but won't display to the terminal.

\$ ls -l | tee abc.txt



Eg 2: To save the output of ls command to a file and send that output as input to wc command

\$ ls -l | tee output.txt | wc -l



xargs Command:

Q. Display the output of date command by using echo command with piping concept?

\$ date | echo → It won't work because the output of date command is stream, but echo command will accept only command line arguments but not stream.

\$ date | xargs echo → xargs command will convert the output stream of date command into command line arguments and these arguments will be passed as input to echo command.

Hence the job of xargs command is to convert output stream into command line arguments

Eg 1: Assume input.txt contains file names. Each file contains some data.

Read file names from the input.txt, write total content to output.txt and display the total number of lines present in output.txt.

```
$ cat input.txt | xargs cat | tee output.txt | wc -l
```

Eg 2: Assume input.txt contains file names. Read file names from the input.txt and remove all these files.

```
$ cat input.txt | xargs rm
```

Assignment:

list out all contents of /dev folder and save to file1.txt.

list out all contents of /bin folder and save to file2.txt.c

Write a single pipeline for the following requirement:

read content of file1 and file2, save to fil3.txt. By using sort command reverse contents of file3.txt and save to sorted.txt.

```
$ ls /dev > file1.txt
```

```
$ ls /bin > file2.txt
```

```
$ cat file1.txt file2.txt | tee file3.txt | sort -r > sorted.txt
```

Q. What is piping?

A) A way of connecting commands together.

B) A way of passing data from the standard output of one command to the standard input of another command.

Ans: A,B



Q. How to pipe standard output of X command to the standard input of Y command?

- A) X > Y
- B) X >> Y
- C) X < Y
- D) X | Y

Ans: D

Q. How we can use tee command when piping together commands A,B and C to save output of B command to results.txt.

A | B | tee results.txt | C

Q. How to pipe data from command A to command B, but B won't accept standard input and accepts only command line arguments.

A | xargs B



Topic-15: How to Use Multiple Commands in a Single Line

We can execute multiple independent commands in a single line by using the following two ways

1st Way: By using semicolon (;)

`cmd1;cmd2;cmd3;.....;cmdn`

First cmd1 will be executed and then cmd2 followed by rest of the commands.
If any command fails in the middle, still rest of the commands will be executed.

2nd Way: By using &&

`cmd1 && cmd2 && cmd3 &&..... && cmdn`

First cmd1 will be executed and then cmd2 followed by rest of the commands.
If any command fails in the middle, then rest of the commands won't be executed.

Eg:

create a directory dir1
create files a.txt,b.txt,c.txt in that dir1
write current system date and time to a.txt
write current month calendar to b.txt

`mkdir dir1 ; touch dir1/{a,b,c}.txt ; date > dir1/a.txt ; cal > dir1/b.txt`
`mkdir dir1 && touch dir1/{a,b,c}.txt && date > dir1/a.txt && cal > dir1/b.txt`

`mkdir dir1 ; touch dir1/{a,b,c}.txt ; Date > dir1/a.txt ; cal > dir1/b.txt`
Here 3rd command fails, but still 4th command will be executed.

`mkdir dir1 && touch dir1/{a,b,c}.txt && Date > dir1/a.txt && cal > dir1/b.txt`
Here 3rd command fails, and hence 4th command won't be executed.



Topic-16: Regular Expressions and Wildcard Characters

If we want to represent a group of strings according to a particular pattern, then we should go for regular expressions.

By using wildcard characters, we can build regular expressions.

A wildcard character can be used as a substitute for required sequence of characters in the regular expression.

- 1) \rightarrow Represents zero or more characters
- 2) $?$ \rightarrow Represents only one character
- 3) $[]$ \rightarrow Range of characters
- 4) $[abc]$ \rightarrow Either a or b or c
- 5) $[!abc]$ \rightarrow Any character except a,b and c
- 6) $[a-z]$ \rightarrow Any lower case alphabet symbol
- 7) $[A-Z]$ \rightarrow Any upper case alphabet symbol
- 8) $[a-zA-Z]$ \rightarrow Any alphabet symbol
- 9) $[0-9]$ \rightarrow Any digit from 0 to 9
- 10) $[a-zA-Z0-9]$ \rightarrow Any alphanumeric character
- 11) $[!a-zA-Z0-9]$ \rightarrow Except alpha numeric character (i.e special symbol)
- 12) $[[:lower:]]$ \rightarrow Any lower case alphabet symbol
- 13) $[[:upper:]]$ \rightarrow Any upper case alphabet symbol
- 14) $[[:alpha:]]$ \rightarrow Any alphabet symbol
- 15) $[[:digit:]]$ \rightarrow Any digit from 0 to 9
- 16) $[[:alnum:]]$ \rightarrow Any alpha numeric character
- 17) $[![:digit:]]$ \rightarrow Any character except digit
- 18) $\{\}$ \rightarrow List of files with comma separator

- 1) To list out all files present in current working directory $\rightarrow \$ ls *$
- 2) To list out all files with some extension $\rightarrow \$ ls *.*$
- 3) To list out all files starts with a $\rightarrow \$ ls a*$
- 4) To list out all files starts with a and ends with t $\rightarrow \$ ls a*t$
- 5) To list out all .java files $\rightarrow \$ ls *.java$
- 6) To list out all files where file name contains only 2 characters and first character should be 'a' $\rightarrow \$ ls a?$



- 7) To list out all files where file name contains only 3 characters → `$ ls ???`
 - 8) To list out all files where file name contains atleast 3 characters → `$ ls ???*`
 - 9) To list out all files where file name starts with a or b or c → `$ ls [abc]*`
 - 10) To list out all files where file name should not starts with a, b and c → `$ ls [!abc]*`
 - 11) To list out all files starts with lower case alphabet symbol
`$ ls [a-z]*` OR `$ ls [[:lower:]]*`
 - 12) To list out all files starts with upper case alphabet symbol
`$ ls [A-Z]*` OR `$ ls [[:upper:]]*`
 - 13) To list out all files starts with digit.
`$ ls [0-9]*` OR `$ ls [[:digit:]]*`
 - 14) To list out all files where first letter should be upper case alphabet symbol, second letter should be digit and third letter should be lower case alphabet symbol.
`$ ls [[:upper:]][[:digit:]][[:lower:]]`
 - 15) To list out all files starts with special symbol
`$ ls [![:alnum:]]*`
 - 16) To list out all files with .java and .py extension
`$ ls {*.java, *.py}`
- Note:** We can use these wildcard characters with the following commands also.
cp, mv, rm
- 17) To copy all files starts with digit to dir1 directory.
`$ cp [[:digit:]]* dir1`
`$ cp [0-9]* dir1`
 - 18) To move all files starts with alphabet symbol and with .txt extension to dir2 directory?
`$ mv [[:alpha:]]*.txt dir2`
 - 19) Remove all files starts with a or b or c and ends with e or t.
`$ rm [abc]*[et]`



Q1. Which of the following command will list all files that has exactly three characters present in current working directory?

- A) ls ***
- B) ls ???
- C) ls !!!
- D) ls &&&

Ans: B

Q2. Which of the following commands will copy all files that ends with .pdf to dir1?

- A) cp ?.pdf dir1
- B) cp .pdf* dir1
- C) cp *.pdf dir1

Ans: C

Q3. Which of the following command will move all the files that begins with letter a and ends with letter n to dir1?

`$mv a*n dir1`

Q4. Which of the following commands will display contents of all files that begins with a digit and ends with letter a or e or i or o or u?

`$cat [[:digit:]]*[aeiou]`
`$cat [0-9]*[aeiou]`

Q5. Which of the following commands will list all files that begins with a lower case alphabet symbol and has a letter d in the third character position, and ends with an upper case letter?

`$ls [[:lower:]]?d*[[:upper:]]`

Q6. Which of the following command will list all .jpg files present in Pictures directory?

`$ls /home/durgasoft/Pictures/*.jpg`
`$ls ~/Pictures/*.jpg`



Q7. Which of the following regular expressions will match the file named with demoA.txt ?

- A) *
- B) demo?.txt
- C) demo*
- D) *.txt
- E) demo[A-Z].txt
- F) All of these

Q8. Which of the following regular expressions can match the files?

student_reportA.pdf
student_reportB.pdf
student_reportC.pdf

- A) ?? .pdf
- B) report*.pdf
- C) *[A-Z].pdf
- D) student*.pdf

Ans: C, D



Topic-17: Command Aliasing

Alias means other alternative name or nickname.

We can give our own more convenient nicknames for unix commands. This concept is called command aliasing.

Note: we can use type command, to check whether the command is already available or not.

How to create alias names:

```
$ alias nickname='original command'
$ alias nickname="original command"
```

After aliasname space is not allowed. Hence the following are invalid

```
alias nickname ='original command'
alias nickname= 'original command'
alias nickname = 'original command'
```

How to list all available aliases:

By using alias command without any arguments

```
$alias
```

How to remove alias names:

By using unalias command.

```
$unalias alias_name
```

Where we can use aliasing:

1. If any lengthy command repeatedly required, then we can create shortcut alias name and we can use that short alias name every time.

```
alias d20f='mkdir dir1;touch dir1/file{1..20}.txt'
```

```
$d20f
```



Eg: To list out all files present in current working directory, save this data to output.txt and display the number of lines to the terminal. Define alias name 'current' for this total activity.

```
alias current='ls -l | tee output.txt | wc -l'
```

2. To use other operating system (like windows) commands directly in linux

```
alias cls='clear'
alias rename='mv'
```

3. To handle typing mistakes

```
alias grpe='grep'
```

4. To handle language barriers:
In Germany datum means date.

```
alias datum='date'
```

How to persist aliases permanently?

Whatever aliases we created, are by default available only in the current session. Once we close the terminal, all aliases will be lost.

But we can make our created aliases permanently in our system by using the following 2 ways:

1st Way:

We have to define our aliases in .bashrc file present in our home directory.

```
gedit .bashrc
```

Add the following lines in that file.

```
# myown aliases
alias cls='clear'
alias ddd='date;date;date'
```

Note: To reflect these aliases, compulsory we have to close and open terminal.

2nd Way:

Instead of editing .bashrc file, we can create our own file to maintain our defined aliases. The name of the file should be .bash_aliases and should be present in home directory.



.bash_aliases

```
alias ccc='cal;cal;cal'
```

```
alias ct='cal;date'
```

Note: To reflect these aliases, compulsory we have to close and open terminal.

Q. What is the purpose of alias command?

To list out all available aliases and to create new alias.

Q. How to use unalias command?

unalias alias_name → To remove a particular alias

unalias -a → To remove all aliases

Q. To persist aliases, in which file we have to define aliases?

.bashrc OR .bash_aliases

Q. Which of the following is valid way of creating alias?

A) alias rename ="mv"

B) alias rename= "mv"

C) alias rename = "mv"

D) alias rename="mv"

Ans: D

While creating aliases, we should not give space.



Topic-18: Locate and Find Commands

locate Command:

We can use locate command to locate files and directories in our system.

Internally locate command will search in the database for the required files and directories and returns the results.

As locate command is searching in the database instead of filesystem, performance will be improved.

1. To locate all .jpg files
\$ locate *.jpg

To ignore case:

By default locate command will consider case. If we want to ignore case, we have to use -i option.

```
$locate -i *.jpg
```

Note: In ubuntu -i option is not working

We can limit the number of lines in the result by using --limit option.

```
$ locate --limit 5 *.conf
```

```
/etc/adduser.conf  
/etc/apg.conf  
/etc/appstream.conf  
/etc/brltty.conf  
/etc/ca-certificates.conf
```

It display only 5 lines.

Before display results, to check whether the file exists or not , we have to use -e option or --existing option.

```
$locate --existing *.jpg
```



Reason:

locate command using database to find results. This database will be updated only once per day by default. After updating database, some files may be deleted. Hence before printing results, to check whether files are existing or not, we have to use -e or --existing option.

Before displaying results, to check whether symbolic links pointing to original files or not, we have to use -L or --follow option. i.e if we use --follow option, broken symbolic links won't be displayed in the output.

```
$locate --follow *.txt
```

Note: We can use all these options together

```
$locate --existing --follow -i --limit 5 *.conf | wc -l
```

observe the difference in results:

```
$locate *.conf | wc -l
```

```
$locate -i *.conf | wc -l
```

```
$locate --existing -i 5 *.conf | wc -l
```

```
$locate --existing --follow -i *.conf | wc -l
```

```
$locate --existing --follow -i --limit 10 *.conf | wc -l
```

How to update database?

We can see the database by using locate command with -S option.

```
$ locate -S
```

Database /var/lib/mlocate/mlocate.db:

39,527 directories

3,69,078 files

2,42,42,520 bytes in file names

93,40,974 bytes used to store database

This database will be updated only once per day. If we are creating or removing files and directories, to reflect these changes we have to update database explicitly by using updatedb command. But admin privileges must be required.

```
$ sudo updatedb
```

[sudo] password for durgasoft:

Q1. Which of the following are valid options to locate command?

- A) -i
- B) --limit
- C) -L
- D) -e
- E) --existing



- F) --follow
G) All of these

Ans: G

Q. locate command internally uses database to find results?

- True
False

Ans: True

Q. We created a file called demo.txt. But it has been 2 hours since locate database has been updated. Is locate command able to find this demo.txt?

- A) Yes
B) No

Ans: No

Q. How to update database which is used by locate command?

- A) update locatedb
B) updatedb
C) sudo updatedb

Ans: C

find Command:

We can use find command to find files and directories present in our system. It provides more search options when compared with locate command like

1. Search only files
 2. Search only directories
 3. Search by name
 4. Search by size
 5. We can use search results automatically for some other commands
 6. We can restrict maxdepth
- etc

1. \$find

It will find all files and directories in current working directory and below in linux file system. This is the default behaviour.

2. We can find all files and directories in the specified directory and below.

```
$ find /dev  
$ find /etc
```



3. maxdepth Option:

usually find command will search in all depth levels. But we can specify the required depth level by using maxdepth option.

Desktop

```
| -file1.txt
| -level_1_dir
|   | -file2.txt
|   | -level_2_dir
|   |   | -file3.txt
|   |   | -level_3_dir
|   |   |   | -file4.txt
|   |   |   | -level_4_dir
|   |   |   |   | -file5.txt
```

```
$mkdir -p level_1_dir/level_2_dir/level_3_dir/level_4_dir
$touch file1.txt level_1_dir/file2.txt level_1_dir/level_2_dir/file3.txt
level_1_dir/level_2_dir/level_3_dir/file4.txt
level_1_dir/level_2_dir/level_3_dir/level_4_dir/file5.txt
```

Observe the difference in results by executing the following commands:

1. \$ find . -maxdepth 1
2. \$ find . -maxdepth 2
3. \$ find . -maxdepth 3
4. \$ find . -maxdepth 4
5. \$ find . -maxdepth 100

Note:

1. For maxdepth option we should use singl - but not double --
-maxdepth → Valid
--maxdepth → Invalid
2. find command will find hidden files and directories also.

find by type:

We can find only files or only directories by using type option.

-type f → means only files
-type d → means only directories

```
durgasoft@durgasoft-VirtualBox:~/Desktop$ find -type f
./level_1_dir/file2.txt
./level_1_dir/level_2_dir/file3.txt
./level_1_dir/level_2_dir/level_3_dir/level_4_dir/file5.txt
./level_1_dir/level_2_dir/level_3_dir/file4.txt
./file1.txt
```



`./securefile1.txt`

`durgasoft@durgasoft-VirtualBox:~/Desktop$ find -type d`

```
.  
./db_info  
./level_1_dir  
./level_1_dir/level_2_dir  
./level_1_dir/level_2_dir/level_3_dir  
./level_1_dir/level_2_dir/level_3_dir/level_4_dir
```

Note: We can use these options simultaneously, but we should use first `-maxdepth` and then `-type`.

`$find -type f -maxdepth 2` → Generates warning

`$find -maxdepth 2 -type f` → No warnings

find by name:

We can find files and directories by name by using `-name` option.

`$ touch {A..D}.txt`

`$ touch {A,B}{A,B}.txt`

`$ find . -name 'A.txt'`

`$ find . -name '?.txt'`

`$ find . -name '???.txt'`

`$ find . -name '*.txt'`

`$ find . -maxdepth 2 -name '*.txt'`

If we want to ignore case then we should use `-iname` option.

`$ find -iname 'a.txt'`

find files by size:

We should use `-size` option.

+ symbol means greater than (over)

- symbol means less than

1. To list out all file names where size is over 200kb

`$ find / -type f -size +200k` This command required root privileges

`$ sudo find / -type f -size +200k | wc -l`

2. To list out all file names where size is over 200kb but less than 4MB.

`$ find / -type f -size +200k -size -4M | wc -l`

3. To list out all file names where file size is less 200kb or more than 4MB.

`$ find / -type f -size -200k -o -size +4M | wc -l`

-o means or



Q. What is the output of the following command?

```
$ find / -type f -size -200k -size +4M | wc -l
```

The answer should be: 0

Note:

+n for greater than n

-n for less than n

n for exactly n

-empty File is empty and is either a regular file or a directory

How to use search results of find command?

We can perform any operation (like cp, mv, rm etc) on the results of find command. For this we have to use -exec option.

-exec means execution.

Q. To copy all files present in /etc folder where file size is < 2KB to dir1 directory present in the Desktop?

```
$ find /etc -type f -size -2k -exec cp {} dir1 \;
```

Before performing required copy operation if we want confirmation then we should use -ok option instead of -exec.

```
$ find /etc -type f -size -2k -ok cp {} dir1 \;
```

```
durgasoft@durgasoft-VirtualBox:~/Desktop$ find /etc -type f -size -2k -ok cp {} dir1 \;
```

```
< cp ... /etc/magic.mime > ? y
```

```
< cp ... /etc/fwupd/uefi.conf > ? y
```

```
< cp ... /etc/fwupd/daemon.conf > ? y
```

Magic Assignment:

```
$ mkdir magic
```

```
$ mkdir magic/dir{1..100}
```

```
$ touch magic/dir{1..100}/file{1..100}.txt
```

```
$ mkdir magic;mkdir magic/dir{1..100};touch magic/dir{1..100}/file{1..100}.txt
```

```
$ touch magic/dir$(shuf -i 1-100 -n 1)/sunny.txt
```

```
$ find magic -type f -name 'sunny.txt'
```

```
$ find magic -type f -name 'sunny.txt' -exec mv {} ~/Desktop \;
```

```
$ find magic -type f -name '*.txt' -exec rm {} \;
```



Q. The find command uses a database to search files and directories.

- A) True
- B) False

Ans: B

Q. To find files and directories inside /dev folder and limit its search to only 2 levels of deep?

- A) find -start /dev -depth 2
- B) find /dev -depth 2
- C) find /dev -maxdepth 2

Ans: C

Q. To find only directories inside /dev folder and limit its search to only 2 levels of deep?

- A) find /dev -maxdepth 2 -type f
- B) find /dev -maxdepth 2 -type d
- C) find /dev -type d -maxdepth 2

Ans: B

Q. To find only files starts from root directory (/) where file name ends with .txt?

- A) find / '*.txt'
- B) find / -type d -name '*.txt'
- C) find / -type f -name '*.txt'

Ans: C

Q. To find all files and directories inside /dev directory upto maximum of 3 levels deep and size is greater than 200Kilobytes?

- A) find / -maxdepth 3 -size 200k
- B) find /dev -maxdepth 3 -size 200k
- C) find /dev -maxdepth 3 -size -200k
- D) find /dev -maxdepth 3 -size +200k

Ans: D

Q. Which of the following command will find all files below our home directory where file size is greater than 3 MegaBytes and remove all those files?

- A) find ~ -type f -size +3M -exec rm {} \;
- B) find ~ -type f -size -3M -exec rm {}



- C) `find ~ -type f -size -3M -exec rm {} \;`
D) `find ~ -type f -size +3M rm {} \;`

Ans: A

Difference between find and locate Commands

find	locate
1. It searches directly in the file system for the required files and directories.	1. It searches in the database for the required files and directories.
2. find command has number of options and easy to customize based on our requirement.	2. locate command has very few options.
3. We can find files based on name, type, size, depth, age, user, group, permissions etc	3. We can find files only based on name and permissions.
4. We can reduce the depth of search.	3. We cannot reduce depth of search.
5. find command will produce accurate results as it searches directly in the file system.	5. locate command won't produce accurate results as it searches in the database which will be updated only once per day by default.
6. find command won't produce deleted files in search results as it searches directly in the file system.	6. locate command may produce deleted files in search results as it searches in the database which may not be updated.
7. There is a way to use search results by using -exec option.	7. There is no direct way to use search results.
8. find command operates slowly.	8. locate command operates fastly.



Topic-19: Compression and Uncompression of Files (tar, gzip, gunzip, bzip2, bunzip2)

As the part of admin activity, it is very common requirement to pack and compress a group of files. The main advantages are:

- 1) It improves memory utilization
- 2) Transportation will become very easy
- 3) It reduces download times
- etc

This process involves the following 2 activities:

- 1) Creation of Archive file
- 2) Apply compression algorithms on that archive file

1. Creation of Archive File

We can group multiple files and directories into a single archive file by using tar command.

tar → tape archive

A) To create tar file

```
tar -cvf demo.tar file1.txt file2.txt file3.txt
```

```
tar -cvf demo.tar *
```

B) To display table of contents of tar file

```
tar -tvf demo.tar
```

C) To Extract contents of tar file

```
tar -xvf demo.tar
```

2. Apply Compression Algorithms on that Archive File:

There are multiple compression and decompression algorithms.

1. gzip → It is very fast but less compression power
2. bzip2 → It is a bit slow but more compression power

Compression and Decompression by using gzip:

1. To Compress tar file

```
$ gzip demo.tar
```

demo.tar.gz → This file got created

2. To uncompress gz file:

```
$ gzip -d demo.tar.gz OR $ gunzip demo.tar.gz
```

→ This command will provide our original tar file



Compression and Decompression by using bzip2:

1. To compress tar file:
\$ bzip2 demo.tar
demo.tar.bz2 this file will be created.
2. To uncompress bz2 file:
\$ bunzip2 demo.tar.bz2

How to create tar File and compress in a Single Command:

1. By using gzip compression algorithm

To create tar and then compress

```
$ tar -cvzf demo.tar *.txt
```

z option will do compression
demo.tar will be created and it is already compressed

To uncompress and extract tar file

```
$ tar -xvzf demo.tar
```

2. By using bzip2 compression algorithm

Instead of 'z', we have to use 'j'

To create tar and then compress

```
$ tar -cvjf demo.tar *.txt
```

j option will do compression
demo.tar will be created and it is already compressed

To uncompress and extract tar file

```
$ tar -xvjf demo.tar
```

Use Case: Backup of Total User Home Directory

```
$ pwd  
/home/durgasoft  
$ tar -cvzf backup.tar *
```

```
$ mkdir newhome  
$ mv backup.tar newhome  
$ cd newhome  
$ tar -xvzf backup.tar
```



Topic-20: grep Command

grep stands for
globally search a regular expression and print it
global regular expression print.
global regular expression parser.

We can use grep command to search the given pattern in a single or multiple files.

```
grep <pattern> filename
```

It prints all matched lines.

```
durgasoft@durgasoft-VirtualBox:~/Desktop$ cat subjects.txt
```

Sno	Subjectname	Faculty	Fee
1.	Python	Nagoor	1000
2.	Java	Sriman	2000
3.	Unix	Durga	250
4.	DevOps	Sriman	3500
5.	UNIX	Durga	400
6.	Java	Durga	1000

1) To Search Data in a Single File:

```
$ grep 'durga' subjects.txt  
$ grep "durga" subjects.txt  
$ grep durga subjects.txt
```

it prints all matched lines.

3.	Unix	durga	250
5.	UNIX	durga	400
6.	Java	durga	1000



2) To Search in Multiple Files:

```
$ grep durga subjects.txt career.txt
subjects.txt:3.  unix      durga  250
subjects.txt:5.  UNIX      durga  400
subjects.txt:6.  Java      durga  1000
career.txt:durga
career.txt:durga ksdjfdlakjklfjad
```

```
$ grep durga *.txt
career.txt:durga
career.txt:durga ksdjfdlakjklfjad
subjects.txt:3.  unix      durga  250
subjects.txt:5.  UNIX      durga  400
subjects.txt:6.  Java      durga  1000
```

```
$ grep durga *
career.txt:durga
career.txt:durga ksdjfdlakjklfjad
subjects.txt:3.  unix      durga  250
subjects.txt:5.  UNIX      durga  400
subjects.txt:6.  Java      durga  1000
```

3) To Search Data by ignoring Case:

By default grep command will consider case. If we want to ignore case then we should use -i option.

```
durgasoft@durgasoft-VirtualBox:~/Desktop$ grep unix *.txt
career.txt:unix jksadjfklasjdkflajs
subjects.txt:3.  unix      durga  250
durgasoft@durgasoft-VirtualBox:~/Desktop$ grep -i unix *.txt
career.txt:unix jksadjfklasjdkflajs
subjects.txt:3.  unix      durga  250
subjects.txt:5.  UNIX      durga  400
```

4) To Display the Number of Occurrences:

We have to use -c option.
c means count

```
$ grep -c unix *.txt
career.txt:1
subjects.txt:1
```



5) To Display Line Numbers before Results:

We have to use -n option.

```
durgasoft@durgasoft-VirtualBox:~/Desktop$ grep -n is *.txt
career.txt:1:Java → It is ocean but ever green!!!!
durgasoft@durgasoft-VirtualBox:~/Desktop$ grep -n durga *.txt
career.txt:4:durga
career.txt:7:durga ksdjfdlakjklfjad
subjects.txt:4:3.    unix      durga  250
subjects.txt:6:5.    UNIX      durga  400
subjects.txt:7:6.    Java      durga  1000
```

6) To Display only File Names in which Pattern exists:

We have to use -l option.

```
durgasoft@durgasoft-VirtualBox:~/Desktop$ grep -l durga *.txt
career.txt
subjects.txt
```

7) To Print except matched Lines remaining Lines:

We have to use -v option. It means inverted.

```
durgasoft@durgasoft-VirtualBox:~/Desktop$ grep durga subjects.txt
3.      unix      durga    250
5.      UNIX      durga    400
6.      Java      durga    1000
durgasoft@durgasoft-VirtualBox:~/Desktop$ grep -v durga subjects.txt
sno  subjectname  faculty  fee
1.   Python    Nagoor   1000
2.   java      Sriman   2000
4.   devops    sriman   3500
```

8) To Search for exact word in the File:

For this we have to use -w option.

Eg 1:

```
durgasoft@durgasoft:~/Desktop$ grep -i unix demo.txt
UnixDemo session
unix material
durgasoft@durgasoft:~/Desktop$ grep -iw unix demo.txt
unix material
```




Eg 2:

```
durgasoft@durgasoft:~/Desktop$ grep -in unix demo.txt
```

```
5:UnixDemo session
```

```
6:unix material
```

```
9:UNIX classes and videos
```

```
durgasoft@durgasoft:~/Desktop$ grep -win unix demo.txt
```

```
6:unix material
```

```
9:UNIX classes and videos
```

9) Display before, after and surrounding lines including search results:

We have to use -A,-B,-C options

-A means after

-B means before

-C means before and after

```
durgasoft@durgasoft:~/Desktop$ grep friends demo.txt
```

```
Hello friends how are you
```

```
durgasoft@durgasoft:~/Desktop$ grep -A 2 friends demo.txt
```

```
Hello friends how are you
```

```
DataScienceDemo
```

```
UnixDemo session
```

```
durgasoft@durgasoft:~/Desktop$ grep -B 2 friends demo.txt
```

```
This is java demo
```

```
this is python demo
```

```
Hello friends how are you
```

```
durgasoft@durgasoft:~/Desktop$ grep -C 2 friends demo.txt
```

```
This is java demo
```

```
this is python demo
```

```
Hello friends how are you
```

```
DataScienceDemo
```

```
UnixDemo session
```

```
durgasoft@durgasoft:~/Desktop$ grep -2 friends demo.txt
```

```
This is java demo
```

```
this is python demo
```

```
Hello friends how are you
```

```
DataScienceDemo
```

```
UnixDemo session
```

Note: C is optional.



Search Multiple Content in a File:

```
durgasoft@durgasoft:~/Desktop$ grep -i -e "java" -e "unix" demo.txt
```

This is java demo

UnixDemo session

unix material

UNIX classes and videos

java is slowly going down

Instead of using -e option, we can use egrep command directly.

It is extended grep. It interprets patterns as an extended regular expression.

```
durgasoft@durgasoft:~/Desktop$ egrep -i "(java|unix)" demo.txt
```

This is java demo

UnixDemo session

unix material

UNIX classes and videos

java is slowly going down

grep with -F Option OR fgrep:

fgrep → "Fixed String Global Regular Expression Print"

It will take a group of fixed strings and search for those in the given file. Strings should be separated by new line.

```
durgasoft@durgasoft:~/Desktop$ grep -F "java
```

```
> unix" demo.txt
```

This is java demo

unix material

java is slowly going down

Instead of using , -F option, we can use directly fgrep.

```
durgasoft@durgasoft:~/Desktop$ fgrep "java
```

```
> unix
```

```
> friends
```

```
> demo" demo.txt
```

This is java demo

this is python demo

Hello friends how are you

unix material

java is slowly going down

Note: fgrep can be used only for Strings and cannot be used for regular expressions.

Normal grep command cannot understand some regular expression patterns like | symbol. But egrep command can understand any regular expression pattern. Hence egrep is the more powerful than normal grep command.



Eg:

```
durgasoft@durgasoft:~/Desktop$ cat demo.txt
```

This is java demo

UnixDemo session

unix material

UNIX classes and videos

java is slowly going down

(java|unix) this is extra line added

```
durgasoft@durgasoft:~/Desktop$ grep -i "(java|unix)" demo.txt
```

(java|unix) this is extra line added

In this case (java|unix) is considered as a string and cannot be processed as regular expression.

```
durgasoft@durgasoft:~/Desktop$ egrep -i "(java|unix)" demo.txt
```

This is java demo

UnixDemo session

unix material

UNIX classes and videos

java is slowly going down

(java|unix) this is extra line added

In this case, (java|unix) is treated as regular expression. It will search for either java or unix.

egrep and fgrep are deprecated and hence it is recommend to use grep -E and grep -F commands.

If strings are available in the file, then we can use -f option to specify file name.

```
durgasoft@durgasoft:~/Desktop$ fgrep -f file1.txt demo.txt
```

This is java demo

this is python demo

Hello friends how are you

unix material

java is slowly going down

This is java demo

this is python demo

Hello friends how are you

DataScienceDemo

UnixDemo session

unix material

dsfjalkjfdlsda

sadkfjklsadjfkld

UNIX classes and videos



java is slowly going down
demounix
unixforall
grtgrgtujmik, fgtunix dghjhrunix fghubunixdfgghj

file1.txt

java
unix
python
demo
-i,-n,-c,-v,-l,-A,-B,-C,-e,-E, -F,-f, -o, -R

To Print only matched Patterns instead of Total Line:

We have to use -o option.

```
durgasoft@durgasoft:~/Desktop$ grep c[aeiou]ll demo.txt
call center
cell point
college for unix
durgasoft@durgasoft:~/Desktop$ grep -o c[aeiou]ll demo.txt
call
cell
coll
```

Q. Write grep Command to extract only Mobile Numbers present in the Input File and write to mobile.txt?

```
$ grep -o [6-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9] input.txt > mobile.txt
$ grep -o "[6-9][0-9]{9}" input.txt > mobile.txt → This command won't work because {9}
cannot be understandable by grep command. But egrep can understand.
```

```
$ egrep -o "[6-9][0-9]{9}" input.txt > mobile.txt
```

```
durgasoft@durgasoft:~/Desktop$ cat demo.txt
call center
cell point
college for unix
extra line 1
extra line2
c[aeiou]ll point for fgrep
durgasoft@durgasoft:~/Desktop$ grep c[aeiou]ll demo.txt
call center
cell point
college for unix
durgasoft@durgasoft:~/Desktop$ fgrep c[aeiou]ll demo.txt
```



```
c[aeiou]ll point for fgrep
durgasoft@durgasoft:~/Desktop$ egrep c[aeiou]ll demo.txt
call center
cell point
college for unix
```

To Search in the Files recursively inside a Directory:

If we want to use grep command for all files present in the specified directory and sub directory recursively then we should use R option.

```
resumes
  java
    students.txt
  python
    students.txt
  testing
    students.txt
$ mkdir resumes
```

```
durgasoft@durgasoft:~/Desktop$ egrep "[6-9][0-9]{9}" resumes/
grep: resumes/: Is a directory
durgasoft@durgasoft:~/Desktop$ egrep -R "[6-9][0-9]{9}" resumes/
resumes/python/students.txt:Durga 9898989898
resumes/python/students.txt:Ravi 8096969696
resumes/python/students.txt:Shiva 9797975656
resumes/java/students.txt:Durga 9898989898
resumes/java/students.txt:Ravi 8096969696
resumes/java/students.txt:Shiva 9797975656
resumes/testing/students.txt:Durga 9898989898
resumes/testing/students.txt:Ravi 8096969696
resumes/testing/students.txt:Shiva 9797975656
```

Note: If we want to use grep command for directory, compulsory we should use -R option, which means recursive.

Assignment:

1. Save all running processes information inside a file.
\$ ps -ef > results.txt
2. Display all lines which contains 'lib'
\$ grep 'lib' results.txt
3. Display all lines which does not contain 'lib'
\$ grep -v 'lib' results.txt



4. Count all lines where we are getting 'lib'
`$ grep -c 'lib' results.txt`
5. Display all lines which contain 'lib' and preceding with line numbers?
`$ grep -n 'lib' results.txt`
6. Display lines which do not contain 'lib', but only top 2 lines.
`$ grep -v 'lib' results.txt | head -2`

Assignment-2:

```
durgasoft@durgasoft:~/Desktop$ cat emp.dat
eno|ename|esal|eaddr|dept|gender
100|sunny|1000|mumbai|admin|female
200|bunny|2000|chennai|sales|male
300|chinny|3000|delhi|accounting|female
400|vinny|4000|hyderabad|admin|male
500|pinny|5000|mumbai|sales|female
```

1. Display all employees belongs to admin department
`$ grep 'admin' emp.dat`
2. Display all employees in admin and sales department
`$ egrep '(admin|sales)' emp.dat`
3. Display all male employees in admin and sales department
`$ egrep '(admin|sales)' emp.dat | grep -w 'male'`

Types of Regular Expressions/Patterns:

All Regular expression patterns are divided into 3 types.

- 1) Character Patterns
- 2) Word Patterns
- 3) Line Patterns

1) Character Patterns:

- 1) `$ grep 'd*' demo.txt`
 - It display all lines which contains d followed by any number of characters.
 - ubuntu not providing support for this.
- 2) `$ grep 'c[aeiou]ll' demo.txt`
 - It will search for call, cell, cill, coll, cull
- 3) `$ grep 'b..l' demo.txt`
 - . means any character. It will search for all 4 letter words where first letter should be b and last letter should be l.



2) Word Patterns:

`\<word\>` → It will always searches for the given word
It is exactly same as
`grep -w word demo.txt`

`\<xyz` → It will search for the word starts with xyz
`xyz\>` → It will search for the word ends with xyz

```
durgasoft@durgasoft:~/Desktop$ cat > demo.txt
durga
durgasoft
techdurga
softwaredurgasolutions
```

```
durgasoft@durgasoft:~/Desktop$ grep 'durga' demo.txt
durga
durgasoft
techdurga
softwaredurgasolutions
```

```
durgasoft@durgasoft:~/Desktop$ grep '\<durga\>' demo.txt
durga
```

```
durgasoft@durgasoft:~/Desktop$ grep '\<durga' demo.txt
durga
durgasoft
```

```
durgasoft@durgasoft:~/Desktop$ grep 'durga\>' demo.txt
durga
techdurga
```

```
durgasoft@durgasoft:~/Desktop$ cat demo.txt
```

```
1234
123456
6789
67543
12
1
```

```
durgasoft@durgasoft:~/Desktop$ grep '\<[0-9][0-9][0-9][0-9]\>' demo.txt
1234
6789
```

It will always search for 4 digit words.



3) Line Patterns (Anchors):

^ → Line starts with

\$ → Line ends with

1) \$ grep '^d' demo.txt

It will display all lines starts with d

2) \$ grep '^the' demo.txt

It will display all lines starts with the

3) \$ grep '^<the\>' demo.txt

It will display all lines starts with the word 'the'

4) \$ grep '^[aeiou]' demo.txt

It will display all lines starts with vowel.

5) \$ grep '^[^aeiou]' demo.txt

It will display all lines not starts with vowel.

6) \$ grep 't\$' demo.txt

It will display all lines ends with t

7) \$ grep '[aeiou]\$' demo.txt

It will display all lines ends with vowel

8) \$ grep '[0-9]\$' demo.txt

It will display all lines ends with digit.

9) \$ grep '^unix\$' demo.txt

It will display all lines where total line content should be unix

```
durgasoft@durgasoft:~/Desktop$ cat > demo.txt
```

```
unix
```

```
unix software
```

```
unix material
```

```
material unix
```

```
unix unix
```

```
durgasoft@durgasoft:~/Desktop$ grep '^unix$' demo.txt
```

```
unix
```

10) \$ grep '^....\$' demo.txt

It will display all lines where line contains exactly 4 characters.



11) `$ grep '^\. ' demo.txt`

It will display all lines starts with .

12) `$ grep '\$ $' demo.txt`

It will display all lines ends with \$

13) `$ grep '^$' demo.txt`

It will display all blank lines

14) `$ grep -v '^$' demo.txt`

It will display all lines except blank lines

Q. How to delete blank lines present in the given file?

```
$ grep -v '^$' demo.txt > temp.txt
```

```
$ mv temp.txt demo.txt
```

Note: `$ grep -v '^$' demo.txt > demo.txt`

The above command won't work

grep: input file 'demo.txt' is also the output

Additional patterns supported by only egrep but not grep:

1. `()` It matches any of the string in the given list

```
$ egrep '(unix|java|oracle)' demo.txt
```

2. `{m}` → It matches exact number of preceding character.

```
$ egrep '[6-9][0-9]{9}' demo.txt
```

It will search for 10 digit mobile numbers

3. `{m,n}` → The preceding character should match minimum m times and maximum n times.

```
$ egrep '[0-9]{1,5}' demo.txt
```

It will search for lines which contains minimum 1 digit and maximum 5 digit numbers.

4. `{m,}` → minimum m number of times but no restriction on maximum number

```
$ egrep '[0-9]{3,}' demo.txt
```

minimum 3 digits but no restriction on maximum number.



Q1. Write grep Command to retrieve Date Values Present in the given Input File where Date Values are in the Form dd-mm-yyyy OR dd/mm/yyyy ?

input.txt

Durga 28-05-1947

Ravi 23/07/1957

shiva 23-09-2015

kljdsjgklfjd

dsklfjakdlsfjkl

sdlkfskfjdklsjfdkl

```
$ egrep -o '\<[0123][0-9][-/][01][0-9][-/][0-9]{4}\>' input.txt
```

Q2. Write Commands to Count the Number of Directories Present in the Current Working Directory?

```
$ ls -l | grep '^d' | wc -l
```

```
$ ls -F | grep '/$' | wc -l
```

Q3. Write Commands to Count the Number of Files Present in the Current Working Directory?

```
$ ls -l | grep '^-' | wc -l
```

Q4. Write Commands to Count the Number of Link Files Present in the /etc Directory?

```
$ ls -F /etc | grep '@$' | wc -l
```

Q5. Write Command to Display all executable Files Present in /bin Directory?

```
$ ls -F /bin | grep '*$'
```

```
$ ls -F /bin | grep '*$' | wc -l
```



Topic-21: Cut, Paste and tr Commands

We can use cut command to extract data from the file.
The file should contain column formatted data, ie tabular data.

emp.dat

```
eno|ename|esal|eaddr|dept|gender
100|sunny|1000|mumbai|admin|female
200|bunny|2000|chennai|sales|male
300|chinny|3000|delhi|accounting|female
400|vinny|4000|hyderabad|admin|male
500|pinny|5000|mumbai|sales|female
```

1) Display Character on specific Position in every Record:

```
$ cut -c 9 emp.dat
    -c meant for specific character
```

```
e
y
y
n
y
y
```

2) Display Range of Characters in every Record:

- \$cut -c 5-9 emp.dat
It will display 5th to 9th characters in every record
- \$cut -c 5- emp.dat
It will display 5th character to last character in every record
- \$ cut -c -3 emp.dat
It will display from 1st character to 3rd character in every record.
- \$ cut -c 3-5,7-10 emp.dat
It will display 3rd to 5th character and 7th to 10th character in every record.



3) Display Specific Column Data:

- `$ cut -d '|' -f 3 emp.dat`
-d means delimiter (separator). The default delimiter is tab.
-f means field
- It will display 3rd Field (OR Column) data.

4) Display Range of Columns:

- `$ cut -d '|' -f 2-3 emp.dat`
It will display 2nd and 3rd Columns data.
- `$ cut -d '|' -f 2- emp.dat`
It will display from 2nd column to last column data.
- `$ cut -d '|' -f -3 emp.dat`
It will display from 1st column to 3rd column data.
- `$ cut -d '|' -f 1,3,5 emp.dat`
It will display 1st, 3rd and 5th column data.

5) Skip specific Column:

- Display all columns except specific column
- `$ cut -d '|' --complement -f3 emp.dat`
It will display all columns data except 3rd column.
- `$ cut -d '|' --complement -f3,5 emp.dat`
- `$ cut -d '|' --complement -f3-5 emp.dat`
- `$ cut -d '|' --complement -f3- emp.dat`
- `$ cut -d '|' --complement -f-5 emp.dat`

paste Command:

We can use paste command to join two or more files horizontally by using some delimiter.
Default delimiter is tab.

Syntax: `$ paste file1 file2 ...`

Eg:

`durgasoft@durgasoft:~/Desktop$ cat > subjects.txt`

Core Java

Adv Java

Python

Django

UNIX



```
durgasoft@durgasoft:~/Desktop$ cat > fee.txt
1000
2000
2500
1500
250
durgasoft@durgasoft:~/Desktop$ paste subjects.txt fee.txt
Core Java      1000
Adv Java       2000
Python         2500
Django         1500
UNIX           250
```

We can specify delimiter explicitly by using -d option.

```
$ paste -d '-' subjects.txt fee.txt
Core Java-1000
Adv Java-2000
Python -2500
Django-1500
UNIX-250
```

Note: Delimiter should be only one character. If we are providing more than one character, then it will consider only first character.

tr Command:

tr means translate.

This command translates character by character.

```
durgasoft@durgasoft:~/Desktop$ cat > demo.txt
While learning unix not required to eat.
```

1) \$ tr 'aeiou' 'AEIOU' < demo.txt

- It will replace lower case vowels with upper case vowels in demo.txt
- WHILE LEARNING UNIX NOT REQUIRED TO EAT.

2) \$ tr '[a-z]' '[A-Z]' < demo.txt

- Every lower case alphabet symbol will be replaced with upper case alphabet symbol.
- WHILE LEARNING UNIX NOT REQUIRED TO EAT.

3) \$ tr '[a-zA-Z]' '[A-Za-z]' < demo.txt

- Every lower case character will be replaced with upper case character and every upper case character will be replaced with lower case character.
- tr '[a-zA-Z]' '[A-Za-z]' < demo.txt > temp.txt



4) `$ tr 'aeiou' '7' < demo.txt`

To replace every lower case vowel with digit 7.

5) `$ tr '|' '\t' < emp.dat`

- Replace | symbol with tab in emp.dat

• `$ tr '|' ':' < emp.dat`

6) `$ tr -d 'a' < demo.txt`

- -d means delete
- It will delete all occurrence of 'a' in demo.txt
- `$ tr -d 'aeiou' < demo.txt`
- It will delete all lower case vowels in demo.txt

7) `$ tr -s 'a' < demo.txt`

- It replaces sequence of a's with a single a.
- -s means squeeze-repeats.



Topic-22: File Permissions

File Permissions describe the allowed operations by various users.

With respect to file permissions, all users are categorized into the following 4 types.

User Categories:

user/owner → Represented by 'u'

group → Represented by 'g'

others → Represented by 'o'

all → Represented by 'a'

Use Case to understand Types of Users:

Project: DURGASOFT COLLEGE AUTOMATION SYSTEM

This project divided into multiple modules. In each module multiple developers are working.

- 1) STUDENTS MODULE
A, B, C, D ARE WORKING
- 2) EMPLOYEES MODULE
X, Y, Z ARE WORKING
- 3) COURSES MODULE
M, N ARE WORKING
- 4) INFRASTRUCTURE MODULE
G, H ARE WORKING



DEVELOPER 'A' CREATED ONE FILE demo.txt

For demo.txt

user/owner: A (The person who created the file)

group: B,C,D (The persons who are working in the same module)

others: X,Y,Z,M,N,G,H (The persons who are working on other modules)

Permission Types:

For files and directories, there are 4 types of permissions.

- 1) r → Read
- 2) w → Write
- 3) x → Execute
- 4) - → No Permission

Table_FilePermissions

Permission Type	For Files	For Directories
1. Read (r)	Read permission for the file means we can view content of the file.	Read permission for the directory means we can list out contents of that directory. i.e we can use ls command.
2. Write(w)	Write permission for the file means we can modify the content of the file.	Write permission for the directory means we can modify contents of that directory. i.e we can create a new file and we can delete an existing file etc
3. Execute	Execute permission for the file means we can execute the file just like a normal program.	Execute permission to the directory means we can enter into that directory. i.e we can use cd command.

Operations related to permissions:

We can perform the following 3 operations.

- + → Add a particular permission to user|group|other|all
- → Remove a particular permission to user|group|other|all
- = → Assignment a particular permission to user|group|other|all

chmod Command:

chmod means change mode.

We can use chmod command to change file or directory permissions.

Syntax: \$ chmod <user_category><operation><permission> file_name/directory_name

Eg: For user add execute permission, for group add write permission, for others remove read permission

\$ chmod u+x,g+w,o-r demo.txt



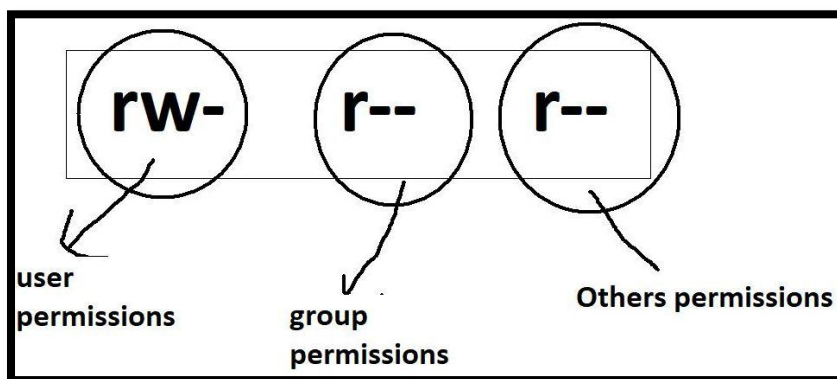
Note: Only owner and super user (root) can change file permissions.

How to check Permissions of existing File:

By using ls -l command:

```
durgasoft@durgasoft:~/Desktop$ ls -l
total 0
-rw-r--r-- 1 durgasoft durgasoft 0 Nov 27 21:19 demo.txt
```

The file permissions are



Total 9 permissions. First 3 are user permissions, next 3 are group permissions and next 3 are others permissions.

user permissions: rw-

user can perform both read and write operations but not execute operation

group permissions: r--

group members can perform only read operation and cannot perform write and execute operations

others permissions: r--

other members can perform only read operation and cannot perform write and execute operations.

User Permissions + Group Permissions + Others Permissions
order is important

Read permission + Write Permission + Execute Permission
order is important

Eg 1: \$ chmod u+x demo.txt
adding execute permission to the user



Eg 2: \$ chmod u+w,g+rw,o+r demo.txt
adding write permission to the user
adding read and write permissions to the group
adding read permission to the others

Eg 3: \$ chmod u+x,g-w,o+w demo.txt
adding execute permission to the user
removing write permission from the group
adding write permission to the others

Eg 4: \$ chmod u=rw,g=rw,o=r demo.txt
Now user permissions: rw-
group permission: rw-
others permission: r--

Eg 5: \$ chmod a=- demo.txt
Now user permissions: ---
group permission: ---
others permission: ---

Eg 6: \$ chmod a=rwx demo.txt
Now user permissions: rwx
group permission: rwx
others permission: rwx

Read Permission to the File:

If the file not having read permission then we are not allowed to view content of the file.
Hence cat, head, tail, more, less commands won't work.

```
durgasoft@durgasoft:~/Desktop$ ls -l
total 4
-rw-r--r-- 1 durgasoft durgasoft 34 Nov 28 20:56 demo.txt
durgasoft@durgasoft:~/Desktop$ cat demo.txt
This is file content at beginning
durgasoft@durgasoft:~/Desktop$ chmod u-r demo.txt

--w-r--r-- 1 durgasoft durgasoft 34 Nov 28 20:56 demo.txt
durgasoft@durgasoft:~/Desktop$ cat demo.txt
cat: demo.txt: Permission denied
```



Write Permission to the File:

If the file not having write permission, then we cannot modify the content of the file.

```
durgasoft@durgasoft:~/Desktop$ chmod u-w demo.txt
durgasoft@durgasoft:~/Desktop$ ls -l
total 4
-r--r--r-- 1 durgasoft durgasoft 66 Nov 28 21:04 demo.txt
durgasoft@durgasoft:~/Desktop$
```

```
durgasoft@durgasoft:~/Desktop$ cat >> demo.txt
bash: demo.txt: Permission denied
durgasoft@durgasoft:~/Desktop$ chmod u+w demo.txt
durgasoft@durgasoft:~/Desktop$ cat >> demo.txt
```

Execute Permission to the File:

If the user not has executed permission on any file, then he cannot execute that file as a program.

```
durgasoft@durgasoft:~/Desktop$ cat > first.sh
echo "Hello Friends..."
echo "File Permissions is very very important than Sunny"
```

```
durgasoft@durgasoft:~/Desktop$ ls -l first.sh
-rw-r--r-- 1 durgasoft durgasoft 82 Nov 28 21:10 first.sh
durgasoft@durgasoft:~/Desktop$ ./first.sh
bash: ./first.sh: Permission denied
durgasoft@durgasoft:~/Desktop$ chmod u+x first.sh
durgasoft@durgasoft:~/Desktop$ ls -l first.sh
-rwxr--r-- 1 durgasoft durgasoft 82 Nov 28 21:10 first.sh
durgasoft@durgasoft:~/Desktop$ ./first.sh
Hello Friends...
File Permissions is very very important than Sunny
```

Read Permission to the Directory:

If the user has read permission on any directory, then he can list out the contents of that directory. i.e he can use ls command.

```
durgasoft@durgasoft:~/Desktop$ ls -l
total 4
drwxr-xr-x 2 durgasoft durgasoft 4096 Nov 28 21:18 dir1
durgasoft@durgasoft:~/Desktop$ ls dir1
demo.txt first.sh
durgasoft@durgasoft:~/Desktop$ chmod u-r dir1
durgasoft@durgasoft:~/Desktop$ ls -l
```



```
total 4
d-wxr-xr-x 2 durgasoft durgasoft 4096 Nov 28 21:18 dir1
durgasoft@durgasoft:~/Desktop$ ls dir1
ls: cannot open directory 'dir1': Permission denied
durgasoft@durgasoft:~/Desktop$
```

Write Permission on the Directory:

If the user has write permission on any directory, then he is allowed to modify the content of that directory. i.e he can add new files and remove existing files.

```
durgasoft@durgasoft:~/Desktop$ ls -l
total 4
dr-xr-xr-x 2 durgasoft durgasoft 4096 Nov 28 21:18 dir1
durgasoft@durgasoft:~/Desktop$ touch dir1/newfile.txt
touch: cannot touch 'dir1/newfile.txt': Permission denied
durgasoft@durgasoft:~/Desktop$ rm -f dir1/demo.txt
rm: cannot remove 'dir1/demo.txt': Permission denied
durgasoft@durgasoft:~/Desktop$ chmod u+w dir1
durgasoft@durgasoft:~/Desktop$ ls -l
total 4
drwxr-xr-x 2 durgasoft durgasoft 4096 Nov 28 21:18 dir1
durgasoft@durgasoft:~/Desktop$ touch dir1/newfile.txt
durgasoft@durgasoft:~/Desktop$ rm -f dir1/demo.txt
durgasoft@durgasoft:~/Desktop$ ls dir1
first.sh newfile.txt
```

Execute Permission to the Directory:

If the user not has executed permission on any directory, then he is not allowed to enter into that directory. i.e he cannot use cd command.

```
durgasoft@durgasoft:~/Desktop$ ls -l
drwxr-xr-x 2 durgasoft durgasoft 4096 Nov 28 21:26 dir1
durgasoft@durgasoft:~/Desktop$ chmod u-x dir1
durgasoft@durgasoft:~/Desktop$ ls -l
total 4
drw-r-xr-x 2 durgasoft durgasoft 4096 Nov 28 21:26 dir1
\durgasoft@durgasoft:~/Desktop$ cd dir1
bash: cd: dir1: Permission denied
durgasoft@durgasoft:~/Desktop$ chmod u+x dir1
durgasoft@durgasoft:~/Desktop$ cd dir1
```

***Note: If the user not having execute permission on any directory, then he cannot perform read and write operations also, because to perform these operations he should enter into that directory which is not possible.



Note: If the user not having read permission on any file, then he cannot execute that file even though he has executed permission.

```
durgasoft@durgasoft:~/Desktop$ chmod u-x dir1
durgasoft@durgasoft:~/Desktop$ ls -l
total 4
drw-r-xr-x 2 durgasoft durgasoft 4096 Nov 28 21:26 dir1
durgasoft@durgasoft:~/Desktop$ ls dir1
ls: cannot access 'dir1/first.sh': Permission denied
ls: cannot access 'dir1/newfile.txt': Permission denied
first.sh newfile.txt
durgasoft@durgasoft:~/Desktop$ touch dir1/demo.txt
touch: cannot touch 'dir1/demo.txt': Permission denied
durgasoft@durgasoft:~/Desktop$ rm dir1/first.sh
rm: cannot remove 'dir1/first.sh': Permission denied
```

Linux vs Security:

The virus files usually created by others.

others are not having execute permission on our directories. Hence others are not allowed to add virus files to our directories.

Hackers are not having executed permission on our directories. Hence they cannot read our file data.

Because of this, Linux is considered as more secured operating system.

Linux follows 2 levels of security.

1st level: login with credentials

2nd level: File and Directory permissions

Note: We are using permission types as r,w,x and these are considered as symbolic permissions. But we can also specify permissions by using octal number, such type of permissions are called numeric permissions.

Numeric Permissions:

We can specify permissions by using octal number.

Octal means base-8 and allowed digits are 0 to 7.

0 → 000 → No Permission

1 → 001 → Execute Permission

2 → 010 → Write Permission

3 → 011 → Write and execute Permissions

4 → 100 → Read Permission

5 → 101 → Read and execute Permissions

6 → 110 → Read and write Permission

7 → 111 → Read, Write and execute Permissions



Note:

- 4 → Read Permission
- 2 → Write Permission
- 1 → Execute Permission

It is more easy to remember

- 5 → 4+1 → r-x
- 3 → 2+1 → -wx
- 6 → 4+2 → rw-
- etc

1. Write command for the following permissions

- For user → Read and write (6)
- For group → Write and execute (3)
- For others → Write (2)

```
$ chmod 632 demo.txt
durgasoft@durgasoft:~/Desktop$ ls -l demo.txt
-rw--wx-w- 1 durgasoft durgasoft 0 Nov 29 20:57 demo.txt
```

```
$ chmod 135 demo.txt
---x-wxr-x 1 durgasoft durgasoft 0 Nov 29 20:57 demo.txt
```

```
$ chmod 777 demo.txt
-rwxrwxrwx 1 durgasoft durgasoft 0 Nov 29 20:57 demo.txt
```

```
$ chmod 77 demo.txt
durgasoft@durgasoft:~/Desktop$ ls -l demo.txt
----rwxrwx 1 durgasoft durgasoft 0 Nov 29 20:57 demo.txt
```

77 means 077

```
$ chmod 7 demo.txt
durgasoft@durgasoft:~/Desktop$ ls -l demo.txt
-----rwx 1 durgasoft durgasoft 0 Nov 29 20:57 demo.txt
7 means 007
```

```
$ chmod demo.txt
durgasoft@durgasoft:~/Desktop$ chmod demo.txt
chmod: missing operand after 'demo.txt'
Try 'chmod --help' for more information.
```

```
$ chmod 0 demo.txt
durgasoft@durgasoft:~/Desktop$ ls -l demo.txt
----- 1 durgasoft durgasoft 0 Nov 29 20:57 demo.txt
```



umask Command:

umask means user mask. Hiding permissions.

Based on umask value, default permissions will be there for files and directories.

The default umask value: 022

```
durgasoft@durgasoft:~/Desktop$ umask  
0022
```

First 0 is sticky bit mostly used in admin related activities. We have to consider only last 3 digits as umask value.

Default permissions to the file: 666 - umask value

= 666 - 022

= 644 (user → r&w, group → read, others → read)

```
durgasoft@durgasoft:~/Desktop$ ls -l file1.txt  
-rw-r--r-- 1 durgasoft durgasoft 0 Nov 29 21:18 file1.txt
```

Default permissions to the directory = 777 - umask value

= 777 - 022

= 755 (user → r&w&x, group → r&x, others → r&x)

```
$ mkdir dir1
```

```
$ ls -l
```

```
drwxr-xr-x 2 durgasoft durgasoft 4096 Nov 29 21:21 dir1
```

Based on our requirement we can set our own umask values.

```
durgasoft@durgasoft:~/Desktop$ umask 002
```

```
durgasoft@durgasoft:~/Desktop$ touch file2.txt
```

```
durgasoft@durgasoft:~/Desktop$ ls -l file2.txt
```

```
-rw-rw-r-- 1 durgasoft durgasoft 0 Nov 29 21:24 file2.txt
```

```
durgasoft@durgasoft:~/Desktop$ mkdir dir2
```

```
durgasoft@durgasoft:~/Desktop$ ls -l
```

```
drwxrwxr-x 2 durgasoft durgasoft 4096 Nov 29 21:24 dir2
```

Q. For newly created files default permissions should be 444. Then what should be umask value?

Ans: 222

Everyone has only read access.

Note: The most commonly used umask values are: 022

002

077

007



Eg:

```
durgasoft@durgasoft:~/Desktop$ umask 077
durgasoft@durgasoft:~/Desktop$ umask
0077
durgasoft@durgasoft:~/Desktop$ touch file3.txt
durgasoft@durgasoft:~/Desktop$ ls -l file3.txt
-rw----- 1 durgasoft durgasoft 0 Nov 29 21:30 file3.txt
Negative values won't be considered in linux.
```

chown Command:

chown means change owner.
Only root user can perform this activity.

```
# chown root demo.txt
Now the owner of demo.txt is root.
```

chgrp Command:

chgrp means change group.
Only root user can perform this activity.

```
# chgrp root demo.txt
Now the demo.txt belongs to root group.
```

```
durgasoft@durgasoft:~$ sudo -i
[sudo] password for durgasoft:
root@durgasoft:~#
root@durgasoft:~# whoami
root
root@durgasoft:~# cd /home/durgasoft/Desktop
root@durgasoft:/home/durgasoft/Desktop# ls
demo.txt
root@durgasoft:/home/durgasoft/Desktop# ls -l
total 0
-rw-r--r-- 1 durgasoft durgasoft 0 Nov 29 21:33 demo.txt
root@durgasoft:/home/durgasoft/Desktop# chown root demo.txt
root@durgasoft:/home/durgasoft/Desktop# ls -l
total 0
-rw-r--r-- 1 root durgasoft 0 Nov 29 21:33 demo.txt
root@durgasoft:/home/durgasoft/Desktop#
```




Case Study:

```
-rwxrw-r-- 1 root durgasoft 42 Nov 30 19:47 demo.txt
owner: root
group: durgasoft
```

Group members can perform both read and write operations.

Eg 2:

```
-rwxrwxr-- 1 root root 90 Nov 30 19:55 demo.txt
owner: root
group: root
others can perform only read operation.
durgasoft user can perform only read operation
durgasoft@durgasoft:~/Desktop$ cat demo.txt
echo "Hello Friends"
echo "Hello Friends"
echo "Durgasoft group member can perform write"
durgasoft@durgasoft:~/Desktop$ cat >> demo.txt
bash: demo.txt: Permission denied
durgasoft@durgasoft:~/Desktop$ ./demo.txt
bash: ./demo.txt: Permission denied
```

Group Permissions:

```
durgasoft durgasoft
```

```
-rwxr--r-- 1 root durgasoft 46 Nov 30 20:11 demo.txt
```

```
r-- group members can perform only read operation
durgasoft@durgasoft:~/Desktop$ cat demo.txt
echo "easy but listen"
echo "easy but listen"
durgasoft@durgasoft:~/Desktop$ cat >> demo.txt
bash: demo.txt: Permission denied
durgasoft@durgasoft:~/Desktop$ ./demo.txt
bash: ./demo.txt: Permission denied
durgasoft@durgasoft:~/Desktop$
```

```
-rwxrw-r-- 1 root durgasoft 46 Nov 30 20:11 demo.txt
rw-
```

Others Permissions:

```
-rwxrwxr-- 1 root root 114 Nov 30 20:22 demo.txt
```

```
durgasoft will become others
r-- only read permission
```



Topic-23: Working with Editors

We can use editors to edit file content.
There are multiple editors are available.

- 1) gedit
- 2) vi editor
- 3) nano editor
- etc

1) working with gedit:

It is graphical editor. It is simply same as window's notepad.

```
$ gedit file1.txt  
$ gedit first.sh
```

```
echo "This is my first shell script"  
mkdir dir{1..6}  
echo "six directories got created"  
date  
cal
```

```
$ chmod u+x first.sh  
$ ./first.sh
```

Note: gedit can work only in the desktop version and cannot work in server version.
By using putty if we are connecting remote server, then we cannot use gedit and compulsory we should go for vi editor.

vi editor is unix based where as gedit and nano are linux based.
vi can work anywhere.



2) Working with vi Editor:

vi means visual editor.

We can use vi to create new files and to edit content of the existing files.

```
$ vi file1.txt
```

If file1.txt is not available, then a new file will be created and opened that file for editing purpose.

To save this empty file we should use :wq (w means save and q means exit)

If the file already contains some data then editor will be opened that file and ready for edit.

How to edit the File:

There are 3 types of modes in file editing

1) Command Mode:

- It is the default mode.
- Here we can use any vi command.
- From command mode, we can enter into insert mode by using multiple ways, but mostly by using i.

2) Insert Mode OR Input Mode:

- In this mode, we can modify file data. We can insert/append new data.
- From insert mode, we can enter into command mode by using <Esc> key.

3) Exit Mode:

- To quit from the editor.
- From the command mode we have to press: then we can enter into exit mode.

How to Insert and Append Data:

A → To Append data at the end of the line

I → To Insert data at the beginning of the line

a → To append data to the right side of the cursor position (Just after the cursor position)

i → To insert data to the left side of the cursor position (Just before the cursor position)



How to Delete Data:

We can delete data either by character wise or by word wise or by line wise.

deletion character wise:

x → To delete a single character (del key)

3x → To delete 3 characters. Instead of 3, we can pass any number.

X → To delete previous character (backspace key)

3X → To delete last 3 previous characters

Deletion Words wise:

dw → To delete current word.

3dw → To delete 3 words

Deletion Line wise:

dd → To delete current line

3dd → To delete 3 lines

d\$ → Deletes from current position to end of line.

d^ → Deletes from current position to beginning of the line.

dgg → Deletes from beginning of the file to current cursor position.

Dg → Deletes from current position to end of file.

How to Replace Data:

r → Replace current character.

R → To replace multiple characters from the current position.

S OR cc → To replace a single line

Opening New Lines to Insert Data:

O → To open a line above the cursor position.(i.e before current line)

o → To open a line below the cursor position (i.e after current line)

Copy and Paste Data:

yy → To Copy a Line (yanking)

3yy → To copy 3 lines

yw → To copy a word

3yw → To copy 3 words

y\$ → To copy from current cursor position to end of line.

y^ → To copy from beginning of the line to current cursor position.

p → Paste above the cursor position

P → Paste below the cursor position



Cursor Navigation Commands:

k → Top Arrow
j → Bottom Arrow
l → Right Arrow
h → Left Arrow
3k → 3 Times Top Arrow
3j → 3 Times Bottom Arrow
3l → 3 Times Right Arrow
3h → 3 Times Left Arrow
\$ → End of the Current Line (End Key)
^ → Beginning of the Current Line (Home Key)
H → Beginning of the Current Page
M → Middle of the Current Page
L → End of the Current Page

We are not required to use these because keyboard arrow keys already working.

b(nb) → Back to beginning of the Word
3b → Back to beginning of the 3rd (ie 3rd previous Word)
e(ne) → End of the Current Word
w(nw) → Forward to beginning of next Word.

G → Move to last line (ie end of the file)

at exit mode

:1 → To go to 1st line
:7 → To go to 7th line

ctrl+f → One Page Forward (Page Down)
ctrl+b → One Page Backward (Page Up)

Note: If we want to perform undo previous operation then we should use 'u'
u → Means undo previous operation

Exit Mode Commands:

:w → Save File Data
:wq → Save and Quit from the Editor
:q → Quit Editor
:q! → Force Quit. If we perform any changes those will be discarded.

:set nu → To set line numbers in the editor
:set nonu → To remove line numbers
:n → Place the cursor to the nth line
:\$ → Place the cursor to the last line
:!<unix_command> → To execute any command



Eg:

:!date → To check system date

:!cal → To check calander

Note: vi +7 demo.txt

open demo.txt and enter into 7th line

3) Working with nano Editor:

- It is command line editor.
- It can be used to create new files and edit content of existing files.
- It is almost like notepad.

Various Options:

ctrl+g (F1) Display this help text

ctrl+x (F2) Close the current file buffer / Exit from nano

ctrl+o (F3) Write the current file to disk

ctrl+r (F5) Insert another file into the current one

ctrl+w (F6) Search forward for a string or a regular expression

ctrl+\ (M-R) Replace a string or a regular expression

ctrl+k (F9) Cut the current line and store it in the cutbuffer

ctrl+u (F10) Uncut from the cutbuffer into the current line

But main important options:

ctrl+o → To save content

ctrl+x → To quit from the editor



Part-2



Shell Scripting



Agenda

Topic-24: Shell Scripting / Shell Programming:

Topic-25: What is shell script, Sha-Bang and First Script

Topic-26: Shell Variables

Topic-27: Variable Substitution and Command Substitution

Topic-28: Command Line Arguments

Topic-29: How to read dynamic data from the user

Topic-30: Operators

Topic-31: Control Statements

Topic-32: Arrays

Topic-33: Shell Script Functions

Topic-34: Project on Shell Programming

Topic-35: Stream Editor(SED)

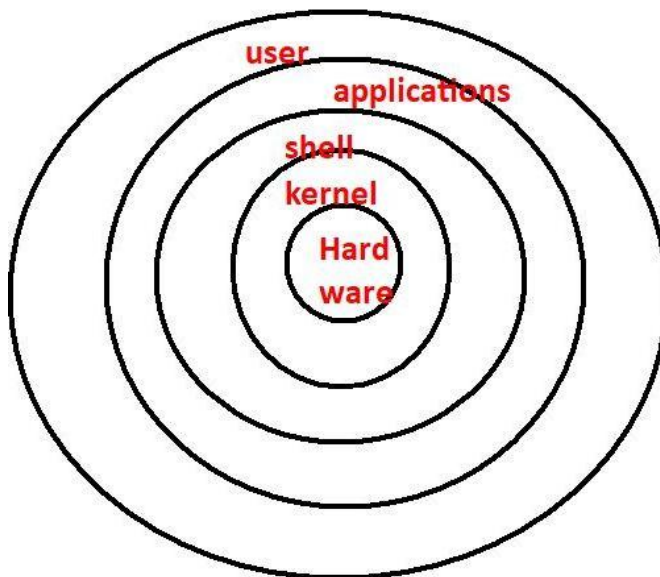
Topic-36: AWK Programming

Topic-37: Project on awk programming



Topic-24: What is Shell and Types of Shells

What is Shell?



- 1) Shell is responsible to read commands/applications provided by user.
- 2) Shell will check whether command is valid or not and whether it is properly used or not. If everything is proper then shell interpretes (converts) that command into kernel understandable form. That interpreted command will be handover to kernel.
- 3) Kernel is responsible to execute that command with the help of hardware.

Shell acts as interface between user and kernel.
shell+kernel is nothing but operating system.

Types of Shells:

There are multiple types of shells are available.

1) Bourne Shell:

- It is developed by Stephen Bourne.
- This is the first shell which is developed for UNIX.
- By using sh command we can access this shell.



2) BASH Shell:

- BASH → Bourne Again SHell.
- It is advanced version of Bourne Shell.
- This is the default shell for most of the linux flavours.
- By using bsh command we can access this shell.

3) Korn Shell:

- It is developed by David Korn.
- Mostly this shell used in IBM AIX operating system.
- By using ksh command, we can access this shell.

4) CShell:

- Developed by Bill Joy.
- C meant for California University.
- It is also by default available with UNIX.
- By using csh command, we can access this shell.

5) TShell:

- T means Terminal.
- It is the advanced version of CShell.
- This is the most commonly used shell in HP UNIX.
- By using tcsh command, we can access this shell.

6) Z Shell:

- Developed by Paul.
- By using zsh command we can access this shell.

Note: The most commonly used shell in linux environment is BASH. It is more powerful than remaining shells.

How to Check Default Shell in our System:

```
$ echo $0
bash
$ echo $SHELL
/bin/bash
```

We can also check the default shell information inside /etc/passwd file

```
$ cat /etc/passwd
```

```
durgasoft:x:1000:1000:durgasoft,,,:/home/durgasoft:/bin/bash
```



How to check all available shells in our system:

/etc/shells file contains all available shells information.

```
$ cat /etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/bin/rbash
/bin/dash
```

How to switch to other shells:

Based on our requirement we can switch from one shell to another shell.

```
durgasoft@durgasoft:~/Desktop$ sh
$ echo $0
sh
$ exit
durgasoft@durgasoft:~/Desktop$ echo $0
bash
durgasoft@durgasoft:~/Desktop$ rbash
durgasoft@durgasoft:~/Desktop$ echo $0
rbash
durgasoft@durgasoft:~/Desktop$ exit
exit
durgasoft@durgasoft:~/Desktop$ dash
$ echo $0
dash
$ exit
```



Topic-25: What is shell script, Sha-Bang and First Script

What is Shell Script:

A sequence of commands saved to a file and this file is nothing but shell script.

Inside shell script, we can also use programming features like conditional statements, loops, functions etc. Hence we can write scripts very easily for complex requirements also.

Best suitable for automated tasks.

How to write and run Shell Script:

Step - 1: Write script

demo.sh:

```
echo "Welcome to shell script"
date
cal
```

Step - 2: Provide execute permissions to the script:

```
$ chmod a+x demo.sh
```

Step - 3: Run the script

We can run the script in multiple ways

```
$ /bin/bash ./demo.sh
$ bash ./demo.sh
$ /bin/bash /home/durgasoft/scripts/demo.sh
$ ./demo.sh # default shell is bash
```

Note: The default shell is bash. Hence bash is responsible to execute our script. Instead of bash, if we want to use Bourne shell then we have to use the following command

```
$ /bin/sh ./demo.sh
$ sh ./demo.sh
```



Importance of Sha-Bang:

By using sha-bang, we can specify the interpreter which is responsible to execute the script.

→ Sharp

! → Bang

#! → Sharp Bang or Shabang or Shebang

#!/bin/bash → It means the script should be executed by bash

#!/bin/sh → It means the script should be executed by Bourne Shell

#!/usr/bin/python3 → It means the script should be executed by Python3 interpreter

If we write shabang in our script at the time of execution, we are not required to provide command to execute and we can execute script directly.

Q. Write a python script and execute without shabang and with shabang?

test.py

```
#!/usr/bin/python3
```

```
import random
```

```
name = input("Enter Your Name:")
```

```
l=["Sunny","Katrina","Kareena","Mallika","Malaika"]
```

```
print("Hello:",name)
```

```
print("Congratulations..You are going to marry:",random.choice(l))
```

Without Shabang:

```
$ python3 ./test.py
```

```
$ python3 /home/durgasoft/scripts/test.py
```

With Shabang:

```
$ ./test.py
```

```
$ /home/durgasoft/scripts/test.py
```

Q. Consider the folloiwng Script:

demo.sh

```
#!/bin/rm
```

```
echo "It is beautiful script"
```

If we execute this script what will happend?

```
$ ./demo.sh
```

It is equivalent to



```
$ rm ./demo.sh
```

demo.sh will be removed as this script executed by rm command.

Q. Write and Run shell script that prints Current system date and time.

date.sh

```
#!/bin/bash
```

```
echo "The current System Date and Time:"
```

```
date
```

Provide execute permissions for this script

```
$ chmod a+x date.sh
```

```
$/date.sh
```

How to run our script from anywhere in our system:

For any command or script, by default shell will check locations specified by PATH variable.

If we add our script location to PATH Variable value, then we can run our script from anywhere in our system. We can do this in the following two ways:

1) Session Level:

```
durgasoft@durgasoft:~/scripts$ echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

```
$ export PATH=$PATH:/home/durgasoft/scripts
```

```
durgasoft@durgasoft:~/scripts$ echo $PATH
```

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/durgasoft/scripts
```

Now we can run our script from anywhere. We are not required to provide either absolute path or relative path.

```
$date.sh
```

```
The current System Date and Time:
```

```
Wed Dec 4 21:01:49 IST 2019
```

Note: This way of setting PATH variable is applicable only for current session. Once we close terminal then automatically these changes will be lost.



2) Setting PATH permanently at User Level:

In our user home directory there is a file named with .bashrc. This file will be executed every time automatically whenever we open the terminal. If we define PATH variable in this file then that PATH value will become permanent and we are not required to set every time.

Add the following line at bottom of this file

```
export PATH=$PATH:/home/durgasoft/scripts
```

Q. What is the meaning of startup File?

.bashrc is the startup file and will be executed automatically at the time of terminal starting.

Hence if we want to perform any activity while starting the terminal we have to define that activity in this file.

eg: creating aliases and updating PATH variable value etc

Q. What is meant by logout file?

.bash_logout is logout file and will be executed automatically at the time of terminal exit.

Hence if we want to perform any activity at terminal exit, then we have to define that activity inside this file.



Topic-26: Shell Variables

Variables are place holders to hold values.

Variables are key-value pairs.

In Shell programming, there are no data types. Every value is treated as text type/ String type.

All variables are divided into 2 types

- 1) Environment variables / predefined variables
- 2) User defined variables

1) Environment Variables:

These are predefined variables and mostly used internally by the system. Hence these variables also known as System variables.

But based on our requirement, we can use these variables in our scripts.

We can get all environment variables information by using either env command or set command.

```
durgasoft@durgasoft:~$ env
LANG=en_IN
USERNAME=durgasoft
USER=durgasoft
PWD=/home/durgasoft
HOME=/home/durgasoft
SHELL=/bin/bash
LOGNAME=durgasoft
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/durgasoft/scripts
...

durgasoft@durgasoft:~$ set
BASH=/bin/bash
LANG=en_IN
```




```
LOGNAME=durgasoft
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/
games:/snap/bin:/home/durgasoft/scripts
PS1='\[\e]0;\u@\h:
\w\a\]${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[0
1;34m\]\w\[\033[00m\]\$ '
PWD=/home/durgasoft
SHELL=/bin/bash
```

How to Change Prompt:

Internally PS1 Environment variable is responsible to display terminal prompt. By reassigning the value we can change prompt.

```
durgasoft@durgasoft:~$
durgasoft@durgasoft:~$ PS1=DURGA$
DURGA$ls
abc.txt Documents d.txt Pictures scripts Videos y
Desktop Downloads Music Public Templates x
DURGA$cat > abc.txt
asdfjkasjfdsa
asfdkasklfdjlads
DURGA$PS1=Sunny#
Sunny#ls -l
total 52
-rw-r--r-- 1 durgasoft durgasoft 31 Dec 5 20:58 abc.txt
drwxr-xr-x 2 durgasoft durgasoft 4096 Dec 5 18:04 Desktop
drwxr-xr-x 2 durgasoft durgasoft 4096 Nov 21 17:06 Documents
drwxr-xr-x 2 durgasoft durgasoft 4096 Nov 21 17:06 Downloads
```

Demo Script to use some environment Variables:

env.sh

```
#!/bin/bash
echo "User Name: $USER"
echo "User Home Directory: $HOME"
echo "Default Shell Name: $SHELL"
echo "PATH: $PATH"

$chmod 755 env.sh
$./env.sh
User Name: durgasoft
User Home Directory: /home/durgasoft
Default Shell Name: /bin/bash
```



PATH:

```
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/durgasoft/scripts
```

2) User defined Variables:

Based on our programming requirement, we can define our own variables also.

```
durgasoft@durgasoft:~/scripts$ GUEST=Dhoni
durgasoft@durgasoft:~/scripts$ echo "Hello $GUEST You are my Hero"
Hello Dhoni You are my Hero
```

Rules to define Variables:

1. It is recommended to use only UPPER CASE characters.
2. If variable contains multiple words, then these words should be separated with _ symbol.
3. Variable names should not starts with digit.

```
durgasoft@durgasoft:~/scripts$ 123A=40
123A=40: command not found
```
4. We should not use special symbols like -, @, # etc

How to define readonly Variables:

We can define read only variables by using readonly keyword.

```
$A=100
$readonly A
$A=300
bash: A: readonly variable
```

If the variable is readonly then we cannot perform reassignment for that variable. It will become constant.

Variable Scopes:

There are 3 scopes available for variables.

- 1) Session Scope
- 2) User Scope
- 3) System Scope

1) Session Scope:

The variables which are declared in the terminal are said to be in session scope. Once we close the terminal (ie exit session) automatically all variables will be gone.

```
$ X=10
$ Y=10
```



2) User Scope:

- The variables which are declared inside .bashrc file, are said to be in user scope.
- These variables are available for all sessions related to current user. These variables cannot be accessed by other users.

.bashrc

....

```
export GUEST=Dhoni  
export FRIEND=Sunny
```

3) System Scope:

If the variable available for all users and for all sessions, such type of variables are said to be in system scope.

We have to declare these variables inside /etc/profile file.

But to edit this file compulsory root permissions must be required.

```
$ sudo gedit /etc/profile
```

...

```
export NAME=DURGA  
export COURSE=PYTHON
```



Topic-27: Variable Substitution and Command Substitution

Variable Substitution:

Accessing the value of a variable by using \$ symbol is called variable substitution.

Syntax:

\$variablename
\${variablename}

Recommended to use \${variablename}.

test.sh

```
#!/bin/bash
a=10
b=20
COURSE="linux"
ACTION="SLEEP"
echo "Values of a and b are: $a and $b"
echo "My Course is: ${COURSE}"
echo "My Favourite Action: $ACTIONING"
echo "My Favourite Action: ${ACTION}ING"
```

Output:

Values of a and b are: 10 and 20
My Course is: linux
My Favourite Action:
My Favourite Action: SLEEPING

Q. Consider the following variable declaration NAME="durga"
Which of the following is valid way to print value of NAME?

- A) echo NAME
- B) echo \$NAME
- C) echo 'NAME'
- D) echo "NAME"

Ans: B, D

Note: If we use single quotes then variable substitution won't be happen. But we can use double quotes.



```
durgasoft@durgasoft:~/scripts$ NAME='durga'
durgasoft@durgasoft:~/scripts$ echo NAME
NAME
durgasoft@durgasoft:~/scripts$ echo $NAME
durga
durgasoft@durgasoft:~/scripts$ echo '$NAME'
$NAME
durgasoft@durgasoft:~/scripts$ echo "$NAME"
durga
```

Command Substitution:

We can execute command and we can substitute its result based on our requirement by using command substitution.

Syntax:

Old style: ``command`` These are backquotes but not single quotes

New Style: `$(command)`

Eg 1:

```
durgasoft@durgasoft:~/scripts$ echo "Today Date is: `date +%D` "
```

Today Date is: 12/06/19

```
durgasoft@durgasoft:~/scripts$ echo "Today Date is: $(date +%D) "
```

Today Date is: 12/06/19

Eg 2:

```
durgasoft@durgasoft:~/scripts$ echo "Number of files in Current Working Directory: `ls | wc -l` "
```

Number of files in Current Working Directory: 5

```
durgasoft@durgasoft:~/scripts$ echo "Number of files in Current Working Directory: $(ls | wc -l) "
```

Number of files in Current Working Directory: 5

Eg 3:

```
durgasoft@durgasoft:~/scripts$ echo "The No of Lines in test.sh: `cat test.sh | wc -l` "
```

The No of Lines in test.sh: 9

```
durgasoft@durgasoft:~/scripts$ echo "The No of Lines in test.sh: $(wc -l test.sh) "
```

The No of Lines in test.sh: 9 test.sh

Eg 4:

```
durgasoft@durgasoft:~/scripts$ echo "My Current Working Directory: `pwd` "
```

My Current Working Directory: /home/durgasoft/scripts

```
durgasoft@durgasoft:~/scripts$ echo "My Current Working Directory: $(pwd) "
```

My Current Working Directory: /home/durgasoft/scripts



Topic-28: Command Line Arguments

The arguments which are passing from the command prompt at the time of executing our script, are called command line arguments.

`$./test.sh learning linux is very easy`

The command line arguments are learning, linux, is, very, easy.

Inside script we can access command line arguments as follows:

`$#` → Number of Arguments (5)
`$0` → Script Name (`./test.sh`)
`$1` → 1st Argument (learning)
`$2` → 2nd Argument (linux)
`$3` → 3rd Argument (is)
`$4` → 4th Argument (very)
`$5` → 5th Argument (easy)
`$*` → All Arguments (learning Linux is very easy)
`$@` → All Arguments (learning Linux is very easy)
`$?` → Represents exit code of previously executed command or script.

Eg: test.sh

```
#!/bin/bash
echo "Number of arguments: $#"
```

```
echo "Script Name: $0"
```

```
echo "First argument: $1"
```

```
echo "Second argument: $2"
```

```
echo "Third argument: $3"
```

```
echo "Fourth argument: $4"
```

```
echo "Fifth argument: $5"
```

```
echo "Total arguments: $*"
```

```
durgasoft@durgasoft:~/scripts$ ./test.sh learning linux is very easy
```

```
Number of arguments: 5
```

```
Script Name: ./test.sh
```

```
First argument: learning
```

```
Second argument: linux
```



```
Third argument: is
Fourth argument: very
Fifth argument: easy
Total arguments: learning linux is very easy
```

Difference between \$@ and \$*:

\$@ → All command line arguments with space separator

"\$1" "\$2" "\$3" ...

\$* → All command line arguments as single string

"\$1c\$2c\$3c.."

Where c is the first character of the Internal Field Separator (IFS).

The default first character is space.

How to Check Default IFS:

```
$ set | grep "IFS"
```

```
IFS=$' \t\n'
```

How to Change IFS:

```
#!/bin/bash
```

```
IFS="-"
```

```
echo "Number of arguments: $#"
```

```
echo "Script Name: $0"
```

```
echo "First argument: $1"
```

```
echo "Second argument: $2"
```

```
echo "Third argument: $3"
```

```
echo "Fourth argument: $4"
```

```
echo "Fifth argument: $5"
```

```
echo "Total arguments: $@"
```

```
echo "Total arguments: $*"
```

```
durgasoft@durgasoft:~/scripts$ test.sh learning unix is very easy
```

```
Number of arguments: 5
```

```
Script Name: /home/durgasoft/scripts/test.sh
```

```
First argument: learning
```

```
Second argument: unix
```

```
Third argument: is
```

```
Fourth argument: very
```

```
Fifth argument: easy
```

```
Total arguments: learning unix is very easy
```

```
Total arguments: learning-unix-is-very-easy
```



Note: The main purpose of command line arguments is to customize behaviour of the script.

test.sh

```
#!/bin/bash
l=$(echo -n "DURGA" | wc -c)
echo "The Length of given String: $l"
```

This script will work only for string: DURGA

test.sh

```
#!/bin/bash
len=$(echo -n "$1" | wc -c)
echo "The Length of given string $1 : $len"
```

This script will work for any string provided from command prompt.

```
durgasoft@durgasoft:~/scripts$ test.sh
The Length of given string : 0
durgasoft@durgasoft:~/scripts$ test.sh durgasoft
The Length of given durgasoft : 9
durgasoft@durgasoft:~/scripts$ test.sh adsfkjshfdjkhsfkhkjsfhk
The Length of given string adsfkjshfdjkhsfkhkjsfhk : 24
```

Q. Write Script to Create Log File with Timestamp

test.sh

```
#!/bin/bash
timestamp=$(date +%d_%m_%Y_%H_%M_%S)
echo "This is data to log file" >> ${timestamp}.log
echo "This is extra data to log file" >> ${timestamp}.log
date >> ${timestamp}.log
echo >> ${timestamp}.log
echo "Data Written to log file successfully"
```

If we execute this script, every time a new file will be created.

If we take

```
timestamp=$(date +%d_%m_%Y_%H_%M)
```

Then log file will be created for every new minute.

For every hour if want a new log file: `timestamp=$(date +%d_%m_%Y_%H)`

For every day if want a new log file: `timestamp=$(date +%d_%m_%Y)`



Topic-29: How to Read Dynamic Data from the User

By using read keyword we can read dynamic data from the end user.

Without Prompt Message:

```
durgasoft@durgasoft:~/scripts$ read a b
```

```
100 200
```

```
durgasoft@durgasoft:~/scripts$ echo "The values of a and b are:$a and $b"
```

```
The values of a and b are:100 and 200
```

With Prompt Message:

Approach-1

test.sh

```
#!/bin/bash
```

```
echo "Enter A Value:"
```

```
read A
```

```
echo "Enter B Value:"
```

```
read B
```

```
echo "A Value: $A"
```

```
echo "B Value: $B"
```

```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
Enter A Value:
```

```
100
```

```
Enter B Value:
```

```
200
```

```
A Value: 100
```

```
B Value: 200
```

test.sh

```
#!/bin/bash
```

```
echo -n "Enter A Value:"
```

```
read A
```

```
echo -n "Enter B Value:"
```

```
read B
```



```
echo "A Value: $A"
echo "B Value: $B"
durgasoft@durgasoft:~/scripts$ test.sh
Enter A Value:100
Enter B Value:200
A Value: 100
B Value: 200
```

Approach-2

```
#!/bin/bash
read -p "Enter A Value:" A
read -p "Enter B Value:" B
```

```
echo "A Value: $A"
echo "B Value: $B"
```

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter A Value:100
Enter B Value:200
A Value: 100
B Value: 200
```

```
#!/bin/bash
read -p "Enter User Name:" user
read -p "Enter Password:" password
```

```
echo "$user thanks for your information"
```

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter User Name:durga
Enter Password:anushka123
durga thanks for your information
```

Note:

read -p : Just to display prompt message
read -s : It hides input on screen which is provided by end user.

```
#!/bin/bash
read -p "Enter User Name:" user
read -s -p "Enter Password:" password
echo
echo "Hello $user , thanks for your information"
```

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter User Name:Durga
```



Enter Password:

Hello Durga , thanks for your information

Q. Write a script to read student data and display to the console for confirmation?

```
#!/bin/bash
read -p "Enter Student Name:" name
read -p "Enter Student RollNo:" rollno
read -p "Enter Student Age:" age
read -p "Enter Student Marks:" marks
echo "Please confirm your details"
echo "-----"
echo "Student Name: $name"
echo "Student Rollno: $rollno"
echo "Student Age: $age"
echo "Student Marks: $marks"
```

durgasoft@durgasoft:~/scripts\$ test.sh

Enter Student Name:Durga

Enter Student RollNo:101

Enter Student Age:40

Enter Student Marks:89

Please confirm your details

Student Name: Durga

Student Rollno: 101

Student Age: 40

Student Marks: 89

Q. Write a script to read employee details and save to emp.txt file?

```
#!/bin/bash
read -p "Enter Employee Number:" eno
read -p "Enter Employee Name:" ename
read -p "Enter Employee Salary:" esal
read -p "Enter Employee Address:" eaddr

echo "Below details are saved to the file"
echo "$eno:$ename:$esal:$eaddr"
echo "$eno:$ename:$esal:$eaddr" >> emp.txt
```

durgasoft@durgasoft:~/scripts\$ test.sh

Enter Employee Number:100

Enter Employee Name:Sunny

Enter Employee Salary:1000

Enter Employee Address:Mumbai



Below details are saved to the file

100:Sunny:1000:Mumbai

durgasoft@durgasoft:~/scripts\$ test.sh

Enter Employee Number:200

Enter Employee Name:Bunny

Enter Employee Salary:2000

Enter Employee Address:Hyderabad

Below details are saved to the file

200:Bunny:2000:Hyderabad

durgasoft@durgasoft:~/scripts\$ test.sh

Enter Employee Number:300

Enter Employee Name:Chinny

Enter Employee Salary:3000

Enter Employee Address:Chennai

Below details are saved to the file

300:Chinny:3000:Chennai

durgasoft@durgasoft:~/scripts\$

durgasoft@durgasoft:~/scripts\$ cat emp.txt

100:Sunny:1000:Mumbai

200:Bunny:2000:Hyderabad

300:Chinny:3000:Chennai

emp.txt

durgasoft@durgasoft:~/scripts\$ cat emp.txt

100:Sunny:1000:Mumbai

200:Bunny:2000:Hyderabad

300:Chinny:3000:Chennai

Q. Write a Script that takes a string from the end user and print its length?

```
#!/bin/bash
```

```
read -p "Enter any string to find length:" str
```

```
len=$(echo -n $str | wc -c)
```

```
echo "Length of $str : $len"
```

durgasoft@durgasoft:~/scripts\$ test.sh

Enter any string to find length:apple

Length of apple : 5

durgasoft@durgasoft:~/scripts\$ test.sh

Enter any string to find length:durgasoftwaresolutions

Length of durgasoftwaresolutions : 22



Q. Write a script to read filename from the end user and display its content?

#!/bin/bash

```
read -p "Enter any File name to display its content:" fname
echo "-----"
cat $fname
echo "-----"
durgasoft@durgasoft:~/scripts$ test.sh
Enter any File name to display its content:emp.txt
```

```
-----
100:Sunny:1000:Mumbai
200:Bunny:2000:Hyderabad
300:Chinny:3000:Chennai
-----
```

Q. Write a script to read filename from the end user and remove blank lines present in that file?

#!/bin/bash

```
read -p "Enter any File name to remove blank lines:" fname
grep -v "^$" $fname > temp.txt
mv temp.txt $fname
echo "In $fname all blank lines deleted"
```

Q. Write a script to read filename from the end user and remove duplicate lines present in that file?

#!/bin/bash

```
read -p "Enter any File name to remove duplicate lines:" fname

sort -u $fname > temp.txt
mv temp.txt $fname
echo "In $fname all duplicate lines deleted"
```



Topic-30: Operators

1. Arithmetic Operators

- + → Addition
- → Subtraction
- * → Multiplication (we should use * as it is wild card character)
- / → Division
- % → Modulo Operator

Relational Operators: (Numeric Comparison Operators)

- gt → Greater than
- ge → Greater than or equal to
- lt → Less than
- le → Less than or equal to
- eq → Is equal to
- ne → Not equal to

These operators return boolean value (true/false)

Logical Operators:

- a → Logical AND
- o → Logical OR
- ! → Logical Not

Assignment operator =

Note: Except assignment operator, for all operators we have to provide space before and after operator.

How to perform Mathematical Operations:

There are multiple ways

- 1) By using expr keyword
- 2) By using let keyword
- 3) By using (())
- 4) By using []



1) By using expr Keyword:

expr means expression and it is a keyword.

```
#!/bin/bash
read -p "Enter First Number:" a
read -p "Enter First Number:" b
```

```
sum=`expr $a + $b`
echo "The Sum: $sum"
```

```
sum=$(expr $a + $b)
echo "The Sum: $sum"
```

Note: While using expr keyword, \$ symbol is mandatory.
Space must be required before and after + symbol.

2) By using let Keyword:

```
let sum=a+b
echo "The Sum: $sum"
```

```
let sum=$a+$b
echo "The Sum: $sum"
```

Here \$ symbol is optional. But we should not provide space before and after +.

3) By using (()):

```
sum=$((a+b))
echo "The Sum: $sum"
```

```
sum=$(( $a+$b ))
echo "The Sum: $sum"
```

Here space and \$ symbol, both are optional.

4) By using []:

```
sum=$((a+b))
echo "The Sum: $sum"
```

```
sum=$(( $a+$b ))
echo "The Sum: $sum"
```

Here also space and \$ symbol, both are optional.

Note: All the above 4 approaches will work only for integral arithmetic (only for integer numbers).
If we want to perform floating point arithmetic then we should use bc command.



bc Command:

bc means binary calculator

We can start binary calculator by using bc command.

```
durgasoft@durgasoft:~/scripts$ bc
```

```
bc 1.07.1
```

```
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software  
Foundation, Inc.
```

```
This is free software with ABSOLUTELY NO WARRANTY.
```

```
For details type `warranty'.
```

```
10.5+30.6
```

```
41.1
```

```
10.2^2
```

```
104.0
```

```
10.5*3.6
```

```
37.8
```

```
ctrl+d → To exit bc
```

Note: bc can perform both integral and floating point arithmetic.

Script for floating Point Arithmetic:

```
#!/bin/bash
```

```
read -p "Enter First Number:" a
```

```
read -p "Enter Second Number:" b
```

```
sum=$( echo $a+$b | bc)
```

```
echo "The Sum:$sum"
```

```
echo "The Difference: $( echo $a-$b | bc)"
```

```
echo "The Product: $( echo $a*$b | bc)"
```

output:

```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
Enter First Number:10.5
```

```
Enter Second Number:5.3
```

```
The Sum:15.8
```

```
The Difference: 5.2
```

```
The Product: 55.6
```




Q. Write a script to read two integer numbers and print sum?

```
#!/bin/bash
```

```
read -p "Enter First Number:" a
read -p "Enter Second Number:" b
sum=$((a+b))
echo "The Sum:$sum"
```

Q. Write a script to read 4-digit integer number and print the sum of digits present in that number?

```
#!/bin/bash
```

```
read -p "Enter Any 4-digit Integer Number:" n
```

```
a=$(( echo $n | cut -c 1 ))
b=$(( echo $n | cut -c 2 ))
c=$(( echo $n | cut -c 3 ))
d=$(( echo $n | cut -c 4 ))
```

```
echo "The Sum of 4 digits : $((a+b+c+d))"
```

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter Any 4-digit Integer Number:1234
The Sum of 4 digits : 10
durgasoft@durgasoft:~/scripts$ test.sh
Enter Any 4-digit Integer Number:3456
The Sum of 4 digits : 18
```

Q. Write a script to read employee monthly salary and print his bonus. The bonus should be 25% of annual salary

```
#!/bin/bash
```

```
read -p "Enter Employee Monthly Salary:" salary
```

```
annual_salary=$((salary*12))
bonus=$((annual_salary*25/100))
echo "The Bonus:$bonus"
```

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter Employee Monthly Salary:10000
The Bonus:30000
```



Topic-31: Control Statements

- 1) if statement
- 2) case statement
- 3) while loop
- 4) for loop
- 5) until loop
- 6) break
- 7) continue
- 8) exit

1) if Statement:

There are 4 types of if statements

- 1) simple if
- 2) if-else
- 3) nested if
- 4) ladder if

1) simple if:

```
if [ condition ]  
then  
    action  
fi
```

If condition is true then only action will be executed.

Q. WAS to read name from the end user and if name is sunny then display some special message?

```
#!/bin/bash
```

```
read -p "Enter Your Name:" name
```

```
if [ $name = "sunny" ]  
then  
    echo "Hello Sunny Very Good Evening"  
fi  
echo "How are you"
```

```
durgasoft@durgasoft:~/scripts$ test.sh  
Enter Your Name:durga  
How are you
```



```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
Enter Your Name:sunny
```

```
Hello Sunny Very Good Evening
```

```
How are you
```

```
Note:
```

```
x=10 → Assignment
```

```
x = 10 → Comparison
```

2) if -else:

```
if [ condition]
```

```
then
```

```
    action if condition is true
```

```
else
```

```
    action if condition is false
```

```
fi
```

3) Nested if:

```
if [ condition ]
```

```
then
```

```
    .....
```

```
    .....
```

```
    if [ condition ]
```

```
    then
```

```
        .....
```

```
        .....
```

```
    else
```

```
        .....
```

```
    fi
```

```
    .....
```

```
else
```

```
    .....
```

```
fi
```

4) ladder -if:

```
if [condition]
```

```
then
```

```
    action-1
```

```
elif [ condition ]
```

```
then
```

```
    action-2
```

```
elif [ condition ]
```

```
then
```

```
    action-3
```

```
else
```

```
    default action
```

```
fi
```



Q. Write a script to find greater of two numbers?

```
#!/bin/bash
```

```
read -p "Enter First Number:" a
read -p "Enter Second Number:" b
```

```
if [ $a -gt $b ]
then
    echo "Greater Number is:$a"
else
    echo "Greater Number is:$b"
fi
```

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter First Number:10
Enter Second Number:20
Greater Number is:20
durgasoft@durgasoft:~/scripts$ test.sh
Enter First Number:20
Enter Second Number:10
Greater Number is:20
durgasoft@durgasoft:~/scripts$ test.sh
Enter First Number:-10
Enter Second Number:-20
Greater Number is:-10
```

Q. Write a script to check whether numbers are equal or not. If the numbers are not equal then print greater number?

```
#!/bin/bash
```

```
read -p "Enter First Number:" a
read -p "Enter Second Number:" b
```

```
if [ $a -eq $b ]
then
    echo "Both Numbers are equal"
elif [ $a -gt $b ]
then
    echo "First Number is greater than Second Number"
else
    echo "First Number is less than Second Number"
fi
```



Q. Write a script to find greater of 3 numbers?

1st Way:

#!/bin/bash

```
read -p "Enter First Number:" a
read -p "Enter Second Number:" b
read -p "Enter Third Number:" c
```

```
if [ $a -gt $b ]
then
    if [ $a -gt $c ]
    then
        echo "The Greater Number:$a"
    else
        echo "The Greater Number:$c"
    fi
elif [ $b -gt $c ]
then
    echo "The Greater Number:$b"
else
    echo "The Greater Number:$c"
fi
```

2nd Way:

#!/bin/bash

```
read -p "Enter First Number:" a
read -p "Enter Second Number:" b
read -p "Enter Third Number:" c
```

```
if [ $a -gt $b -a $a -gt $c ]
then
    echo "The Greater Number: $a"
elif [ $b -gt $c ]
then
    echo "The Greater Number: $b"
else
    echo "The Greater Number: $c"
fi
```

WAS to check whether student is pass or fail based on given 3 subjects marks.
In any subject if marks less than 35 then status is fail.



1st Way:

#!/bin/bash

```
read -p "Enter First Subject Marks:" a
read -p "Enter Second Subject Marks:" b
read -p "Enter Third Subject Marks:" c
```

```
if [ $a -lt 35 ]
then
    echo "Student Failed"
elif [ $b -lt 35 ]
then
    echo "Student Failed"
elif [ $c -lt 35 ]
then
    echo "Student Failed"
else
    echo "Result is Pass"
fi
```

2nd Way:

#!/bin/bash

```
read -p "Enter First Subject Marks:" a
read -p "Enter Second Subject Marks:" b
read -p "Enter Third Subject Marks:" c
```

```
if [ $a -ge 35 -a $b -ge 35 -a $c -ge 35 ]
then
    echo "Result is Pass"
else
    echo "Result is Fail"
fi
```

The Funniest Example with if-else

#!/bin/bash

```
read -p "Enter Your Favourite Brand:" brand
```

```
if [ $brand = "KF" ]
then
    echo "It is Childrens Brand"
```

```
elif [ $brand = "KO" ]
then
```



```
echo "It is not that much Kick"
```

```
elif [ $brand = "RC" ]
```

```
then
```

```
    echo "It is too light"
```

```
elif [ $brand = "FO" ]
```

```
then
```

```
    echo "Buy One Get One FREE"
```

```
else
```

```
    echo "Other Brands are not recommended"
```

```
fi
```

```
durgasoft@durgasoft:~/scripts$ ./test.sh
```

```
Enter Your Favourite Brand:KF
```

```
It is Childrens Brand
```

```
durgasoft@durgasoft:~/scripts$ ./test.sh
```

```
Enter Your Favourite Brand:KO
```

```
It is not that much Kick
```

```
durgasoft@durgasoft:~/scripts$ ./test.sh
```

```
Enter Your Favourite Brand:RC
```

```
It is too light
```

```
durgasoft@durgasoft:~/scripts$ ./test.sh
```

```
Enter Your Favourite Brand:FO
```

```
Buy One Get One FREE
```

```
durgasoft@durgasoft:~/scripts$
```

```
durgasoft@durgasoft:~/scripts$ ./test.sh
```

```
Enter Your Favourite Brand:KALYANI
```

```
Other Brands are not recommended
```

Exit Codes/ Status Codes:

Every command and script return some value after execution, which indicates that whether it is successfully executed or not. This return value is called exit code or status code.

We can find exit code by using "\$?".

zero means command/script executed successfully.

non-zero means command/scrip not executed successfully.

```
durgasoft@durgasoft:~/scripts$ echo "Hello Friends"
```

```
Hello Friends
```

```
durgasoft@durgasoft:~/scripts$ echo "$?"
```

```
0
```

```
durgasoft@durgasoft:~/scripts$ rm sdfjkslajfdksajfjdsakjfdksajfkd
```



```
rm: cannot remove 'sdfjkslajfdksajfjdsakjfdksajfd': No such file or directory
durgasoft@durgasoft:~/scripts$ echo "$?"
```

1

exit Command:

In the script to stop execution in the middle, we can use exit command.

Syntax:

exit status_code

The allowed values for status_code are 0 to 255.

256 → 0

257 → 1

258 → 2

259 → 3

....

Q. Write a script that takes two integer numbers as command line arguments and prints sum

If number of arguments is not 2, then we have to get message saying "You should provide only 2 arguments"

If the arguments are not integer numbers then we have to get message saying "You should provide integer numbers only"

test.sh

```
#!/bin/bash
```

```
if [ $# -ne 2 ]
```

```
then
```

```
    echo "You Should provide exactly two arguments"
```

```
    exit 1
```

```
fi
```

```
x=$1
```

```
y=$2
```

```
sum=`expr $x + $y`
```

```
if [ $? -ne 0 ]
```

```
then
```

```
    echo "You should provide integer numbers only"
```

```
    exit 2
```

```
else
```

```
    echo "The Sum:$sum"
```

```
fi
```




Output:

```
durgasoft@durgasoft:~/scripts$ test.sh 10 20
The Sum:30
durgasoft@durgasoft:~/scripts$ test.sh
You Should provide exactly two arguments
durgasoft@durgasoft:~/scripts$ test.sh 10
You Should provide exactly two arguments
durgasoft@durgasoft:~/scripts$ test.sh 10 20 30
You Should provide exactly two arguments
durgasoft@durgasoft:~/scripts$ test.sh 10 durga
expr: non-integer argument
You should provide integer numbers only
```

Q. Write a script that reads an integer number and check whether the given number is +ve number or -ve number?

```
#!/bin/bash

read -p "Enter integer number to check:" n
if [ $n -gt 0 ]; then
    echo "$n is +ve number"
else
    echo "$n is -ve number"
fi
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter integer number to check:10
10 is +ve number
durgasoft@durgasoft:~/scripts$ test.sh
Enter integer number to check:-10
-10 is -ve number
```

Note: If we want to take then in the same line then we should use ;
if [\$n -gt 0]; then
The advantage is we can reduce the number of lines.

Q. Write a script that reads an integer number and checks whether it is even number or not?

```
#!/bin/bash

read -p "Enter integer number to check:" n
if [ ${n%2} -eq 0 ]; then
    echo "$n is even number"
else
```



```
echo "$n is odd number"
fi
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter integer number to check:10
10 is even number
durgasoft@durgasoft:~/scripts$ test.sh
Enter integer number to check:15
15 is odd number
```

Q. Write a script that reads an integer number and checks whether it is 3-digit number or not?

```
#!/bin/bash
```

```
read -p "Enter integer number to check:" n

if [ $n -ge 100 -a $n -le 999 ]; then
    echo "$n is 3-digit number"
else
    echo "$n is not 3-digit number"
fi
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter integer number to check:123
123 is 3-digit number
durgasoft@durgasoft:~/scripts$ test.sh
Enter integer number to check:45
45 is not 3-digit number
durgasoft@durgasoft:~/scripts$ test.sh
Enter integer number to check:1234
1234 is not 3-digit number
```

File Test Options:

- e → Returns true if file/directory exists
- s → Returns true if the file is non-empty
- f → Returns true if the file is a regular file
- d → Returns true if the file is a directory
- l → Returns true if the file is link file
- b → Returns true if the file is block special file
- c → Returns true if the file is character special file
- r → Returns true if current user has read permission on the file
- w → Returns true if current user has write permission on the file



-x → Returns true if current user has execute permission on the file
-o → Returns true if current user is owner of the file.
file1 -ot file2 → Returns true if file1 is older than file2 (last modified time)
file1 -nt file2 → Returns true if file1 is newer than file2 (last modified time)

Eg 1: Script to test whether the given file exists or not?

#!/bin/bash

read -p "Enter File Name to test:" fname

```
if [ -e $fname ]
then
    echo "$fname exists"
else
    echo "$fname not exists"
fi
```

Eg 2: Script to test whether the given file is regular file or directory?

#!/bin/bash

read -p "Enter File Name to test:" fname

```
if [ -e $fname ]; then
    if [ -f $fname ]; then
        echo "It is regular file"
    elif [ -d $fname ]; then
        echo "It is Directory file"
    else
        echo "It is special file"
    fi
else
    echo "$fname does not exist"
fi
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter File Name to test:a.txt
It is regular file
durgasoft@durgasoft:~/scripts$ test.sh
Enter File Name to test:dir1
It is Directory file
durgasoft@durgasoft:~/scripts$ test.sh
Enter File Name to test:dddddd
dddddd does not exist
durgasoft@durgasoft:~/scripts$ test.sh
```



Enter File Name to test:/bin
It is Directory file
durgasoft@durgasoft:~/scripts\$ test.sh
Enter File Name to test:/dev/vcsa3
It is special file

Write a script that reads a file name and display its content to the terminal?

#!/bin/bash

```
read -p "Enter File Name to test:" fname

if [ -e $fname ]; then
    if [ -f $fname ]; then
        if [ -r $fname ]; then
            cat $fname
        else
            echo "User not having Read permission"
        fi
    else
        echo "It is not a regular file"
    fi
else
    echo "$fname does not exist"
fi
```

Q. Write a script that reads filename and check whether it is empty file or not?

#!/bin/bash

```
read -p "Enter File Name to test:" fname

if [ -e $fname ]; then
    if [ -f $fname ]; then
        if [ -s $fname ]; then
            echo "$fname is not empty file"
        else
            echo "$fname is empty file"
        fi
    else
        echo "It is not a regular file"
    fi
else
    echo "$fname does not exist"
fi
```



Q. Write a script that accepts 2 file names and check whether both files having same content or not?

```
#!/bin/bash
```

```
read -p "Enter First File Name: " fname1
read -p "Enter Second File Name: " fname2

result=$(cmp $fname1 $fname2)
if [ -z "$result" ]; then
    echo "The given 2 files having same content"
else
    echo "The given 2 files having different content"
fi
```

Note:

-z is string comparison option.
returns True if the string is empty.

Q. Write a script that accepts a file name and display user permissions?

```
#!/bin/bash
```

```
read -p "Enter First File Name: " fname
```

```
READ=NO
WRITE=NO
EXECUTE=NO
```

```
if [ -r $fname ]; then
    READ=YES
fi
```

```
if [ -w $fname ]; then
    WRITE=YES
fi
```

```
if [ -x $fname ]; then
    EXECUTE=YES
fi
```

```
echo "User Permissions Summary"
echo "-----"
echo "Read Permission: $READ"
echo "Write Permission: $WRITE"
echo "Execute Permission: $EXECUTE"
```



Output:

durgasoft@durgasoft:~/scripts\$ test.sh
Enter First File Name: a.txt

User Permissions Summary

Read Permission: YES
Write Permission: YES
Execute Permission: NO

Q. Write a script that reads file name and remove the specified file?

```
#!/bin/bash

read -p "Enter file/directory name to delete:" fname
if [ -e $fname ]; then
    rm -r $fname
    echo "$fname removed successfully"
else
    echo "$fname does not exist"
fi
```

Mini Application:

Copy all files and directories present in the first directory to the second directory. We should create compressed tar file and have to move that tar file.
After moving tar file to the second directory, extract all files and directories and remove that tar file.
copy /home/durgasoft/x /home/durgasoft/y

Tests to perform:

- 1) The number of command line arguments should be 2
- 2) The source and destination directories should be available already
- 3) The source and destination arguments should be directories
- 4) The user should have read and execute permissions on source directory
- 5) The user should have write and execute permissions on destination directory
- 6) All error messages should be sent to error file and the file name should contain timestamp.
- 7) All intermediate steps should be displayed to the terminal.

String Test Options:

- 1) str1 = str2 → Returns true if both strings are same
- 2) str1 != str2 → Returns true if both strings are different
- 3) -z str → Returns true if the string is empty
- 4) -n str → Returns true if the string is not empty
- 5) str1 > str2 → Returns true if the str1 is alphabetically greater than str2
- 5) str1 < str2 → Returns true if the str1 is alphabetically less than str2



Demo Program For String Comparison:

```
#!/bin/bash
```

```
read -p "Enter First String:" str1
read -p "Enter Second String:" str2
```

```
if [ $str1 = $str2 ]; then
    echo "Both strings are equal"

elif [ $str1 \< $str2 ]; then
    echo "$str1 is less than $str2"
else
    echo "$str1 is greater than $str2"
fi
```

Note:

```
elif [ $str1 \< $str2 ]; then
```

If we are not using \ symbol then < acts as input redirection operator. To consider < as symbol only we should use \.

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter First String:Durga
Enter Second String:Durga
Both strings are equal
durgasoft@durgasoft:~/scripts$ test.sh
Enter First String:Apple
Enter Second String:Banana
Apple is less than Banana
durgasoft@durgasoft:~/scripts$ test.sh
Enter First String:Banana
Enter Second String:Apple
Banana is greater than Apple
```

Q. Write a script that checks whether login user is root or not. If login user is root then display first 5 current running processes information.

```
#!/bin/bash
```

```
user=$(whoami)

if [ $user != "root" ]; then
    echo "Current user not root user and hence exiting"
    exit 1
fi
```



```
echo "The Five Current Running Processes information"
echo "===== "
ps -ef | head -5
```

```
durgasoft@durgasoft:~/scripts$ test.sh
Current user not root user and hence exiting
durgasoft@durgasoft:~/scripts$ sudo -i
[sudo] password for durgasoft:
root@durgasoft:~# whoami
root
root@durgasoft:~# test.sh
test.sh: command not found
root@durgasoft:~# pwd
/root
root@durgasoft:~# /home/durgasoft/scripts/test.sh
```

The Five Current Running Processes information

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	19:32	?	00:00:03	/sbin/init splash
root	2	0	0	19:32	?	00:00:00	[kthreadd]
root	3	2	0	19:32	?	00:00:00	[rcu_gp]
root	4	2	0	19:32	?	00:00:00	[rcu_par_gp]

Q. Write a script to test whether the given string is empty or not? If it is not empty then print its length.

```
#!/bin/bash

read -p "Enter Any String to test:" str

if [ -z $str ]; then
    echo "Provided input string is empty string"
else
    echo "Provided input string is not empty string"
    echo "Its length is : $( echo -n $str | wc -c )"
fi
```




2) case Statement:

If multiple options are available then it is not recommended to use nested if-else statements. It increases length of the code and reduces readability.

To handle such type of requirements we should go for case statement.

Syntax:

case \$variable in

```
    option1 )
        action-1
        ;;
    option2 )
        action-2
        ;;
    option3 )
        action-3
        ;;
    * )
        default action
        ;;
esac
```

Note:

- 1) space is optional while defining options.
- 2) ;; can be used to come out of case statement.
- 3) ;; is mandatory for all options except for default option.
- 4) If we take default option at the beginning, then for any input the same default option will be executed.

Eg 1: Write a script to read a number from 0 to 9 and print equivalent English word?

#!/bin/bash

read -p "Enter Any digit from 0 to 9:" n

```
case $n in
    0) echo "ZERO"
        ;;
    1) echo "ONE"
        ;;
    2) echo "TWO"
        ;;
    3) echo "THREE"
        ;;
    *) echo "Invalid input"
        ;;
esac
```



```
4) echo "FOUR"
;;
5) echo "FIVE"
;;
6) echo "SIX"
;;
7) echo "SEVEN"
;;
8) echo "EIGHT"
;;
9) echo "NINE"
;;
*) echo "Please enter a digit from 0 to 9 only"
esac
```

Q. Write a script that reads favourite brand and prints corresponding meaningful message?

```
#!/bin/bash
```

```
read -p "Enter Your Favourite Brand: " brand
```

```
case $brand in
    "KF")
        echo "It is childrens brand"
        ;;
    "KO")
        echo "It is not that much kick"
        ;;
    "RC")
        echo "It is too light"
        ;;
    "FO")
        echo "Buy one get one FREE"
        ;;
    *)
        echo "Other brands are not recommended"
esac
```



Q. Write a script that accepts a single character and check whether the given character is alphabet or digit or special character?

```
#!/bin/bash
```

```
read -p "Enter Any Character to check: " ch
```

```
case $ch in
```

```
  [a-zA-Z])
```

```
    echo "It is an Alphabet symbol"
```

```
    ;;
```

```
  [0-9])
```

```
    echo "It is a Digit"
```

```
    ;;
```

```
  [^a-zA-Z0-9])
```

```
    echo "It is a Special Symbol"
```

```
    ;;
```

```
*)
```

```
  echo "Enter only one character"
```

```
esac
```

```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
Enter Any Character to check: a
```

```
It is an Alphabet symbol
```

```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
Enter Any Character to check: 8
```

```
It is a Digit
```

```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
Enter Any Character to check: $
```

```
It is a Special Symbol
```

```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
Enter Any Character to check: durga
```

```
Enter only one character
```



Q. Write a script that accepts a single character and checks whether it is digit or special character or vowel or consonant?

```
#!/bin/bash
```

```
read -p "Enter Any Character to check: " ch
```

```
case $ch in
    [^a-zA-Z0-9])
        echo "It is a Special Character"
        ;;
    [0-9])
        echo "It is a Digit"
        ;;
    [aeiouAEIOU])
        echo "It is a Vowel"
        ;;
    [^aeiouAEIOU])
        echo "It is a Consonant"
        ;;
    *)
        echo "Enter only one character"
esac
```

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter Any Character to check: a
It is a Vowel
durgasoft@durgasoft:~/scripts$ test.sh
Enter Any Character to check: s
It is a Consonant
durgasoft@durgasoft:~/scripts$ test.sh
Enter Any Character to check: 7
It is a Digit
durgasoft@durgasoft:~/scripts$ test.sh
Enter Any Character to check: $
It is a Special Character
```

Q. Write a script that performs file operations based on provided option?

- A → Display Content
- B → Append Content
- C → Overwrite Content
- D → Delete Content



The file name is abc.txt

```
#!/bin/bash
echo "A → Display Content"
echo "B → Append Content"
echo "C → Overwrite Content"
echo "D → Delete Content"
read -p "Choose Your Option A|B|C|D: " option

case $option in
    A)
        if [ ! -s "abc.txt" ]; then
            echo "It is an empty file"
        else
            echo "The content of the file is:"
            echo "-----"
            cat abc.txt
        fi
        ;;
    B)
        read -p "Provide your data to append:" data
        echo $data >> abc.txt
        echo "Data Appended"
        ;;
    C)
        read -p "Provide your data to overwrite:" data
        echo $data > abc.txt
        echo "Old data Overwritten with new data "
        ;;
    D)
        cat /dev/null > abc.txt
        echo "Deleted the content of the file"
        ;;
    *)
        echo "Choose only A|B|C|D. Execute Again"
esac
```

Q. Write a script that reads two integer numbers and perform required mathematical operation based on provided option?

- 1 → Addition Operation
- 2 → Subtraction Operation
- 3 → Multiplication Operation
- 4 → Division Operation



#!/bin/bash

```
read -p "Enter First Number: " n1
read -p "Enter Second Number: " n2
echo ""
echo "1 --> Addition Operation"
echo "2 --> Subtraction Operation"
echo "3 --> Multiplication Operation"
echo "4 --> Division Operation"
read -p "Choose Your Option 1|2|3|4: " option
```

case \$option in

```
1)
    echo "$n1 + $n2 = $((n1+n2))"
    ;;
2)
    echo "$n1 - $n2 = $((n1-n2))"
    ;;
3)
    echo "$n1 * $n2 = $((n1*n2))"
    ;;
4)
    echo "$n1 / $n2 = $((n1/n2))"
    ;;
*)
    echo "Choose only 1|2|3|4. Execute Again"
```

esac

Iterative Statements:

If we want to execute a group of commands multiple times then we should go for iterative statements.

There are 3 types of iterative statements

- 1) while loop
- 2) until loop
- 3) for loop



1) while Loop:

If we don't know the number of iterations in advance, then we should go for while loop.

Syntax:

```
while [ condition ]  
do  
    body  
done
```

As long as condition is true, then body will be executed. Once condition fails then only loop will be terminated.

Q1. Write a script to print numbers from 1 to 10

```
#!/bin/bash
```

```
i=1
```

```
while [ $i -le 10 ]  
do  
    echo $i  
    let i++  
done
```

Q2. Write a script to generate numbers until limit which is provided by end user?

```
#!/bin/bash
```

```
read -p "Enter Limit:" n  
i=1
```

```
while [ $i -le $n ]  
do  
    echo $i  
    sleep 2  
    let i++  
done
```

Note: If we don't want to perform any operation for a particular amount of time (i.e just pausing is required) then we should go for sleep command. The argument to the sleep command is seconds.

Eg:

sleep 2

sleep 3

sleep 0.5

Q3. Write a script to find the sum of first n integers, where n is provided by end user?

```
#!/bin/bash
```

```
read -p "Enter n value:" n
```

i=1

sum=0

while [\$i -le \$n]

do

```
let sum=sum+i
```

let i++

done

```
echo "The Sum of first $n numbers: $sum"
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
```

Enter n value:3

The Sum of first 3 numbers: 6

```
durgasoft@durgasoft:~/scripts$ test.sh
```

Enter n value:10

The Sum of first 10 numbers: 55

```
durgasoft@durgasoft:~/scripts$ test.sh
```

Enter n value:12345

The Sum of first 12345 numbers: 76205685

Q. Write a script to display timer (digital timer)?

```
#!/bin/bash
```

```
while [ true ]
```

do

clear

```
printf "\n\n\n\n\n\n\t\t\t\t\t$(date +%H:%M:%S)"
```

sleep 1

done

Note: To use escape characters like \n and \t, we should not use echo and we should use printf command.

Note: true and false are keywords which represents boolean values.

break Statement:

Based on some condition, if we want to break loop execution (i.e to come out of loop) then we should go for break statement.

Eg 1:

```
#!/bin/bash
```

i=1

```
while [ $i -le 10 ]
do
    if [ $i -eq 5 ]; then
        break
    fi
    echo $i
    let i++
done
```

Output:

1
2
3
4

Eg 2:

```
#!/bin/bash
```

```
while [ true ]
do
    clear
    printf "\n\n\n\n\n\n\n\t\t\t$(date +%H:%M:%S)"
    sleep 1
    h=$(date +%H)
    m=$(date +%M)
    if [ $h -eq 20 -a $m -eq 35 ]; then
        break
    fi
done
```



continue Statement:

We can use continue statement to skip current iteration and continue for the next iteration.

Eg:

```
#!/bin/bash
```

```
i=0
```

```
while [ $i -lt 10 ]
do
    let i++
    if [ ${i%2} -eq 0 ]; then
        continue
    fi
    echo $i
done
```

Output:

```
1
3
5
7
9
```

Write a script to read file name and display its content?

```
#!/bin/bash
```

```
while [ true ]
do
    read -p "Enter File Name to Display content: " fname

    # Checking whether the file exists or not and whether regular file or not
    if [ -f $fname ]; then
        echo "The content of $fname :"
        echo "-----"
        cat $fname
    else
        echo "$fname does not exist"
    fi

    read -p "Do you want to display content of another file [Yes|No]:" option
    case $option in
        [yY]|[Yy])[eE][sS])
```



```
        continue
    ;;
    [nN]||[nN][oO])
        break
    ;;
esac
done
echo "Thanks for using application"
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
Enter File Name to Display content: abc.txt
```

The content of abc.txt:

```
Java
Python
Do you want to display content of another file [Yes|No]:y
Enter File Name to Display content: a.txt
```

The content of a.txt:

```
Tue Dec 10 17:00:48 IST 2019
Do you want to display content of another file [Yes|No]:YES
Enter File Name to Display content: b.txt
```

The content of b.txt:

```
December 2019
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 31
```

```
Do you want to display content of another file [Yes|No]:Y
Enter File Name to Display content: durga.txt
durga.txt does not exist
Do you want to display content of another file [Yes|No]:N
Thanks for using application
```



Write a script that reads a string as input and find its reverse?

Write a script that performs reverse of given string?

```
#!/bin/bash
```

```
read -p "Enter any string to find reverse: " str
```

```
len=$( echo -n $str | wc -c )  
output=""
```

```
while [ $len -gt 0 ]  
do  
    ch=$( echo -n $str | cut -c $len )  
    output=$output$ch  
    let len--  
done  
echo "The Original String: $str"  
echo "The Reversed String: $output"
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh  
Enter any string to find reverse: durga  
The Original String: durga  
The Reversed String: agrud  
durgasoft@durgasoft:~/scripts$ test.sh  
Enter any string to find reverse: abc  
The Original String: abc  
The Reversed String: cba  
durgasoft@durgasoft:~/scripts$ test.sh  
Enter any string to find reverse: durgasoft  
The Original String: durgasoft  
The Reversed String: tfoSagrud  
durgasoft@durgasoft:~/scripts$
```

Eg:

```
#!/bin/bash
```

```
while [ true ]  
do  
    clear  
    tput cup 7 25  
    echo "WELCOME TO DURGASOFT"  
    sleep 2  
    clear  
    tput cup 7 25
```



```
echo "UNIX/LINUX CLASSES"  
sleep 2  
done
```

Note: tput cup 7 25

It moves the cursor position to 7th row and 25th column.

Q. Write a script to read employee data and insert into emp.txt file?

#!/bin/bash

```
while [ true ]  
do  
    read -p "Employee Number: " eno  
    read -p "Employee Name: " ename  
    read -p "Employee Salary: " esal  
    read -p "Employee Address: " eaddr  
    echo "$eno:$ename:$esal:$eaddr" >> emp.txt  
    echo "Employee Record Inserted Successfully"  
    read -p "Do you want to insert one more record [Yes|No]: " option  
    case $option in  
        [yY]|[Yy][eE][sS])  
            continue  
            ;;  
        [nN]|[nN][oO])  
            break  
            ;;  
    esac  
done  
echo "Open emp.txt to see all employees information"
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh  
Employee Number: 100  
Employee Name: Sunny  
Employee Salary: 1000  
Employee Address: Hyderabad  
Employee Record Inserted Successfully  
Do you want to insert one more record [Yes|No]: y  
Employee Number: 200  
Employee Name: Bunny  
Employee Salary: 2000  
Employee Address: Mumbai  
Employee Record Inserted Successfully  
Do you want to insert one more record [Yes|No]: Yes  
Employee Number: 300
```



Employee Name: Chinny
Employee Salary: 3000
Employee Address: Hyderabad
Employee Record Inserted Successfully
Do you want to insert one more record [Yes | No]: YES
Employee Number: 400
Employee Name: Vinny
Employee Salary: 4000
Employee Address: Chennai
Employee Record Inserted Successfully
Do you want to insert one more record [Yes | No]: N
Open emp.txt to see all employees information

Q. Write a Script to implement cat command with -n option?

```
durgasoft@durgasoft:~/scripts$ cat -n emp.txt
```

```
1 100:Sunny:1000:Hyderabad
2 200:Bunny:2000:Mumbai
3 300:Chinny:3000:Hyderabad
4 400:Vinny:4000:Chennai
```

Syntax -1: To read data from the file by using while loop

```
cat emp.txt |
while read line
do
    do something with that line
done
```

Syntax -2: To read data from the file by using while loop

```
while read line
do
    do something with that line
done < emp.txt
```

test.sh

```
#!/bin/bash
```

```
fname=$1
if [ ! -f $fname ]; then
    echo "Please provide already existing regular file only"
    exit 1
fi
```

```
count=1
while read line
```



```
do
    echo " $count $line"
    let count++
done < $fname
```

Output

```
durgasoft@durgasoft:~/scripts$ test.sh emp.txt
```

```
1 100:Sunny:1000:Hyderabad
2 200:Bunny:2000:Mumbai
3 300:Chinny:3000:Hyderabad
4 400:Vinny:4000:Chennai
```

```
durgasoft@durgasoft:~/scripts$ test.sh abcd.txt
Please provide already existing regular file only
```

2) until Loop:

It is opposite to while loop.

Syntax:

```
until [ condition ]
do
    body
done
```

The body will be executed as long as condition returns false. Once condition returns true, then loop will be terminated.

Q. Write a script to print 1 to 5 by using until loop?

```
#!/bin/bash
```

```
i=1
```

```
until [ $i -gt 5 ]
do
    echo $i
    let i++
done
```

```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
1
2
3
4
5
```



3) for Loop:

If we want to perform some action for every item in the given list, then we should go for for loop. It is similar to Java's for-each loop.

Syntax:

```
for variable in item1 item2 item3... itemN
do
    action
done
```

Eg 1: To print numbers from 1 to 5.

```
#!/bin/bash
```

```
for i in 1 2 3 4 5
do
    echo "Current Number: $i"
done
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
Current Number: 1
Current Number: 2
Current Number: 3
Current Number: 4
Current Number: 5
```

Eg 2:

```
#!/bin/bash
```

```
for course in java unix python testing datascience
do
    echo "Course Name: $course"
done
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
Course Name: java
Course Name: unix
Course Name: python
Course Name: testing
Course Name: datascience
```




Q. Write a script that display numbers from 1 to 100, which are divisible by 10.

```
#!/bin/bash
```

```
count=0
for num in {1..100}
do
    if [ $( num%10 ) -eq 0 ]; then
        echo "$num"
        let count++
    fi
done
echo "The number of values: $count"
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
```

```
10
20
30
40
50
60
70
80
90
100
```

```
The number of values: 10
```

Q. Write a script to display all file names present in current working directory?

```
#!/bin/bash
```

```
for name in *
do
    if [ -f $name ]; then
        echo $name
    fi
done
```



Q. Write a script to append current date and time, current month calander in every .txt file present in current working directory?

```
#!/bin/bash
```

```
for fname in *.txt
do
    date >> $fname
    cal >> $fname
done
echo "Task Completed"
```

Q. Write a script that print all command line arguments?

```
#!/bin/bash
```

```
if [ $# -ne 0 ]; then

    count=1
    for arg in $@
    do
        echo "Command Line Argument - $count: $arg"
        let count++
    done
else
    echo "Command line arguments are not passed"
fi
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh
Command line arguments are not passed
durgasoft@durgasoft:~/scripts$ test.sh 10 20 30 40
Command Line Argument - 1: 10
Command Line Argument - 2: 20
Command Line Argument - 3: 30
Command Line Argument - 4: 40
durgasoft@durgasoft:~/scripts$ test.sh learning linux is very easy
Command Line Argument - 1: learning
Command Line Argument - 2: linux
Command Line Argument - 3: is
Command Line Argument - 4: very
Command Line Argument - 5: easy
```



Case Study:

emp.txt:

```
100:sunny:1000:Hyderabad
200:bunny:2000:Chennai
300:chinny:3000:Hyderabad
400:vinny:4000:Delhi
500:pinny:5000:Hyderabad
```

1. Write a script to display all employees information where salary is greater than 2500.

```
#!/bin/bash
```

```
for record in $(cat emp.txt)
do
    sal=$(echo $record | cut -d ":" -f 3)
    if [ $sal -gt 2500 ]; then
        echo $record
    fi
done
```

2. Write a script to save all employees information where salary is greater than 2500 and city is Hyderabad to hyd.txt.

```
#!/bin/bash
```

```
for record in $(cat emp.txt)
do
    sal=$(echo $record | cut -d ":" -f 3)
    city=$(echo $record | cut -d ":" -f 4)
    if [ $sal -gt 2500 -a $city = "Hyderabad" ]; then
        echo $record >> hyd.txt
    fi
done
echo "Task Completed"
```

3. Write a script to display minimum and maximum salaries ?

```
#!/bin/bash
max=$(cat emp.txt | head -1 | cut -d ":" -f 3)
min=$(cat emp.txt | head -1 | cut -d ":" -f 3)
max_record=$(cat emp.txt | head -1)
min_record=$(cat emp.txt | head -1)
for record in $(cat emp.txt)
do
```



```
sal=$(echo $record | cut -d ":" -f 3)
if [ $sal -gt $max ]; then
    max=$sal
    max_record=$record
fi
if [ $sal -lt $min ]; then
    min=$sal
    min_record=$record
fi
done
echo "The Maximum Salary:$max"
echo "The Maximum Salaried Employeeed Information:"
echo "Employee No:$(echo $max_record | cut -d ":" -f 1)"
echo "Employee Name:$(echo $max_record | cut -d ":" -f 2)"
echo "Employee Salary:$(echo $max_record | cut -d ":" -f 3)"
echo "Employee Address:$(echo $max_record | cut -d ":" -f 4)"
echo
echo "The Minimum Salary:$min"
echo "The Minimum Salaried Employeeed Information:"
echo "Employee No:$(echo $min_record | cut -d ":" -f 1)"
echo "Employee Name:$(echo $min_record | cut -d ":" -f 2)"
echo "Employee Salary:$(echo $min_record | cut -d ":" -f 3)"
echo "Employee Address:$(echo $min_record | cut -d ":" -f 4)"
```

Q. Write a script to display multiple files content to the terminal and all filenames passed as command line arguments?

```
#!/bin/bash
```

```
if [ $# -ne 0 ]; then
    for fname in $@
    do
        if [ -f $fname ]; then
            echo "$fname content:"
            echo "===== "
            cat $fname
        else
            echo "$fname does not exist or it is not a regular file"
        fi
    done
else
    echo "Please pass atleast one file name"
fi
```



Q. Write a script to append multiple files content to a single file result.txt. File names are passed as command line arguments?

```
#!/bin/bash
```

```
if [ $# -ne 0 ]; then
    for fname in $@
    do
        if [ -f $fname ]; then
            cat $fname >> result.txt
        else
            echo "$fname does not exist or it is not a regular file"
        fi
    done
else
    echo "Please pass atleast one file name"
fi
```



Topic-32: Arrays

If we want to represent a group of values with a single name then we should go for arrays concept.

How to Create Arrays:

1. If we know elements at the beginning:

```
courses=(Java Python Linux Django)
```

2. If we don't know elements at the beginning:

```
courses[0]=Java
```

```
courses[1]=Python
```

```
courses[2]=Linux
```

```
courses[3]=Django
```

The index values need not be consecutive. We can take randomly.

```
courses[10]=DataScience
```

```
courses[20]=Devops
```

How to Access Elements:

We can access array elements by using index which is zero based. i.e the index of first element is zero.

```
${courses[0]} → First element
```

```
${courses[1]} → Second element
```

```
${courses[@]} → All elements present inside array.
```

```
${courses[*]} → All elements present inside array into a single string separated by first character in IFS (Internal Field Separator)
```

```
${!courses[@]} → It returns all indexes where elements are available.
```

```
${#courses[@]} → It returns the number of elements present inside array.
```

```
${#courses[0]} → It returns the length of first element.
```

Demo Script:

```
#!/bin/bash
```

```
courses[0]=Java
```

```
courses[1]=Python
```

```
courses[2]=Linux
```

```
courses[3]=Django
```

```
courses[10]=DataScience
```

```
courses[20]=Devops
```



```
echo "First Element : ${courses[0]}"
echo "Second Element : ${courses[1]}"
echo "All Elements with @ : ${courses[@]}"
echo "All elements with * : ${courses[*]}"
echo "The indices where elements are available : ${!courses[@]}"
echo "The total number of elements : ${#courses[@]}"
echo "The length of first element : ${#courses[0]}"
```

Output:

durgasoft@durgasoft:~/scripts\$ test.sh

First Element : Java

Second Element : Python

All Elements with @ : Java Python Linux Django DataScience Devops

All elements with * : Java Python Linux Django DataScience Devops

The indices where elements are available : 0 1 2 3 10 20

The total number of elements : 6

The length of first element : 4

Demo Script for Arrays:

```
#!/bin/bash
```

```
files=( $(ls *.txt) )
echo -e "FILENAME\tREADABLE\tWRITEABLE\tEXECUTABLE"
echo "-----"
for fname in "${files[@]}"
do
    echo -n $fname
    if [ -r $fname ]; then
        echo -ne "\tYES"
    else
        echo -ne "\tNO"
    fi
    if [ -w $fname ]; then
        echo -ne "\tYES"
    else
        echo -ne "\tNO"
    fi
    if [ -x $fname ]; then
        echo -ne "\tYES"
    else
        echo -ne "\tNO"
    fi
    echo
done
```



Output:

<u>FILENAME</u>	<u>READABLE</u>	<u>WRITEABLE</u>	<u>EXECUTABLE</u>
abc.txt	YES	YES	YES
a.txt	YES	YES	NO
b.txt	YES	YES	NO
emp.txt	YES	YES	NO
file2.txt	YES	YES	NO
file3.txt	YES	YES	NO
hyd.txt	YES	YES	NO
input.txt	YES	YES	NO
output.txt	YES	YES	NO
result.txt	YES	YES	NO
sorted.txt	YES	YES	NO
x.txt	YES	YES	NO
xyz.txt	YES	YES	NO



Topic-33: Shell Script Functions

If any group of commands are repeatedly required, then it is not recommended to write these commands separately everytime. It increases length of the code and reduces readability.

Such type of repeated code we have to define inside a block and we can call that block where ever required. This block of commands is nothing but function.

Hence function is nothing but a block of repeatable commands.

Advantages of Functions:

- 1) It reduces length of the code.
- 2) It improves readability.
- 3) It improves maintainability.
- 4) It promotes DRY principle.
DRY → Don't Repeat Yourself.

How to define a Function:

1st Way:

```
function function_name()  
{  
    commands  
}
```

2nd Way:

```
function_name()  
{  
    commands  
}
```

How to call a Function:

```
function_name param1 param2 param3 ...
```



Q. Write a function to display wish message?

#!/bin/bash

#defining a function

wish()

{

 echo "Hello Friends... Good Evening"

}

wish # calling a function

wish

wish

Output:

durgasoft@durgasoft:~/scripts\$ test.sh

Hello Friends... Good Evening

Hello Friends... Good Evening

Hello Friends... Good Evening

Note: Before calling a function, it should be defined.

Eg 2:

#!/bin/bash

f1()

{

 echo "I am in f1 function"

}

f2()

{

 echo "I am in f2 function"

 f1

}

f1

f2

Output:

durgasoft@durgasoft:~/scripts\$ test.sh

I am in f1 function

I am in f2 function

I am in f1 function



Function with Parameters:

Function can accept parameters also. Within the function we can access parameters as follows:

\$1 → First Parameter

\$2 → Second Parameter

\$@ → All Parameters

\$* → All parameters

\$# → Total number of parameters

\$0 → It is script name but not function name

Eg:

```
#!/bin/bash
```

```
wish()  
{  
    if [ $# -eq 0 ]; then  
        echo "Hello Guest Good Evening"  
    else  
        echo "Hello $1 Good Evening"  
    fi  
}  
wish  
wish Durga  
wish Sunny
```

Output:

```
durgasoft@durgasoft:~/scripts$ test.sh  
Hello Guest Good Evening  
Hello Durga Good Evening  
Hello Sunny Good Evening
```

Q. Write a function that takes two integer numbers as parameters and perform arithmetic operations

```
#!/bin/bash  
calc()  
{  
    if [ $# -ne 2 ]; then  
        echo "You should pass exactly 2 arguments"  
    else  
        a=$1  
        b=$2  
        echo "$a+$b = $((a+b))"  
        echo "$a-$b = $((a-b))"  
        echo "$a*$b = $((a*b))"
```



```
echo "$a/$b = $((a/b))"
echo
fi
}
calc 10
calc 20 10
calc 200 100
calc 2000 1000
```

Output:

You should pass exactly 2 arguments

20+10 = 30

20-10 = 10

20*10 = 200

20/10 = 2

200+100 = 300

200-100 = 100

200*100 = 20000

200/100 = 2

2000+1000 = 3000

2000-1000 = 1000

2000*1000 = 2000000

2000/1000 = 2

Q. Write a function to print all parameters?

```
#!/bin/bash
parameter_printing()
{
    if [ $# -eq 0 ]; then
        echo "No parameters passed to this function"
    else
        echo "All Passed Parameters are:"
        echo "....."
        for p in $@
        do
            echo $p
        done
    fi
}
parameter_printing
parameter_printing A B C D E
```



Output:

No parameters passed to this function

All Passed Parameters are:

A
B
C
D
E

Q. Write a function to find maximum of two given integer numbers?

```
#!/bin/bash
max()
{
    if [ $1 -gt $2 ]; then
        echo "The Maximum of $1 and $2 :$1"
    else
        echo "The Maximum of $1 and $2: $2"
    fi
}
max 10 20
max 200 100
```

Output:

The Maximum of 10 and 20: 20
The Maximum of 200 and 100 :200

Q. Write a function to find maximum of three given integer numbers?**Q. Write a function to compare the given 2 integers?**

```
#!/bin/bash
compare()
{
    if [ $1 -eq $2 ]; then
        echo "Both Numbers are equal"
    elif [ $1 -gt $2 ]; then
        echo "$1 is greater than $2"
    else
        echo "$1 is less than $2"
    fi
}
compare 10 10
compare 10 20
compare 200 100
```



Output:

Both Numbers are equal
10 is less than 20
200 is greater than 100

Q. Write a function to find factorial of given integer number?

```
#!/bin/bash
factorial()
{
    original=$n
    fact=1
    while [ $n -gt 1 ]
    do
        let fact=fact*n
        let n--
    done
    echo "The Factorial of $original is: $fact"
}
read -p "Enter a number to find factorial:" n
factorial $n
}
```

```
#!/bin/bash
factorial()
{
    original=$1
    x=$1
    fact=1
    while [ $x -gt 1 ]
    do
        let fact=fact*x
        let x--
    done
    echo "The Factorial of $original is: $fact"
}
read -p "Enter a number to find factorial:" n
factorial $n
}
```



Variable Scope:

By default every variable in shell script is global. i.e we can access anywhere in our script. But before using a variable, it should be declared already.

Eg 1:

```
#!/bin/bash
f1()
{
    echo "x value : $x"
}
x=10
f1
```

Output: x value : 10

Eg 2:

```
#!/bin/bash
f1()
{
    x=20
    echo "x value : $x"
}
x=10
f1
echo "After f1 execution x value: $x"
```

Output:

x value : 20
After f1 execution x value: 20

Eg 3:

```
#!/bin/bash
f1()
{
    echo "x value : $x"
}
f1
x=10
f1
```

Output:

x value :
x value : 10



The variables which are declared inside a function, can be accessed outside of that function, because every variable has global scope by default.

Eg 4:

```
#!/bin/bash
f1()
{
    x=10
}
f1
echo "x value : $x"
```

Output: x value : 10

If we want a variable only within the function and should not be available outside of that function, then we have to use local keyword for the variable.
local variables can be accessed only inside function and cannot be accessed outside of that function.

Eg:

```
#!/bin/bash
f1()
{
    local x=10
    echo "Inside function x value: $x"
}
f1
echo "outside function x value : $x"
```

Ouput:

Inside function x value: 10
outside function x value :

Return Statement in Functions:

Every function in shell scripting returns some value. The default returned value is the exit code of the last command inside function.

But based on our requirement we can return values explicitly by using return statement.

return <exitcode>

The allowed values for exitcode are 0 to 255.

0 means successful

non-zero means unsuccessful.

We can get return value of function by using \$?.



Eg:

```
#!/bin/bash
sum()
{
    if [ $# != 2 ]; then
        echo "You should pass exactly two numbers"
        return 1
    else
        echo "The SUM:$(($1+$2))"
    fi
}
sum 10 20
echo "The Return value of this function:$?"
echo
sum 10
echo "The Return value of this function:$?"
```

```
durgasoft@durgasoft:~/scripts$ test.sh
The SUM:30
The Return value of this function:0
```

```
You should pass exactly two numbers
The Return value of this function:1
```

Use Case:

```
backup()
{
    commands to take backup
}
backup
if [ $? != 0 ]; then
    something goes wrong backup failed
else
    backup successful
fi
```



Topic-34: Project on Shell Programming

```
# user management application
add_user()
{
    echo "Provide details to add user:"
    read -p "First Name:" fname
    read -p "Last Name:" lname
    read -p "User Id:" uid
    read -s -p "Password:" pwd
    echo
    read -p "Retype Password:" cpwd
    if [ $pwd != $cpwd ]; then
        echo "Password not matching"
        return 1
    fi
    read -p "Zip Code:" zipcode
    if [ ! -e ~/users.dat ];then
        touch ~/users.dat
    fi
    count=$(grep $uid ~/users.dat | wc -l)
    if [ $count -ne 0 ]; then
        echo "User Id: $uid already exists"
        return 2
    fi
    echo "$uid:$pwd:$fname:$lname:$zipcode" >> ~/users.dat
    echo "User Added Successfully"
    echo
    echo
}
search_user()
{
    read -p "Enter User Id: " uid
    read -p "Enter Password: " pwd
    count=$(grep $uid ~/users.dat | wc -l)
    if [ $count -eq 0 ]; then
        echo "User Id: $uid Does not exist"
        return 3
    fi
    count=$(grep $uid ~/users.dat | grep $pwd | wc -l)
    if [ $count -eq 0 ]; then
        echo "Invalid Password"
        return 4
    fi
}
```



```
echo "The complete information of the user:"
record=$(grep $uid ~/users.dat)
echo "User Id: $(echo $record | cut -d ':' -f 1)"
echo "Password: $(echo $record | cut -d ':' -f 2)"
echo "First Name: $(echo $record | cut -d ':' -f 3)"
echo "Last Name: $(echo $record | cut -d ':' -f 4)"
echo "Zip Code: $(echo $record | cut -d ':' -f 5)"
echo
echo
}
change_password()
{
    read -p "Enter User Id: " uid
    read -p "Enter Password: " pwd
    count=$(grep $uid ~/users.dat | wc -l)
    if [ $count -eq 0 ]; then
        echo "User Id: $uid Does not exist"
        return 3
    fi
    count=$(grep $uid ~/users.dat | grep $pwd | wc -l)
    if [ $count -eq 0 ]; then
        echo "Invalid Password"
        return 4
    fi
    record=$(grep $uid ~/users.dat)
    grep -v $uid ~/users.dat > temp.dat
    #rm ~/users.dat
    mv temp.dat ~/users.dat

    read -p "Enter New Password:" npwd
    uid=$(echo $record | cut -d ':' -f 1)
    fname=$(echo $record | cut -d ':' -f 3)
    lname=$(echo $record | cut -d ':' -f 4)
    zipcode=$(echo $record | cut -d ':' -f 5)
    echo "$uid:$npwd:$fname:$lname:$zipcode" >> ~/users.dat
    echo "Password updated successfully"
    echo
    echo
}
delete_user()
{
    read -p "Enter User Id: " uid
    read -p "Enter Password: " pwd
    count=$(grep $uid ~/users.dat | wc -l)
    if [ $count -eq 0 ]; then
```



```
    echo "User Id: $uid Does not exist"
    return 3
fi
count=$(grep $uid ~/users.dat | grep $pwd | wc -l)
if [ $count -eq 0 ]; then
    echo "Invalid Password"
    return 4
fi
grep -v $uid ~/users.dat > temp.dat
#rm ~/users.dat
mv temp.dat ~/users.dat
echo "User Deleted Successfully"
echo
echo
}
show_all_users()
{
    echo "All users Information:"
    echo "....."
    cat < ~/users.dat
    echo
    echo
}
users_count()
{
    count=$(cat ~/users.dat | wc -l)
    echo "The Total Number of Users: $count"
    echo
    echo
}
echo "Welcome to User Management Application"
while [ true ]
do
    echo "1. Add User"
    echo "2. Search User"
    echo "3. Change Password"
    echo "4. Delete User"
    echo "5. Show all Users"
    echo "6. Users count"
    echo "7. Exit"
    read -p "Enter Your Choice [1/2/3/4/5/6/7]: " choice
    case $choice in
        1)
            add_user
            ;;
    esac
done
```



```
2)      search_user
        ;;
3)      change_password
        ;;
4)      delete_user
        ;;
5)      show_all_users
        ;;
6)      users_count
        ;;
7)      echo "Thanks for using Application"
        exit 0
        ;;
*)      echo "Wrong choice... Try again"

esac

done
```



Topic-35: Stream Editor (SED)

We can use sed command to retrieve and edit data present in the given file.

We can perform operations based on line like delete 2nd line etc.

We can also perform operations based on context like delete lines where python present etc

emp.dat

```
eno|ename|esal|eaddr|dept|gender
100|sunny|1000|mumbai|admin|female
200|bunny|2000|chennai|sales|male
300|chinny|3000|delhi|accounting|female
400|vinny|4000|hyderabad|admin|male
500|pinny|5000|mumbai|sales|female
```

1) Display Specific Line multiple times:

```
$ sed '2p' emp.dat
```

p means print

It will display total file content but 2nd line will be displayed 2 times.

```
durgasoft@durgasoft:~/Desktop$ sed '2p' emp.dat
```

```
eno|ename|esal|eaddr|dept|gender
100|sunny|1000|mumbai|admin|female
100|sunny|1000|mumbai|admin|female
200|bunny|2000|chennai|sales|male
300|chinny|3000|delhi|accounting|female
400|vinny|4000|hyderabad|admin|male
500|pinny|5000|mumbai|sales|female
```

2) Display only specific line:

```
$ sed -n '3p' emp.dat
```

It will display only 3rd line

3) Display last line:

```
$ sed -n '$p' emp.dat
```

\$ means last line

4) Display multiple lines in range:

```
$ sed -n '2,4p' emp.dat
```

it will display from 2nd to 4th lines.



5) Display all lines except specific line:

```
$ sed -n '2p' emp.dat
```

It will display only 2nd line

```
$ sed -n '2!p' emp.dat
```

It will display all lines except 2nd line

6) Display all lines except range of lines:

```
$ sed -n '2,4p' emp.dat
```

It will display all lines from 2nd to 4th.

```
$ sed -n '2,4!p' emp.dat
```

It will display all lines except 2 to 4.

Note: We can use cut command to read data column wise, where as we can use sed command to read row wise data.

7) Display all lines having specific word:

```
$ sed -n '/admin/p' emp.dat
```

It will display all lines which contains admin.

8) deleting data present in the file:

```
$ sed '3d' emp.dat
```

d means deletion

While displaying records, it will delete 3rd record. But this line won't be deleted in the file.

```
$ sed '$d' emp.dat
```

While displaying records, it will delete last record. But this line won't be deleted in the file.

```
$ sed '1d' emp.dat
```

While displaying records, it will delete first record. But this line won't be deleted in the file.

```
$ sed -i '5d' emp.dat
```

-i meant for deleting record permanently in the file.

This command won't display anything to the console but 5th record in the file will be deleted.

```
$ sed -i '1,$d' emp.dat
```

It will delete all records from 1st to last in the file. Now file will become empty. (To delete range of records)



```
$ sed -i '1d  
> $d' emp.dat
```

After 1d we are using enter key. It will delete first and last records.

Replacing Data present in the File:

We have to use s option. s means substitute.

```
durgasoft@durgasoft:~/Desktop$ cat > demo.txt  
durga  soft  durga solutions  
durga  tech  durga soft  
durga  jobs  training
```

1. \$sed 's/durga/linux/g' demo.txt

It will replace all occurrences of durga with linux while displaying the content. It won't modify file content.

2. \$sed 's/durga/linux/' demo.txt

It will replace only first occurrence in every line of durga with linux while displaying the content. It won't modify file content.

g option → g means global means every occurrence
without g → only first occurrence in every line.

```
durgasoft@durgasoft:~/Desktop$ sed 's/durga/linux/g' demo.txt  
linux  soft  linux solutions  
linux  tech  linux soft  
linux  jobs  training  
durgasoft@durgasoft:~/Desktop$ cat demo.txt  
durga  soft  durga solutions  
durga  tech  durga soft  
durga  jobs  training  
durgasoft@durgasoft:~/Desktop$ sed 's/durga/linux/' demo.txt  
linux  soft  durga solutions  
linux  tech  durga soft  
linux  jobs  training
```

4. \$ sed 's/DURGA/linux/gi' demo.txt

While performing replacement to ignore case.

5. \$ sed -i 's/DURGA/linux/gi' demo.txt

To replace permanently in the file. This command won't display anything to the console.

6. \$ sed -i 's/^\$/I Like Sunny/g' demo.txt

It replaces every blank line with 'I Like Sunny'



7. `$ sed -i 's/\<sunny\>//gi' demo.txt`

It deletes every occurrence of sunny in the file. Here case won't be considered.

8. `$ sed -i 's/\<[0-9][0-9][0-9]\>//g' demo.txt`

It will delete all 3-digit words.



Topic-36: AWK Programming

We can use awk commands to search in the files and print results in our required format. AWK provides more convenient way for file processing than our regular linux commands. Most commonly used in shell scripting if we want to handle very large files.

Structure of awk Command:

`awk {BEGIN BLOCK}{ACTION BLOCK}{END BLOCK} filenames`

BEGIN block will be executed only once before processing the file.

ACTION block will be executed for every line/record present in the file.

END block will be executed only once after completing all lines/records processing.

Eg: input.txt:

Any subject will provide huge benefit.

The useless waste activities we should not do.

Atleast one new activity we should do per day.

Don't Play games with Money.

```
durgasoft@durgasoft:~/scripts$ awk 'BEGIN{print "BEGIN"}{print "Action"}END{print "END"}' input.txt
```

BEGIN

Action

Action

Action

Action

END

Eg: awk command to print file data and the number of lines:

```
awk 'BEGIN{print "Beginning of the File";print "-----";c=0;}{print $0;c=c+1;}END{print ".....";print "The Number Of Lines:" c}' input.txt
```

awk

'BEGIN

{

 print "Beginning of the File";

 print "-----";

 c=0;

}

{

 print \$0;

 c=c+1;



```
}  
END  
{  
    print ".....";  
    print "The Number Of Lines: " c  
}' input.txt
```

```
durgasoft@durgasoft:~/scripts$ awk 'BEGIN{print "Beginning of the File";print"-----  
---";c=0;}{print $0;c=c+1;}END{print ".....";print "The Number Of Lines:" c}'  
input.txt
```

Beginning of the File

Any subject will provide huge benefit.
The useless waste activities we should not do.
Atleast one new activity we should do per day.
Don't Play games with Money.

.....
The Number Of Lines:4

```
durgasoft@durgasoft:~/scripts$ awk 'BEGIN{print "Beginning of the File";print"-----  
---";c=0;}{c=c+1;print c": "$0;}END{print ".....";print "The Number Of Lines:"  
c}' input.txt
```

```
durgasoft@durgasoft:~/scripts$ awk 'BEGIN{print "Beginning of the File";print"-----  
---";c=0;}{c=c+1;print c": "$0;}END{print ".....";print "The Number Of Lines:"  
c}' input.txt
```

Beginning of the File

1: Any subject will provide huge benefit.
2: The useless waste activities we should not do.
3: Atleast one new activity we should do per day.
4: Don't Play games with Money.

.....
The Number Of Lines:4

Note:

1. Normal Bash commands like echo won't work in AWK.
2. \$0 represents total line/record.



Processing Tabular Data by using awk:

While processing tabular data

\$0 → Represents Each Line/Record

\$1 → Represents First Field in the current line/record

\$2 → Represents Second Field in the current line/record

\$3 → Represents Third Field in the current line/record

etc

emp.txt:

eno	ename	esal	eaddr
100	Sunny	1000	Mumbai
200	Bunny	2000	Hyderabad
300	Chinny	3000	Hyderabad
400	Vinny	4000	Chennai
500	Pinny	5000	Mumbai

Note: In awk, default separator tab.

1. To Print only 1st Column Data:

\$ awk '{print \$1}' emp.txt

```
durgasoft@durgasoft:~/scripts$ awk '{print $1}' emp.txt
```

```
eno
100
200
300
400
500
```

2. To Print 1st and 3rd Columns Data:

```
durgasoft@durgasoft:~/scripts$ awk '{print $1 $3}' emp.txt
```

```
eno esal
1001000
2002000
3003000
4004000
5005000
```

```
durgasoft@durgasoft:~/scripts$ awk '{print $1 " " $3}' emp.txt
```

```
eno esal
100 1000
200 2000
300 3000
```



```
400 4000
500 5000
durgasoft@durgasoft:~/scripts$ awk '{print $1 "→" $3}' emp.txt
eno → esal
100 → 1000
200 → 2000
300 → 3000
400 → 4000
500 → 5000
```

3. To Print all Columns Data:

```
durgasoft@durgasoft:~/scripts$ awk '{print $0}' emp.txt
eno    ename esal    eaddr
100    Sunny 1000   Mumbai
200    Bunny 2000   Hyderabad
300    Chinny 3000  Hyderabad
400    Vinny 4000   Chennai
500    Pinny 5000   Mumbai
```

Processing CSV File Data:

If, is the separator between the values, then such type of file is called csv file.

CSV → Comma Separated Values

By default tab is separator in awk. Instead of tab if we are using any separator then we should use either -F option or FS variable. FS is predefined variable in awk.

emp.csv:

```
eno,ename,esal,eaddr
100,Sunny,1000,Mumbai
200,Bunny,2000,Hyderabad
300,Chinny,3000,Hyderabad
400,Vinny,4000,Chennai
500,Pinny,5000,Mumbai
```



To Print 2nd and 4th Columns Data:

```
durgasoft@durgasoft:~/scripts$ awk '{print $2 " "$4}' emp.csv
```

```
durgasoft@durgasoft:~/scripts$ awk '{print $1}' emp.csv
```

```
eno,ename,esal,eaddr  
100,Sunny,1000,Mumbai  
200,Bunny,2000,Hyderabad  
300,Chinny,3000,Hyderabad  
400,Vinny,4000,Chennai  
500,Pinny,5000,Mumbai
```

```
$ awk -F "," '{print $2 " "$4}' emp.csv
```

```
durgasoft@durgasoft:~/scripts$ awk -F "," '{print $2 " "$4}' emp.csv
```

```
ename eaddr  
Sunny Mumbai  
Bunny Hyderabad  
Chinny Hyderabad  
Vinny Chennai  
Pinny Mumbai
```

```
$ awk '{print $2"→"$4}' FS="," emp.csv
```

```
durgasoft@durgasoft:~/scripts$ awk '{print $2"→"$4}' FS="," emp.csv
```

```
ename→eaddr  
Sunny →Mumbai  
Bunny → Hyderabad  
Chinny → Hyderabad  
Vinny → Chennai  
Pinny → Mumbai
```

Display Table Content without Header:

We can use AWK predefined variable NR, which means Row Number.

```
durgasoft@durgasoft:~/scripts$ awk '{print $0}' FS="," emp.csv
```

```
eno,ename,esal,eaddr  
100,Sunny,1000,Mumbai  
200,Bunny,2000,Hyderabad  
300,Chinny,3000,Hyderabad  
400,Vinny,4000,Chennai  
500,Pinny,5000,Mumbai
```

```
durgasoft@durgasoft:~/scripts$ awk 'NR!=1{print $0}' FS="," emp.csv
```

```
100,Sunny,1000,Mumbai  
200,Bunny,2000,Hyderabad  
300,Chinny,3000,Hyderabad
```



400,Vinny,4000,Chennai
500,Pinny,5000,Mumbai

If Row Number is not equal to 1 then only print row.

```
durgasoft@durgasoft:~/scripts$ awk 'NR==1{print $0}' FS="," emp.csv
eno,ename,esal,eaddr
durgasoft@durgasoft:~/scripts$ awk 'NR>=2{print $0}' FS="," emp.csv
100,Sunny,1000,Mumbai
200,Bunny,2000,Hyderabad
300,Chinny,3000,Hyderabad
400,Vinny,4000,Chennai
500,Pinny,5000,Mumbai
durgasoft@durgasoft:~/scripts$ awk 'NR<3{print $0}' FS="," emp.csv
eno,ename,esal,eaddr
100,Sunny,1000,Mumbai
```

Note: With NR variable we can use all relational operators.
NR!=1, NR==1, NR<1, NR<=1, NR>1,NR>=1

Filter Data based on given Condition:

1. List out all Employees where eaddr is Hyderabad:

```
awk '$4=="Hyderabad"{print $0}' FS="," emp.csv
```

```
durgasoft@durgasoft:~/scripts$ awk '$4=="Hyderabad"{print $0}' FS="," emp.csv
200,Bunny,2000,Hyderabad
300,Chinny,3000,Hyderabad
```

2. List out all employees where salary is greater than 2500.

```
awk '$3>2500{print $0}' FS="," emp.csv
```

```
durgasoft@durgasoft:~/scripts$ awk '$3>2500{print $0}' FS="," emp.csv
eno,ename,esal,eaddr
300,Chinny,3000,Hyderabad
400,Vinny,4000,Chennai
500,Pinny,5000,Mumbai
```



Display Records based on given Search String:

We have to specify the search string as follows

/searchstring/

```
durgasoft@durgasoft:~/scripts$ awk '/20/{print $0}' FS="," emp.csv
200,Bunny,2000,Hyderabad
20,Durga,5000,Hyderabad
```

If search String needs to be Present in specific Column:

```
awk '/20/{print $0}' FS="," emp.csv
```

In total record inside any field if 20 is there, then display that record

```
awk '$3~/20/{print $0}' FS="," emp.csv
```

If 3rd field contains 20 then only display that record.

```
awk '$3!~/20/{print $0}' FS="," emp.csv
```

If 3rd field (column) does not contain 20, then only display that record.

```
durgasoft@durgasoft:~/scripts$ awk '/20/{print $0}' FS="," emp.csv
200,Bunny,2000,Hyderabad
20,Durga,5000,Hyderabad
50,Shiva,2020,Hyderabad
durgasoft@durgasoft:~/scripts$ awk '$3~/20/{print $0}' FS="," emp.csv
200,Bunny,2000,Hyderabad
50,Shiva,2020,Hyderabad
durgasoft@durgasoft:~/scripts$ awk '$3!~/20/{print $0}' FS=","
emp.csveno,ename,esal,eaddr
100,Sunny,1000,Mumbai
300,Chinny,3000,Hyderabad
400,Vinny,4000,Chennai
500,Pinny,5000,Mumbai
20,Durga,5000,Hyderabad
```

If Total Field Value needs to be matched:

```
awk '/2000/{print $0}' FS="," emp.csv
```

```
awk '$3~/2000/{print $0}' FS="," emp.csv
```

```
awk '$3==2000{print $0}' FS="," emp.csv
```

If third field value exactly 2000, then only record will be printed.

```
awk '$4=="Hyderabad"{print $0}' FS="," emp.csv
```

If 4th field value exactly Hyderabad, then only record will be printed.



Increment Salary by 500 where Salary is Less than 3000:

```
awk -F "," ' {if($3<3000) print $0}' emp.csv
```

```
awk -F "," ' {if($3<3000) {$3=$3+500;print $0}}' emp.csv
```

```
durgasoft@durgasoft:~/scripts$ awk -F "," ' {if($3<3000) print $0}' emp.csv
```

```
100,Sunny,1000,Mumbai
```

```
200,Bunny,2000,Hyderabad
```

```
50,Shiva,2020,Hyderabad
```

```
durgasoft@durgasoft:~/scripts$ awk -F "," ' {if($3<3000) {$3=$3+500;print $0}}' emp.csv
```

```
100 Sunny 1500 Mumbai
```

```
200 Bunny 2500 Hyderabad
```

```
50 Shiva 2520 Hyderabad
```

Q. Find the Number of Employees whose Salary is Greater than 3000 by using awk?

```
awk -F "," ' BEGIN{c=0}NR!=1{if($3>3000){c=c+1;print $0;}}END{print "The Total Number of Employees where salary > 3000 :" c}' emp.csv
```

```
durgasoft@durgasoft:~/scripts$ awk -F "," ' BEGIN{c=0}NR!=1{if($3>3000){c=c+1;print $0;}}END{print "The Total Number of Employees where salary > 3000 :" c}' emp.csv
```

```
400,Vinny,4000,Chennai
```

```
500,Pinny,5000,Mumbai
```

```
20,Durga,5000,Hyderabad
```

```
2000,Pavan,20000,Hyderabad
```

```
The Total Number of Employees where salary > 3000 :4
```

Predefined Variables of awk:

FS → Field Separator

NR → Row Number / Number of Rows

NF → Number of Fields

RS → Record Separator

```
echo "Apple Banana Carrot" | awk 'END{print "The Number of Records:" NR;print "The Number of Fields:" NF}'
```

```
durgasoft@durgasoft:~/scripts$ echo "Apple Banana Carrot" | awk 'END{print "The Number of Records:" NR;print "The Number of Fields:" NF}'
```

```
The Number of Records:1
```

```
The Number of Fields:3
```

```
cat emp.txt | awk 'END{print "The Number of Records:" NR;print "The Number of Fields:" NF}'
```

```
awk 'END{print "The Number of Records:" NR;print "The Number of Fields:" NF}' emp.txt
```



Record Separator (RS):

The default Record Separator is new line (\n). If we want to use any other then we can specify by using RS variable.

demo.txt:

Apple,Banana:Cat,Dog:Java,Python

```
durgasoft@durgasoft:~/scripts$ awk -F "," 'BEGIN{RS=":"}{print $1"---->$2}' demo.txt
```

Apple---->Banana

Cat---->Dog

Java---->Python

if-else Syntax in AWK:

if(condition)

{

 commands

}

else

{

 commands

}

Eg:

```
durgasoft@durgasoft:~/scripts$ cat emp.txt
```

eno ename esal eaddr

100 Sunny 1000 Mumbai

200 Bunny 2000 Hyderabad

300 Chinny 3000 Hyderabad

400 Vinny 4000 Chennai

500 Pinny 5000 Mumbai

```
durgasoft@durgasoft:~/scripts$ awk 'NR!=1{if($3>3000){print $2" is very costly  
employee"}}else{print $2" is not costly employee"}}' emp.txt
```

Sunny is not costly employee

Bunny is not costly employee

Chinny is not costly employee

Vinny is very costly employee

Pinny is very costly employee



for Loop Syntax in AWK:

```
for(initialization;condition;increment)
{
    commands
}
```

```
awk 'BEGIN{for(i=1;i<=5;i++){print "The Iteration Number:" i}}'
```

```
durgasoft@durgasoft:~/scripts$ awk 'BEGIN{for(i=1;i<=5;i++){print "The Iteration Number:" i}}'
```

The Iteration Number:1

The Iteration Number:2

The Iteration Number:3

The Iteration Number:4

The Iteration Number:5

```
durgasoft@durgasoft:~/scripts$ echo "CAT DOG RAT TIGER" | awk '{for(i=1;i<=NF;i++){print "Field:" i ":" $i}}'
```

Field:1:CAT

Field:2:DOG

Field:3:RAT

Field:4:TIGER



Topic-37: Project on awk Programming

\$ test.sh

-l location | --location location
-e extension | --extension extension
-s | --stats
-h | --help

test.sh -l /etc

test.sh --location /etc

test.sh -l /etc -e txt

test.sh -e txt

Script to iterate Command Line Arguments:

```
#!/bin/bash
```

```
while [ $# != 0 ]
```

```
do
```

```
  case $1 in
```

```
    -l|--location )
```

```
      echo "-l|--location provided"
```

```
      LOCATION=$2
```

```
      echo "$LOCATION provided as argument"
```

```
      shift
```

```
      shift
```

```
      ;;
```

```
    -e|--extension )
```

```
      echo "-e|--extension provided"
```

```
      EXT=$2
```

```
      echo "$EXT provided as argument"
```

```
      shift
```

```
      shift
```

```
      ;;
```

```
    -s|--stats )
```

```
      echo "-s|--stats provided"
```

```
      shift
```

```
      ;;
```

```
    -h|--help )
```

```
      echo "-h|--help provided"
```

```
      shift
```

```
      ;;
```

```
    * )
```

```
      echo "Invalid option"
```



```
shift
esac
done

location
ls -l | awk '/^-/ ' | grep '\.txt' &> /dev/null
ls -l | awk '/^-/ ' | grep '\.pdf' &> /dev/null

ls -l $LOCATION | awk '/^-/ ' | grep '\.$EXT' &> /dev/null
ls -l /etc | awk '/^-/ ' | grep '\.conf' &> /dev/null
ls -l $LOCATION | awk '/^-/ ' | grep '\.$EXT' &> /dev/null

ls -l $LOCATION | awk '/^-/ ' | grep '\.$EXT' &> /dev/null
echo $?
if [ $? -ne 0 ]; then
    echo "No File with extension: $EXT found"
    exit 2
fi

echo "Location:$LOCATION"
echo "Extension:$EXT"
ls -l /etc | awk '/^-/ ' | grep '\.conf' &> /dev/null
echo $?
if [ $? -ne 0 ]; then
    echo "No File with extension: $EXT found"
    exit 2
fi

ls -l /etc | awk '/^-/ ' | grep '.conf' | awk 'BEGIN{sum=0}{sum=sum+$5}END{print "The
Number of Files:" NR;print "The Total size of all Files:" sum " Bytes";print "The Total size
of all Files:" sum/1024/1024 " MB"}'

ls -l $LOCATION | awk '/^-/ ' | grep ".$EXT" | awk 'BEGIN{sum=0}{sum=sum+$5}END{print
"The Number of Files:" NR;print "The Total size of all Files:" sum " Bytes";print "The Total
size of all Files:" sum/1024/1024 " MB"}'

ls -l /home/durgasoft/scripts | awk '/^-/ ' | grep ""

The Largest File is:
The Smallest File is:

ls -l /etc | awk '/^-/ ' | grep "conf" | awk '{
if(NR==1){
max=$5
max_file=$9
```



```
}  
if($5>max)  
{  
    max=$5  
    max_file=$9  
}  
}  
END  
{  
    print "The Largest File:" max_file "Its Size:" max;  
}'
```

10 40 20 35 65 78 3 90

```
if NR==1  
max=90
```

Complete Project on awk programming:

```
#!/bin/bash  
function help_function()  
{  
    echo "$0 will provide information based on location, extension including memory info"  
    echo "$0 [-l location] [--location location] [-e extension] [--extension extension] [-s|--  
stats] [-h|--help]"  
    echo "Examples:"  
    echo "$0 -l /etc"  
    echo "$0 -l /etc -e txt -s"  
    exit 1  
}  
LOCATION=$(pwd)  
EXT=""  
STATS=0  
while [ $# != 0 ]  
do  
    case $1 in  
        -l|--location )  
  
            LOCATION=$2  
            if [ ! -d $LOCATION ]; then  
                echo "You should provide valid directory as Location"  
                exit 2  
            fi  
            shift  
            shift  
            ;;  
    esac  
done
```



```
-e|--extension )
    EXT=$2
    shift
    shift
    ;;
-s|--stats )
    STATS=1
    shift
    ;;
-h|--help )
    help_function
    shift
    ;;
* )
    help_function
    shift
esac
done
echo -n "This script finding all files:"
if [ "$EXT" = "" ];then
    echo " With all extensions"
else
    echo "With $EXT extension"
fi
echo "In The Location:$LOCATION"

ls -l $LOCATION | awk '/^-/ ' | grep ".$EXT" &> /dev/null
if [ $? -ne 0 ]; then
    echo "No Files with extension: $EXT found"
    exit 2
fi
ls -l $LOCATION | awk '/^-/ ' | grep ".$EXT" | awk 'BEGIN{sum=0}{sum=sum+$5}END{print
"The Number of Files:" NR;print "The Total size of all Files:" sum " Bytes";print "The Total
size of all Files:" sum/1024/1024 " MB"}'

# This will be executed if you provide -s or --stats option
if [ $STATS -ne 0 ]; then
    echo
    echo "Statistics are:"
    ls -l $LOCATION | awk '/^-/ ' | grep ".$EXT" | awk
    '{if(NR==1){max=$5;min=$5;max_file=$9;min_file=$9;}if($5>max){max=$5;max_file=$9;}if(
$5<min){min=$5;min_file=$9;}}END{print "The Largest File:" max_file " Its Size:" max "
Bytes";print "The Smallest File:" min_file " Its Size:" min " Bytes"}'
fi
```



Part-3



Linux Administration Basics



Agenda

Topic-38: Job Scheduling with crontab

Topic-39: User Management

Topic-40: Working with putty, Mobaxterm and XShell

Topic-41: How to customize Open Source Software Code

Topic-42: Process Management

Topic-43: Communication Commands (write, wall, msg, mail)
How to use Gmail from the Ubuntu Terminal to send Emails

Topic-44: Package Management

Topic-45: Install and Working with MySQL Database

Topic-46: Working with Java

Topic-47: Job Control

Topic-48: nohup command

Topic-49: at command

Topic-50: Memory related commands

Topic-51: Networking commands

Topic-52: Working with winscp

Topic-53: Working with Filezilla

Topic-54: init command

Topic-55: TopMost important Linux Questions



Topic-38: Job scheduling with crontab

- * cron is derived from Greek and its meaning is TIME.
- * If any job executes automatically at the specified time intervals, such type of jobs are called cron jobs.
- * The cron job can be either a single command or a script.
- * We have to configure cron jobs in crontab.
- * Cron jobs are best suitable for automating work flows.

Various Important Commands:

\$ crontab -l

To list out cron jobs which are already configured.

\$ crontab -e

To edit already existing cron jobs and to define new cron jobs.

\$ crontab -r

To remove all configured cron jobs.

\$ crontab filename

Install a new crontab from file

Format of crontab:

m h dom mon dow command

m → Minutes. The allowed values range: 0 to 59.

h → Hours. The allowed values range: 0 to 23

dom → Day of Month. Allowed values range: 1 to 31

mon → Month of Year. Allowed values range is: 1 to 12

Even we can use english words like JAN,FEB,MAR etc

dow → Day of week. The allowed values range is: 0 to 6

Even we can use english words like SUN(0),MON(1),TUE(2)

Q. Configure cron job which writes hello message to hello.txt for every minute.

\$ crontab -e

```
* * * * * echo "Hello from cron" >> ~/Desktop/hello.txt
```



Q. Write a script that takes backup of our Desktop directory and place that backup copy inside backup directory present in user home directory? Configure this script as cron job, which should be executed every FRIDAY 23:59.

test.sh:

```
tar -cjf backup$(date +%d_%m_%Y_%H_%M).tar ~/Desktop/*  
mv backup$(date +%d_%m_%Y_%H_%M).tar ~/backup
```

Configure in crontab:

```
# m h dom mon dow command  
59 23 * * FRI bash ~/test.sh
```

Various Possibilities of specifying Date and Time:

```
# m h dom mon dow command
```

1. Every minute * * * * *
2. Every 2 minutes */2 * * * *
*/2 means every 2nd minute
3. Every even minute */2 * * * *
4. Every odd minute 1-59/2 * * * *
1-59 means range from 1 to 59
5. Every 3 minutes */3 * * * *
6. Every 4 minutes */4 * * * *
7. Every 15 minutes (Every quarter hour) */15 * * * *
8. Every hour at 30th minute 30 * * * *
9. Every half hour */30 * * * *
10. Every hour 0 * * * *
11. Every 2 hours 0 */2 * * *
12. Every even hours 0 */2 * * *
13. Every odd hours 0 1-23/2 * * *
14. Execute only at 6'o clock,14'o clock and 22'o clock
0 6,14,22 * * *
15. Execute every day/daily
0 0 * * *
16. Every Day at 1 AM
0 1 * * *
17. Every Night at midnight
0 0 * * *
18. Every Sunday only once
0 0 * * SUN
0 0 * * 0



19. Only on week days

`0 0 * * 1-5`

20. Only on week-ends

`0 0 * * 6,0`

21. Every month on 1st

`0 0 1 * *`

22. Every other month/ Every alternative month/ every 2 months once

`0 0 * */2 *`

23. Every 6 months once

`0 0 * */6 *`

24. Every Year Jan 1st

`0 0 1 1 *`

25. Every Leap Year

`0 0 29 2 *`

26. Every Feb 14th

`0 0 14 2 *`

Crontab Shortcuts:

@yearly `0 0 1 1 *`

@annually `0 0 1 1 *`

@monthly `0 0 1 * *`

@weekly `0 0 * * 0`

@daily `0 0 * * *`

@midnight `0 0 * * *`

@hourly `0 0-23 * * *`

Eg: @hourly echo "Hello from hourly crontab" >> ~/Desktop/hours.txt

Note: Crontab shortcuts may not be supported by most of the linux flavours.

Note: <https://crontab.guru/> can provide option to check our configurations.



Topic-39: User Management

- 1) How to add a new group?
- 2) How to add a new user?
- 3) Switching from one user to another user
- 4) How to get information about a particular user?
- 5) How to delete user?
- 6) How to delete group?
- 7) How to change ownership of a file?
- 8) How to change group membership of a file?
- 9) How to change group of a user?
- 10) Add a User to Multiple Groups

1) How to add a New Group?

We have to use `addgroup` command. For this `sudo` permission is required.

`sudo` means superuser do.

```
durga@durga-VirtualBox:~$ sudo addgroup pythongroup
[sudo] password for durga:
Adding group `pythongroup' (GID 1006) ...
Done.
```

Note: We can see all created groups information inside `/etc/group` file.

2) How to add User:

We have to use `adduser` command. For this `sudo` permission is required.

```
durga@durga-VirtualBox:~$ sudo adduser --ingroup pythongroup pyuser1
Adding user `pyuser1' ...
Adding new user `pyuser1' (1005) with group `pythongroup' ...
Creating home directory `/home/pyuser1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for pyuser1
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
```



Home Phone []:

Other []:

Is the information correct? [Y/n] y

Note: While creating new user, if we are not specifying groupname, then a new group will be created with the same name as username.

```
durga@durga-VirtualBox:~$ sudo adduser pyuser2
Adding user `pyuser2' ...
Adding new group `pyuser2' (1007) ...
Adding new user `pyuser2' (1006) with group `pyuser2' ...
Creating home directory `/home/pyuser2' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for pyuser2
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] y
```

Note: We can see all created users information inside /etc/passwd file.

```
durga@durga-VirtualBox:~$ cat /etc/passwd
...
pyuser1:x:1005:1006:,,,:/home/pyuser1:/bin/bash
pyuser2:x:1006:1007:,,,:/home/pyuser2:/bin/bash
```

3) Switching from one User to another User:

We have to use su command.

su means switch user.

Syntax: su newuser

```
durga@durga-VirtualBox:~$ whoami
durga
durga@durga-VirtualBox:~$ su pyuser1
Password:
pyuser1@durga-VirtualBox:/home/durga$ whoami
pyuser1
pyuser1@durga-VirtualBox:/home/durga$ exit
```



```
exit
durga@durga-VirtualBox:~$ whoami
durga
```

Note: If we use su command without any argument, then it will switch to root user. Hence the following 2 commands are equal. (But in ubuntu this option is not working)

```
$ su
$ su root
```

In Ubuntu, to switch to root user , we have to use:
durga@durga-VirtualBox:~\$ sudo -i

su with and without - option:

If we use su command without - option, then only user will be switched but environment won't be switched.

```
durga@durga-VirtualBox:~$ whoami
durga
durga@durga-VirtualBox:~$ pwd
/home/durga
durga@durga-VirtualBox:~$ export x=9999
durga@durga-VirtualBox:~$ echo $x
9999
durga@durga-VirtualBox:~$ su pyuser1
Password:
pyuser1@durga-VirtualBox:/home/durga$ whoami
pyuser1
pyuser1@durga-VirtualBox:/home/durga$ pwd
/home/durga
pyuser1@durga-VirtualBox:/home/durga$ echo $x
9999
```

If we use - option with su command, then both user and environment will be switched.

```
durga@durga-VirtualBox:~$ whoami
durga
durga@durga-VirtualBox:~$ pwd
/home/durga
durga@durga-VirtualBox:~$ export x=9999
durga@durga-VirtualBox:~$ echo $x
9999
durga@durga-VirtualBox:~$ su - pyuser1
Password:
```



```
pyuser1@durga-VirtualBox:~$ whoami
pyuser1
pyuser1@durga-VirtualBox:~$ pwd
/home/pyuser1
pyuser1@durga-VirtualBox:~$ echo $x
Here x is not available because x is environment variable of durga user.
```

Q. What is the difference between the following ?

\$ su pyuser1 → Only user will be switched but not environment.

\$ su - pyuser1 → Both user and environment will be switched.

4) How to get Information about a Particular User:

We have to use finger command. But by default this command is not available. we have to install manually.

```
durga@durga-VirtualBox:~$ finger durga
```

Command 'finger' not found, but can be installed with:

```
sudo apt install finger
```

```
durga@durga-VirtualBox:~$ sudo apt install finger
```

```
[sudo] password for durga:
```

```
Reading package lists... Done
```

```
Building dependency tree
```

```
Reading state information... Done
```

The following packages were automatically installed and are no longer required:

```
fonts-liberation2 fonts-opensymbol
gir1.2-geocodeglib-1.0 gir1.2-gst-plugins-base-1.0
gir1.2-gstreamer-1.0 gir1.2-gudev-1.0
gir1.2-udisks-2.0 grilo-plugins-0.3-base
gstreamer1.0-gtk3 libboost-date-time1.65.1
libboost-filesystem1.65.1 libboost-iostreams1.65.1
libboost-locale1.65.1 libcdr-0.1-1
libclucene-contribs1v5 libclucene-core1v5
libcmis-0.5-5v5 libcolamd2 libdazzle-1.0-0
libe-book-0.1-1 libedataserverui-1.2-2 libeot0
libepubgen-0.1-1 libetonyek-0.1-1 libevent-2.1-6
libexiv2-14 libfreerdp-client2-2 libfreerdp2-2
libgee-0.8-2 libgexiv2-2 libgom-1.0-0 libgpgmepp6
libgpod-common libgpod4 liblangtag-common liblangtag1
liblirc-client0 liblua5.3-0 libmediaart-2.0-0
libmsspub-0.1-1 libodfgen-0.1-1 libqqwing2v5 libraw16
librevenge-0.0-0 libsgutils2-2 libssh-4
```




```
libsuitesparseconfig5 libvncclient1 libwinpr2-2  
libxapian30 libxmlsec1 libxmlsec1-nss lp-solve  
media-player-info python3-mako python3-markupsafe  
syslinux syslinux-common syslinux-legacy  
usb-creator-common
```

Use 'sudo apt autoremove' to remove them.

The following NEW packages will be installed:

```
finger
```

0 upgraded, 1 newly installed, 0 to remove and 235 not upgraded.

Need to get 15.6 kB of archives.

After this operation, 48.1 kB of additional disk space will be used.

```
Get:1 http://in.archive.ubuntu.com/ubuntu bionic/universe amd64 finger amd64 0.17-15.1 [15.6 kB]
```

```
Fetch: 15.6 kB in 8s (1,977 B/s)
```

Selecting previously unselected package finger.

(Reading database ... 119576 files and directories currently installed.)

```
Preparing to unpack .../finger_0.17-15.1_amd64.deb ...
```

```
Unpacking finger (0.17-15.1) ...
```

```
Setting up finger (0.17-15.1) ...
```

```
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

```
durga@durga-VirtualBox:~$ finger durga
```

```
Login: durga Name: durga
```

```
Directory: /home/durga Shell: /bin/bash
```

```
On since Tue Jan 28 16:56 (IST) on :0 from :0 (messages off)
```

```
Mail last read Wed Jan 8 20:44 2020 (IST)
```

```
No Plan.
```

5) How to delete user?

We have to use deluser command.

To delete user compulsory sudo permission must be required.

```
durga@durga-VirtualBox:~$ sudo deluser pyuser2
```

```
Removing user `pyuser2' ...
```

```
Warning: group `pyuser2' has no more members.
```

```
Done.
```

```
durga@durga-VirtualBox:~$ su pyuser2
```

```
No passwd entry for user 'pyuser2'
```

Note: We can check whether user deleted or not by using /etc/passwd file.



6) How to Delete Group?

We have to use `delgroup` command.

For this `sudo` access must be required and no user associated with the group.

```
durga@durga-VirtualBox:~$ sudo delgroup pythongroup
/usr/sbin/delgroup: `pyuser1' still has `pythongroup' as their primary group!
```

```
durga@durga-VirtualBox:~$ sudo delgroup demogroup1
Removing group `demogroup1' ...
Done.
```

7) How to Change Ownership of a File:

We can change by using `chown` command.

But for this `sudo` permission required.

```
durga@durga-VirtualBox:~$ touch ddd.txt
durga@durga-VirtualBox:~$ ls -l ddd.txt
-rw-r--r-- 1 durga durga 0 Jan 28 17:16 ddd.txt
durga@durga-VirtualBox:~$ chown root ddd.txt
chown: changing ownership of 'ddd.txt': Operation not permitted
durga@durga-VirtualBox:~$ sudo chown root ddd.txt
durga@durga-VirtualBox:~$ ls -l ddd.txt
-rw-r--r-- 1 root durga 0 Jan 28 17:16 ddd.txt
```

8) How to Change Group Membership of a File?

We have to use `chgrp` command.

`chgrp` means change group.

To use this command compulsory `sudo` permission must be required.

```
durga@durga-VirtualBox:~$ ls -l ddd.txt
-rw-r--r-- 1 root durga 0 Jan 28 17:16 ddd.txt
durga@durga-VirtualBox:~$ sudo chgrp root ddd.txt
durga@durga-VirtualBox:~$ ls -l ddd.txt
-rw-r--r-- 1 root root 0 Jan 28 17:16 ddd.txt
```

9) How to Change Group of a User:

We have to use `usermod` command.

Syntax: `$ usermod -g groupname username`

```
durga@durga-VirtualBox:~$ sudo usermod -g durga pyuser7
durga@durga-VirtualBox:~$ su - pyuser7
Password:
pyuser7@durga-VirtualBox:~$ groups
durga
```



Note: We can find groups associated with particular user by using groups command as follows

```
$ groups username
```

10) Add a User to Multiple Groups:

While assigning the secondary groups to a user account, we can easily assign multiple groups at once by separating the list with a comma.

```
$ usermod -a -G group1,group2,group3 username
```

```
durga@durga-VirtualBox:~$ groups uiuser1
```

```
uiuser1 : uiuser1
```

```
durga@durga-VirtualBox:~$ sudo usermod -a -G uiuser2,uiuser3 uiuser1
```

```
durga@durga-VirtualBox:~$ groups uiuser1
```

```
uiuser1 : uiuser1 uiuser2 uiuser3
```

sudo:

sudo means Super User Do.

We can use sudo command to execute commands as another user, mostly root user.

Eg: \$ sudo adduser durga2

We are executing adduser command as root.

```
$ sudo -u durga1 command
```

We are using executing the command as durga1 user.

Note: If we want to use sudo command for any user other than root then we have to use -u option.

The following two commands are equal.

```
$ sudo command
```

```
$ sudo -u root command
```

sudoers File:

Which commands can be executed by using sudo of a particular user, information is configured in sudoers file, which is present in /etc directory.

System Administrators are responsible to configure this file.

```
durgasoft@durgasoft:~$ ls -l /etc/sudoers
```

```
-r--r----- 1 root root 755 Jan 18 2018 /etc/sudoers
```

```
durgasoft@durgasoft:~$ cat /etc/sudoers
```



```
cat: /etc/sudoers: Permission denied
durgasoft@durgasoft:~$ sudo cat /etc/sudoers
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults      env_reset
Defaults      mail_badpass
Defaults
    secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/b
in"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin   ALL=(ALL) ALL

# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

#includedir /etc/sudoers.d
```

How to Check the allowed Commands by sudo for a Particular User:

```
$ sudo -l
```

It will display allowed commands of current user by sudo.

```
$ sudo -l -u username
```

It will display allowed commands of provided user by sudo.

```
durgasoft@durgasoft:~$ sudo -l
```

Matching Defaults entries for durgasoft on durgasoft:

env_reset, mail_badpass,



```
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

User durgasoft may run the following commands on durgasoft:

(ALL : ALL) ALL

Q. What is the difference between adduser and useradd in Linux?

Useradd is built-in Linux command that can be found on any Linux system. However, creating new users with this low-level is a difficult and lengthy task. First we have to create user and then we have to provide password and extra information separately.

Adduser is not a standard Linux command. It is essentially a Perl script that uses the useradd command in the background. This high-level utility is more efficient in properly creating new users on Linux. Default parameters for all new users can also be set through the adduser command.

```
durgasoft@durgasoft:/etc$ sudo adduser durga5
Adding user `durga5' ...
Adding new group `durga5' (1004) ...
Adding new user `durga5' (1004) with group `durga5' ...
Creating home directory `/home/durga5' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for durga5
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n]
durgasoft@durgasoft:/etc$ sudo useradd durga98
```

How to Check available Groups:

To check available groups:

```
$ groups
```

This command lists all the groups that you belong to.

To check which group a certain user belongs to:

```
$ groups [username]
```



How to Change Password of User:

1. To change a password for user named durga:

```
$ sudo passwd durga
```

2. To change a password for root user:

```
$ sudo passwd root
```

3. To change your own password:

```
$ passwd
```



Topic-40: Working with putty, Mobaxterm and XShell

Working with Putty:

Putty can be used to connect with remote linux servers from our local machine.

PuTTY is an SSH and telnet client can be used to connect with remote servers.

It is developed originally by Simon Tatham for the Windows platform.

PuTTY is open source software.

How to download and install:

<https://www.putty.org/>

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

By using Package Files:

64-bit: putty-64bit-0.73-installer.msi

click next next...

By using directly putty.exe File:

Alternative binary files

64-bit: putty.exe

Requirements for connecting with Putty:

1. Compulsory our Linux machine should has IPAddress
2. SSH server should be running on the linux machine

How to configure ipaddress for our ubuntu Machine:

By default ipaddress is not configured for ubuntu.

We can configure as follows

Oracle Virtual Box → ubuntu Machine → Settings → Network →

Attached To: Bridged Adapter

Name: Broadcom 802... (For wireless networks)

RealTekPCle... (For cable internet)

Now ipaddress will be assigned to our linux system.



How to check ipaddress:

From the terminal we can use any of the following commands:

1. ifconfig
2. ip a
3. ip -br a

```
durgasoft@durgasoft:~$ ip -br a
lo                UNKNOWN    127.0.0.1/8  ::1/128
enp0s3            UP          192.168.0.172/24  fe80::7e7f:57d0:5640:414e/64
```

The ipaddress is: 192.168.0.172

How to install ssh Server and Client in the Linux:

By default in ubuntu ssh server is not available and we have to install separately.

```
sudo apt-get install openssh-server openssh-client
```

We have to restart either ssh server or total ubuntu system.

```
sudo service ssh restart
```

We can check ssh status:

```
sudo service ssh status
```

Open putty, provide ipaddress, click Open

Working with Mobaxterm:

<https://mobaxterm.mobatek.net/>

Download → Home Edition → Download Now → MobaXterm Home Edition v12.4
(Installer Edition) → Download zip File and unzip → Click on installer

Open Mobaxterm → Right click on User Sessions → New Session → SSH → Provide IP Address → OK

Working with XShell:

<https://www.netsarang.com/en/xshell-download/>

Free License for

Home and School Users

click on Free Licensing Page → Provide Name and Mail id
we will get download link to the mail.

Open XShell → Right Click on All Sessions → New → Session → Provide IPAddress → Click on Connect.



Topic-41: How to customize Open Source Software Code

Linux is an open source software. We can see source code and we can modify based on our requirement.

- * GNU Project started by Richard Stallman on 27th Sept 1983.
- * The main objective of this project is to develop UNIX like operating system which should be Freeware and Open Source.

GNU means GNU Not UNIX

- * In Jan 1984, Stallman quits his job to work full time on GNU project.
- * By 1991 almost GNU project completed except kernel development. Kernel is very important component which is responsible to communicate with hardware, to execute our commands and allocate system resources.
- * In 1991, Linus Torvalds developed Linux kernel as the part of his M.Sc thesis at University of Helsinki, Finland.
- * With his Linux kernel contribution, GNU project was completed and GNU operating System ready.

uname Command:

We can print system information by using uname command.

We can use -s or --kernel-name option to print the kernel name.

We can use -o or --operating-system option to print the operating system name.

```
durgasoft@durgasoft:~$ uname --kernel-name
```

```
Linux
```

```
durgasoft@durgasoft:~$ uname --operating-system
```

```
GNU/Linux
```

What ever commands we are using in linux are developed by using C language. The corresponding source is available in gnu.org website.

gnu.org → Software → coreutils → Downloads

Stable source releases can be found at <https://ftp.gnu.org/gnu/coreutils/> → Download the file

```
coreutils-8.31.tar.xz
```

```
durgasoft@durgasoft:~/Downloads$ ls
```

```
coreutils-8.31.tar.xz
```



```
durgasoft@durgasoft:~/Downloads$ file coreutils-8.31.tar.xz
coreutils-8.31.tar.xz: XZ compressed data
We have to uncompress and extract this tar file.
```

```
durgasoft@durgasoft:~/Downloads$ tar -xJvf coreutils-8.31.tar.xz
j → meant for bzip2 compression algorithm
J → meant for XZ compression algorithm
```

```
durgasoft@durgasoft:~/Downloads$ ls
coreutils-8.31  coreutils-8.31.tar.xz
```

```
durgasoft@durgasoft:~/Downloads/coreutils-8.31$ ls
ABOUT-NLS      configure.ac      m4               tests
aclocal.m4      COPYING          maint.mk         THANKS
AUTHORS          dist-check.mk    Makefile.am      thanks-gen
bootstrap        doc              Makefile.in      THANKS.in
bootstrap.conf   gnulib-tests     man              THANKS-to-translators
build-aux        GNUmakefile      NEWS             THANKStt.in
cfg.mk           init.cfg         po               TODO
ChangeLog        INSTALL          README
configure        lib              src
```

src directory contains source code of all our core util commands.

```
durgasoft@durgasoft:~/Downloads/coreutils-8.31/src$ gedit date.c
```

```
int
main (int argc, char **argv)
{
    printf("Now onwards in customized way date command will work\n");
    ....
}
```

Now we have to compile this modified c file. For this we required gcc.
gcc is by default not available and we have to install manually.

```
sudo apt-get install gcc
```

We have to execute configure script. It is responsible to configure gcc as per our system architecture.

```
durgasoft@durgasoft:~/Downloads/coreutils-8.31$ ls -l configure
-rwxr-xr-x 1 durgasoft durgasoft 1852444 Mar 11 2019 configure
```



This configure script will create one special file Makefile, which is responsible to install new updated package.

```
durgasoft@durgasoft:~/Downloads/coreutils-8.31$ bash configure
```

```
....
```

```
config.status: creating Makefile
config.status: creating po/Makefile.in
config.status: creating gnulib-tests/Makefile
config.status: creating lib/config.h
config.status: executing depfiles commands
config.status: executing po-directories commands
config.status: creating po/POTFILES
config.status: creating po/Makefile
```

We required make command which is responsible to compile all c files into .o files.
make command is also helpful to install this updated coreutils package.

We can install make command as follows:

```
sudo apt-get install make
```

```
durgasoft@durgasoft:~/Downloads/coreutils-8.31$ sudo apt-get install make
```

Execute make command to compile all c files into .o files

```
durgasoft@durgasoft:~/Downloads/coreutils-8.31$ make
```

We have to install this updated core coreutils package by using make command.

```
sudo make install
```

Summary:

- 1) download coreutils package from gnu.org
- 2) uncompress and extract tar file
- 3) Perform required modifications for the c files in src folder
- 4) install gcc
- 5) execute configure script
- 6) install make command
- 7) make package by compiling all c files
- 8) install updated package by using make command.



Topic-42: Process Management

We can perform process management by using ps command.

ps → Means Process Status

1) Listing Processes:

1) \$ ps

If we are not passing any argument, then it shows processes related to current session.

```
durgasoft@durgasoft:~$ ps
PID TTY      TIME CMD
20881 pts/0    00:00:00 bash
20890 pts/0    00:00:00 ps
```

2) \$ ps -e

-e means Everything. It shows all processes.

3) \$ ps -ef

It shows full listing of all processes
-f means Full Listing

```
durgasoft@durgasoft:~$ ps -ef
UID      PID  PPID  C STIME TTY      TIME CMD
root      1    0  0 19:41 ?        00:00:02 /sbin/init splash
root      2    0  0 19:41 ?        00:00:00 [kthreadd]
root      3    2  0 19:41 ?        00:00:00 [rcu_gp]
```

UID → UserId/ User Name

PID → Process ID (Every Process has Unique ID)

PPID → Parent Process ID

2) Filtering Processes:

We can filter processes based on UID, PID and PPID.

1) \$ ps -f -u durgasoft

It shows all processes which are running related user durgasoft

2) \$ ps -f -u root

It shows all processes which are running related to root user



3) \$ ps -f -p 20906

It shows process information related to 20906 process id.

4) \$ ps -f --ppid 1266

It shows all process information related to parent process id 1266.

Commonly used other ps Commands:

1) \$ ps -eH

Display a process tree.

```
1 ?      00:00:02  systemd
288 ?    00:00:00  systemd-journal
345 ?    00:00:00  systemd-udev
408 ?    00:00:01  systemd-resolve
559 ?    00:00:00  rsyslogd
560 ?    00:00:00  udisksd
564 ?    00:00:00  cupsd
692 ?    00:00:00  dbus
693 ?    00:00:00  dbus
569 ?    00:00:00  cron
570 ?    00:00:02  dbus-daemon
614 ?    00:00:01  NetworkManager
616 ?    00:00:00  ModemManager
```

2) \$ ps -e --forest

Display proces tree

```
994 ?      00:00:00  VBoxClient
995 ?      00:00:00  \_ VBoxClient
1010 ?     00:00:00  gdm3
1027 ?     00:00:00  \_ gdm-session-wor
1051 tty1   00:00:00  |   \_ gdm-wayland-ses
1055 tty1   00:00:00  |       \_ gnome-session-b
1061 tty1   00:00:13  |           \_ gnome-shell
1078 tty1   00:00:00  |               \_ Xwayland
1107 tty1   00:00:00  |                   \_ ibus-daemon
1110 tty1   00:00:00  |                       \_ ibus-dconf
1230 tty1   00:00:00  |                           \_ ibus-engine-sim
1133 tty1   00:00:00  |   \_ gsd-xsettings
1138 tty1   00:00:00  |       \_ gsd-a11y-settin
1143 tty1   00:00:00  |           \_ gsd-clipboard
1146 tty1   00:00:00  |               \_ gsd-color
1151 tty1   00:00:00  |                   \_ gsd-datetime
```



3) \$ pstree

Display processes in a tree format

```
durgasoft@durgasoft:~$ pstree
systemd--ModemManager--2*[{ModemManager}]
        --NetworkManager--dhclient
                        --2*[{NetworkManager}]
        --2*[VBoxClient--VBoxClient]
        --2*[VBoxClient--VBoxClient--{VBoxClient}]
        --VBoxClient--VBoxClient--2*[{VBoxClient}]
        --VBoxService--7*[{VBoxService}]
        --accounts-daemon--2*[{accounts-daemon}]
        --acpid
        --avahi-daemon--avahi-daemon
        --boltd--2*[{boltd}]
        --colord--2*[{colord}]
        --cron
        --cups-browsed--2*[{cups-browsed}]
        --cupsd--2*[dbus]
```

4) \$ top

We can use top command to see processes with consumed resources like memory and cpu utilization. use 'q' to come out of top command.

```
top - 21:41:31 up 2:00, 2 users, load average: 0.06, 0.08, 0.08
Tasks: 239 total, 1 running, 201 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.9 us, 0.7 sy, 0.0 ni, 97.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4169364 total, 1617212 free, 1336860 used, 1215292 buff/cache
KiB Swap: 483800 total, 483800 free, 0 used. 2532032 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1477	durgaso+	20	0	3479664	289172	118692	S	3.0	6.9	3:13.07	gnome-shell
1286	durgaso+	20	0	609356	142852	79532	S	1.7	3.4	0:50.24	Xorg
20871	durgaso+	20	0	804408	38456	28308	S	1.0	0.9	0:14.59	gnome-terminal-
21385	durgaso+	20	0	48996	3872	3152	R	1.0	0.1	0:01.51	top
1591	durgaso+	20	0	504684	23460	18144	S	0.7	0.6	0:00.36	gsd-xsettings
644	root	20	0	1165108	24428	13884	S	0.3	0.6	0:02.41	snapped

5) \$ htop

It is similar to top , but output will come in gui style.

We have to install separately.

```
$ sudo apt install htop
```



How to kill the process:

\$ kill -9 procesid

Eg: kill -9 21484

We can kill multiple processes simultaneously.

kill -9 pid1 pid2 pid3 ...



Topic-43: Communication Commands (write, wall, msg, mail)

How to use Gmail from the Ubuntu Terminal to send Emails

The commands which can be used for communication purpose, are called communication commands.

There are two types of communications

- 1) Online Communication
- 2) Offline Communication

1) Online Communication:

Both sender and receiver should be online.

It is also known as two way communication or synchronous communication.

It is something like phone call.

We can implement online communication by using the following commands:

write, wall, msg

2) Offline Communication:

Only sender should be online, but receiver need not be online.

It is also known as one way communication or asynchronous communication.

It is something like sms or mail.

We can implement offline communication by using mail command.

write Command (For Online Communication):

We can use write command to write message to another user account, but both must be logged into the server.

Syntax: \$ write username/terminal name

Eg:

```
$ write demo1
```

```
Hello i am in linux class
```

```
ctrl+d
```

We can also use write command to send message for multiple users.

```
$ write demo1 demo2 demo3
```

```
Hello i am in linux class
```

```
ctrl+d
```




wall Command (For Online Communication):

We can use wall command to broadcast messages to all users who are logged in.

```
$ wall
```

```
Hello I am in Linux class
```

```
ctrl+d
```

How to disable Messages:

If the receiver is very busy (like durga sir) and he don't want to receive any messages then he can block messages by using mesg command.

```
$ mesg n → Disabling messages
```

```
$ mesg y → Enabling to receive messages
```

```
durgasoft@durgasoft:~$ whoami
```

```
durgasoft
```

```
durgasoft@durgasoft:~$ mesg
```

```
is y
```

```
durgasoft@durgasoft:~$ write demo1
```

```
Hello demo1 i am in Linux class
```

```
Still new year but faculty is very cruel
```

```
ctrl+d
```

```
demo1@durgasoft:~$ whoami
```

```
demo1
```

```
demo1@durgasoft:~$ mesg
```

```
is y
```

```
demo1@durgasoft:~$
```

```
Message from durgasoft@durgasoft on pts/1 at 20:14 ...
```

```
Hello demo1 i am in Linux class
```

```
Still new year but faculty is very cruel
```

```
EOF
```

```
^C
```

mesg n Option:

```
demo1@durgasoft:~$ whoami
```

```
demo1
```

```
demo1@durgasoft:~$ mesg n
```

```
demo1@durgasoft:~$ mesg
```

```
is n
```

```
durgasoft@durgasoft:~$ whoami
```

```
durgasoft
```

```
durgasoft@durgasoft:~$ write demo1
```

```
write: demo1 has messages disabled
```



Checking Wall Command:

```
durgasoft@durgasoft:~$ wall  
Hello Friends This is common message  
ctrl+d
```

Broadcast message from durgasoft@durgasoft (pts/1) (Wed Jan 1 20:25:52 2020):

Hello Friends This is common message

This message will be broadcasted to all active users.

demo1 User:

Broadcast message from durgasoft@durgasoft (pts/1) (Wed Jan 1 20:25:52 2020):

Hello Friends This is common message

mail Command:

mail command is available in mailutils package. Hence we have to install this package if it is not available.

```
$ sudo apt install mailutils
```

Syntax to send mail:

```
durgasoft@durgasoft:~$ mail demo1 demo2 demo3  
Subject: Hello  
Hello This is sample message
```

Regards

Durga

EOT

ctrl+d

```
durga2@durga-VirtualBox:~$ mail
```

Heirloom mailx version 12.5 6/20/10. Type ? for help.

"/var/mail/durga2": 2 messages

>O 1 durga Fri Jan 3 18:29 17/607 Hello Friends

O 2 durga Fri Jan 3 18:30 17/596 Greetings

?

Held 2 messages in /var/mail/durga2



How to use Gmail from the Ubuntu Terminal to send Emails:

Step 1: Open the Terminal application

Step 2: Update the repository index

```
$ sudo apt-get update
```

Step 3: Install Msmtp client

msmtp is an SMTP client that can be used to send mails from Mutt and probably other MUAs (mail user agents). It forwards mails to an SMTP server (for example at a free mail provider), which takes care of the final delivery.

```
$ sudo apt-get install msmtp-mta
```

Step 4: Configure msmtp for gmail

Now is the time to configure msmtp by telling it our gmail credentials, the port to use, the host, and some other authorization and connection details:

Open a file named msmtprc in one of text editors.

```
$ gedit ~/.msmtprc
```

Then, copy the following code in the empty file:

```
#Gmail account
defaults
#change the location of the log file to any desired location.
logfile ~/.msmtprc
account gmail
auth on
host smtp.gmail.com
from durgaunix1@gmail.com
auth on
tls on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
user durgaunix1@gmail.com
password unix1234
port 587
#set gmail as your default mail server.
account default : gmail
```

Saving your password in text format in any of your files is never a good idea. So, you can secure the file by running the following command: `$ chmod 600 ~/.msmtprc`



Step 5: Install heirloom-mailx

At this point, we have configured our computer to talk to the remote Gmail server. What we need to do now is, set up a command-line interface that will let us compose emails to be sent. Mailx is the program that will let us do all this, and here is how we can install it:

```
$ sudo apt-get install heirloom-mailx
```

Note:

Important: If you are not able to find the package in your already added repositories, open the sources.list file as follows:

```
$ nano /etc/apt/sources.list
```

Then, add the following line to add the rusty-security main universe repository from where we will install the mailx utility.

```
deb http://security.ubuntu.com/ubuntu trusty-security main universe
```

Also, do not forget to run the following command before performing the installation:

```
$ sudo apt-get update
```

Step 6: Configure Mailx

Open a file named .mailrc through one of your favorite text editors.

```
$ nano ~/.mailrc
```

Then, add the following lines in that file and save it.

```
set sendmail="/usr/bin/msmtp"  
set message-sendmail-extra-arguments="-a gmail"
```

Step 7: Send an Email through the Terminal

We are now ready to send an email through our configured gmail account to a receiver on any domain. Following is the basic syntax for sending such an email:

```
$ mail -s "subject" -a "attachment-if-any" "receiver@some-domain.com"  
Sending an attachment along with the email is optional.
```

```
$ sudo mail -s "Important file" -a "demo.py" "durgaadvjava@gmail.com"
```

I used the following command to send an email:

```
sudo mail -s "sample text" durgaadvjava@gmail.com
```



Send an Email through the Terminal

As you hit Enter, you will be allowed to enter the body of the email. Once you are done with entering the email body, hit Ctrl+D. This will mark the end of the email body and send it to the respective receiver ID.

Authenticate as admin

The EOT at the end of the output will indicate that your email has been sent.

However, you might encounter the most common error, same as I did:

This error is mostly encountered when you have not allowed access to less secure apps on your gmail. This security setting can be changed through the following link:

<https://myaccount.google.com/lesssecureapps>

When you do so, a notification will be sent to you (mostly on your phone, when you have configured your phone number with gmail). When you permit this change of setting, gmail will allow access to less secure apps such as the one we are using.



Topic-44: Package Management

Package:

Package is nothing but a collection of files. It contains data and metadata like description, version and dependencies etc.

Package Manager:

It is responsible to install, upgrades and removes packages.

It is responsible to manage dependencies.

It will track all installed packages.

Eg: yum is the package manager in RedHat Linux.

apt is the package manager in ubuntu.

Advanced Packaging Tool (apt):

1) Searching for Packages:

\$ apt-cache search string

It will search for packages based on given search string.

```
durgasoft@durgasoft:~$ apt-cache search xeyes
```

```
x11-apps - X applications
```

```
xfce4-eyes-plugin - eyes that follow your mouse for the Xfce4 panel
```

2) Install a New Package:

\$ apt-get install package

3) To remove Package without removing Configuration:

\$ apt-get remove package

Only package will be removed but not configuration

4) To remove Package and its Configuration:

\$ apt-get purge package

both package and its configuration will be removed.

5) To get Information about a Package:

\$ apt-cache show package

Display information about the package

6) dpkg Command:

\$ dpkg -l

List all installed packages



Some Funny Ubuntu Applications:

1) xeyes:

xeyes is a gui program that draw a pair of eyes on the desktop which follow the mouse cursor. The eyes would look where ever the mouse cursor goes.

```
$ sudo apt install x11-apps
$ xeyes
```

2) sl Command (Steam Locomotive)

If we type sl instead of ls, then train image will be displayed on the screen.

```
$ sudo apt install sl
$ sl
```

3) cowsay Command:

An ASCII cow will be displayed in the terminal, which can tell whatever we want.

```
$ sudo apt install cowsay
$ cowsay I love Linux
```

```
durga@durga-VirtualBox:~$ cowsay I Love Sunny
```

```
< I Love Sunny >
-----
      \   ^__^
       \  (oo)\_______
          (__)\\       )\/\
              ||----w |
              ||     ||
```

4) xcowsay Command:

xcowsay is a graphical program and its response is similar to cowsay but in a graphical manner.

```
$ sudo apt install xcowsay
$ xcowsay I Love Linux
```

5) yes Command:

It is responsible to generate specified response keep on.

```
$ yes I Love Linux
I Love Linux
I Love Linux
```

I Love Linux
I Love Linux
I Love Linux
...
ctrl+c → To stop

This command mostly used by system administrators to generate some data for testing purposes.

6) aafire Command:

To display fire in the terminal we can use aafire command. Press any key to interrupt.

```
$ sudo apt install libaa-bin
$ aafire
```

7) figlet Command:

The figlet command can be used to draw large sized text banners.

```
$ sudo apt install figlet
$ figlet Welcome
durga@durga-VirtualBox:~$ figlet Welcome
```

A 4x16 grid of 64 small, stylized, abstract symbols or characters, possibly representing a binary or hexadecimal code. The symbols are arranged in four rows and sixteen columns, with each symbol being a unique, simplified representation of a character or digit.

8) toilet Command:

The toilet command is similar to figlet command and it can be used to draw large sized text banners by using smaller characters

```
$ sudo apt install toilet
$ toilet Welcome
```

durga@durga-VirtualBox:~\$ toilet Welcome

```
m      m      ""#
#   #   #   mmm    #       mmm      mmm      mmmmmm      mmm
" "#"# # #" #     #       #" "      #" "#   # # #   #" #
## ##" #""""      #       #         #        # # #   #""""
#    #    "#mm"     "mm    "#mm"    "#m#"   # # #   "#mm"
```




9) fortune Command:

The fortune command will put up a random fortune message.

```
$ sudo apt install fortune-mod
```

```
$ fortune -s
```

-s option can be used to generate small messages

```
durga@durga-VirtualBox:~$ fortune -s
```

You display the wonderful traits of charm and courtesy.

```
durga@durga-VirtualBox:~$ fortune -s
```

You will be traveling and coming into a fortune.

```
durga@durga-VirtualBox:~$ fortune -s
```

A long-forgotten loved one will appear soon.

Buy the negatives at any price.

```
durga@durga-VirtualBox:~$ fortune -s
```

You are farsighted, a good planner, an ardent lover, and a faithful friend.

```
durga@durga-VirtualBox:~$ fortune -s
```

It is so very hard to be an

on-your-own-take-care-of-yourself-because-there-is-no-one-else-to-do-it-for-you
grown-up.

Note: \$ fortune | xcowsay

10) factor Command:

We can use factor command to generate all factors for the given number.

It is by default available and we are not required to install separately.

```
durga@durga-VirtualBox:~$ factor 60
```

```
60: 2 2 3 5
```

```
durga@durga-VirtualBox:~$ factor 100
```

```
100: 2 2 5 5
```



Topic-45: Install and Working with MySQL Database

1. Update the package list to ensure that the latest version and dependencies for MySQL.
`$ sudo apt update`

2. Download and install MySQL
`$ sudo apt install mysql-server`

Once the package installer has finished, we can check to see if the MySQL service is running.

3. Checking the status of mysql service
`$ sudo service mysql status`

If running, you will see a green Active status like below.
Active: active (running)

press q to exit the service status.

4. Configure Security
We have to run `mysql_secure_installation` to configure security for your MySQL server.

`$ sudo mysql_secure_installation`

`durga@durga-VirtualBox:~$ sudo mysql_secure_installation`

Securing the MySQL server deployment.

Enter password for user root:
The 'validate_password' plugin is installed on the server.
The subsequent steps will run with the existing configuration of the plugin.
Using existing password for root.

Estimated strength of the password: 50
Change the password for root ? ((Press y|Y for Yes, any other key for No) : y

New password:

Re-enter new password:

Estimated strength of the password: 100



Do you wish to continue with the password provided? (Press y|Y for Yes, any other key for No) : y

By default, a MySQL installation has an anonymous user, allowing anyone to log into MySQL without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from 'localhost'. This ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : n

... skipping.

By default, MySQL comes with a database named 'test' that anyone can access. This is also intended only for testing, and should be removed before moving into a production environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : n

... skipping.

Reloading the privilege tables will ensure that all changes made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!

5. Connect to Mysql prompt:

```
durga@durga-VirtualBox:~$ sudo mysql -u root -p
```

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 18

Server version: 5.7.28-0ubuntu0.18.04.4 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.



Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> show databases;
```

```
+-----+  
| Database |  
+-----+  
| information_schema |  
| mydb |  
| mysql |  
| performance_schema |  
| sys |  
+-----+
```

5 rows in set (0.05 sec)

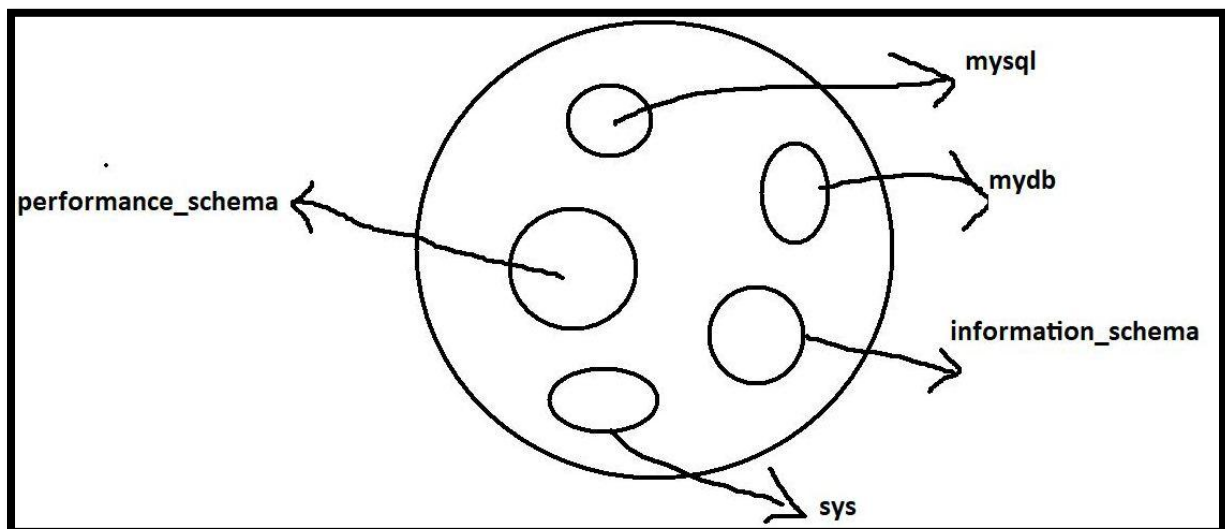
```
mysql> exit
```

Bye

In MySQL, everything we have to work with our own databases, which are also known as logical databases.

The default databases are:

```
information_schema  
mydb  
mysql  
performance_schema  
sys
```



Here only one physical database but 5 logical databases are available.



Commonly used Commands:

1. To know available databases

```
mysql>show databases;
```

2. To create our own logical database

```
mysql>create database durgadb;
```

3. To drop our own database

```
mysql>drop database durgadb;
```

4. To use a particular database:

```
mysql>use durgadb; or mysql>connect durgadb;
```

5. To create a table

```
mysql>create table employees(eno int(5) primary key,ename varchar(20),esal double(10,2),eaddr varchar(20));
```

6. mysql> desc employees;

Field	Type	Null	Key	Default	Extra
eno	int(5)	NO	PRI	NULL	
ename	varchar(20)	YES		NULL	
esal	double(10,2)	YES		NULL	
eaddr	varchar(20)	YES		NULL	

4 rows in set (0.01 sec)

7. To insert data

```
mysql> insert into employees values(100,'durga',1000,'Hyd');
```

```
mysql> insert into employees values(200,'sunny',2000,'Mumbai');
```

```
mysql> insert into employees values(300,'bunny',3000,'Hyd');
```

```
mysql> insert into employees values(400,'chinny',4000,'Hyd');
```

8. mysql> select * from employees;

eno	ename	esal	eaddr
100	durga	1000.00	Hyd
200	sunny	2000.00	Mumbai
300	bunny	3000.00	Hyd
400	chinny	4000.00	Hyd

4 rows in set (0.00 sec)



How to execute SQL Queries from ubuntu Terminal directly:

We have to use mysql command, and we have to provide username,password,database name and sql query as options.

The following are various options allowed with mysql command.

--user, -u The MySQL user name to use when connecting to the server.
--password, -p The password to use when connecting to the server.
--database, -D The database to use.
--host, -h Connect to the MySQL server on the given host.
--skip-column-names, -N Do not write column names in results.
--batch, -B Print results using tab as the column separator,
with each row on a new line.
-e to specify query

Eg:

```
$ mysql -u root -pDurga@123 -Ddurgadb -e "select * from employees"
```

```
durga@durga-VirtualBox:~$ mysql -u root -pDurga@123 -Ddurgadb -e "select * from employees"
```

```
mysql: [Warning] Using a password on the command line interface can be insecure.
```

eno	ename	esal	eaddr
100	durga	1000.00	Hyd
200	sunny	2000.00	Mumbai
300	bunny	3000.00	Hyd
400	chinny	4000.00	Hyd

How to suppress warning:

```
$ mysql -u root -pDurga@123 -Ddurgadb -e "select * from employees" 2>/dev/null
```

```
durga@durga-VirtualBox:~$ mysql -u root -pDurga@123 -Ddurgadb -e "select * from employees" 2>/dev/null
```

eno	ename	esal	eaddr
100	durga	1000.00	Hyd
200	sunny	2000.00	Mumbai
300	bunny	3000.00	Hyd
400	chinny	4000.00	Hyd



Shell Script to fetch and display Records from MySQL Database:

test.sh

```
#!/bin/bash
```

```
#Declaring mysql DB connection
```

```
DB_USER='root'
```

```
DB_PASSWD='Durga@123'
```

```
DB_NAME='durgadb'
```

```
#Prepare sql query
```

```
SQL_Query='select * from employees'
```

```
#sql query execution
```

```
mysql -u $DB_USER -p$DB_PASSWD -D$DB_NAME -e "$SQL_Query" 2>/dev/null
```

```
durga@durga-VirtualBox:~/scripts$ test.sh
```

```
+-----+-----+-----+-----+
| eno |  |  |  |  |
+-----+-----+-----+-----+
| 100 | durga | 1000.00 | Hyd |
| 200 | sunny | 2000.00 | Mumbai |
| 300 | bunny | 3000.00 | Hyd |
| 400 | chinny | 4000.00 | Hyd |
+-----+-----+-----+-----+
```

Shell Script to Read Data from User dynamically and Insert into the Database:

```
#!/bin/bash
```

```
#Declaring mysql DB connection
```

```
DB_USER='root'
```

```
DB_PASSWD='Durga@123'
```

```
DB_NAME='durgadb'
```

```
while [ true ]
```

```
do
```

```
read -p "Employee Number: " eno
```

```
read -p "Employee Name: " ename
```

```
read -p "Employee Salary: " esal
```

```
read -p "Employee Address: " eaddr
```

```
SQL_Query="insert into employees values($eno,'$ename',$esal,'$eaddr');"
```

```
mysql -u $DB_USER -p$DB_PASSWD -D$DB_NAME -e "$SQL_Query" 2>/dev/null
```

```
echo "Employee Record Inserted Successfully"
```

```
read -p "Do you want to insert one more record [Yes|No]: " option
```



```
case $option in
    [yY]|[Yy][eE][sS])
        continue
        ;;
    [nN]|[nN][oO])
        break
        ;;
    esac
done
```

```
durga@durga-VirtualBox:~/scripts$ test.sh
Employee Number: 500
Employee Name: Dhoni
Employee Salary: 5000
Employee Address: Ranchi
Employee Record Inserted Successfully
Do you want to insert one more record [Yes|No]: y
Employee Number: 600
Employee Name: Kohli
Employee Salary: 6000
Employee Address: Delhi
Employee Record Inserted Successfully
Do you want to insert one more record [Yes|No]: NO
```




Topic-46: Working with Java

To develop and run java applications we required java development kit(JDK).
We have to install separately.

```
durga@durga-VirtualBox:~$ javac
```

Command 'javac' not found, but can be installed with:

```
sudo apt install default-jdk
sudo apt install openjdk-11-jdk-headless
sudo apt install ecj
sudo apt install openjdk-8-jdk-headless
```

```
$ sudo apt install openjdk-8-jdk-headless
```

Note: In the old versions we have to set PATH explicitly. But in the newer versions we are not required to set.

After installation we have to set PATH, then only java related commands will work.
All java related commands are available in the following location:
`/usr/lib/jvm/java-8-openjdk-amd64/bin`

```
durga@durga-VirtualBox:/usr/lib/jvm/java-8-openjdk-amd64/bin$ ls
clhsdb      javadoc      jinfo        jstatd       schemagen
extcheck    javah        jjs          keytool       serialver
hsdb        javap        jmap         native2ascii  servertool
idlj        java-rmi.cgi jps          orbd          tnameserv
jar         jcmd         jrunscript   pack200       unpack200
jarsigner   jdb          jsadebugd    rmic          wsgen
java        jdeps        jstack       rmid          wsimport
javac       jhat         jstat        rmiregistry   xjc
```

We have to add this location to the PATH variable then only these commands will be available to ubuntu.

```
export PATH=$PATH:/usr/lib/jvm/java-8-openjdk-amd64/bin
```



Test.java

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Java From Ubuntu");
    }
}
```

\$javac Test.java

\$ java Test

Java From Ubuntu



Topic-47: Job Control

All jobs are divided into 2 types

- 1) Foreground Jobs
- 2) Background Jobs

\$ cp file1.txt file2.txt → Foreground Job
\$ cp file1.txt file2.txt & → Background Job

```
durga@durga-VirtualBox:~$ cp a.txt b.txt &
[1] 2972
durga@durga-VirtualBox:~$ cp a.txt b.txt &
[2] 2974
[1] Done          cp a.txt b.txt
durga@durga-VirtualBox:~$ jobs
[2]+ Done          cp a.txt b.txt
```

How to Kill Foreground Job:

ctrl+c

How to suspend foreground job:

ctrl+z

How to Resume Foreground Job:

\$fg

test.sh

```
#!/bin/bash
while [ true ]
do
    echo "Inside while loop"
    sleep 1
done
```

```
durga@durga-VirtualBox:~/scripts$ test.sh
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
```



```
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
^C
durga@durga-VirtualBox:~/scripts$ test.sh
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
^Z
[1]+  Stopped                  test.sh
durga@durga-VirtualBox:~/scripts$ fg
test.sh
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
Inside while loop
^C
```

How to Kill Background Job:

```
$ kill -9 pid
```

How to List all Background Jobs:

```
$ jobs
durga@durga-VirtualBox:~/scripts$ jobs
[1]+  Killed                  test.sh
```

How to bring Background Job to Foreground?

```
$ fg jobid
```

Eg: \$fg 1



How to send Foreground Job to Background:

First suspend foreground job by using ctrl+z and then use the following command
\$ bg

Note: In the foreground only one can be executed, but in the background multiple jobs can be executed.



Topic-48: nohup Command

The nohup jobs will create in server account. Hence nohup jobs will be executed even the user disconnected from his account.

Eg: `$ nohup cp file1.txt file2.txt&`

The nohup command will create one nohup.out file and it appends the nohup job output to that nohup.out file.

```
durga@durga-VirtualBox:~/scripts$ nohup test.sh&
```

```
[1] 3395
```

```
durga@durga-VirtualBox:~/scripts$ nohup: ignoring input and appending output to 'nohup.out'
```

```
durga@durga-VirtualBox:~/scripts$ ls
```

```
demo.sh env.sh nohup.out Test.class Test.java test.py test.sh
```

```
durga@durga-VirtualBox:~/scripts$ cat nohup.out
```

```
Inside while loop
```

```
Inside while loop
```

```
Inside while loop
```

Even the user exited or switched to some other user still this nohup job will continue its execution.



Topic-49: at Command

It executes the given job only once, but at specified time.

```
durga@durga-VirtualBox:~/scripts$ at now
```

Command 'at' not found, but can be installed with:

```
sudo apt install at
```

```
$ sudo apt install at
```

Eg 1:

```
durga@durga-VirtualBox:~$ at now
warning: commands will be executed using /bin/sh
at> ls
at> date
at> cal
at> ctrl+d
```

It executes all commands immediately and writes output into user's mailbox.

Eg 2:

```
durga@durga-VirtualBox:~$ at now + 2 minutes
warning: commands will be executed using /bin/sh
at> ls > abcd.txt
at> ctrl+d
job 7 at Wed Jan 8 21:08:00 2020
```

ls will be executed just after 2 minutes and it writes output to abcd.txt

Eg 3:

```
durga@durga-VirtualBox:~$ at now + 2 Hours
warning: commands will be executed using /bin/sh
at> test.sh > a.txt
at> ctrl+d
```

Just after 2 hours test.sh will be executed and writes output to a.txt

Note:

\$ at now + 3 days → It will be executed after 3 days

\$ at now + 1 month → It will be executed after 1 month



\$ at 3PM tomorrow → It will be executed tomorrow at 3PM
\$ at 12:30am aug 01 → It will be executed at 1st august 12:30 AM.

How to List out at Command Jobs:

\$atq

Which jobs are in the queue for at

```
durga@durga-VirtualBox:~$ at now + 30 minutes
warning: commands will be executed using /bin/sh
at> ls > abc.txt
at> <EOT>
job 8 at Wed Jan  8 21:46:00 2020
durga@durga-VirtualBox:~$ date
Wed Jan  8 21:16:32 IST 2020
durga@durga-VirtualBox:~$ at now + 40 minutes
warning: commands will be executed using /bin/sh
at> date > abc.txt
at> <EOT>
job 9 at Wed Jan  8 21:57:00 2020
durga@durga-VirtualBox:~$
```

```
durga@durga-VirtualBox:~$ atq
8      Wed Jan  8 21:46:00 2020 a durga
9      Wed Jan  8 21:57:00 2020 a durga
```

How to Remove at Command Jobs:

\$atrm jobid

Eg: \$ atrm 8

batch Command:

The batch jobs will be executed when ever server is free or server load is less.

```
durga@durga-VirtualBox:~$ batch
warning: commands will be executed using /bin/sh
at> ls
at> date
at> cal
at> ctrl+d
```




Topic-50: Memory related Commands

1) df Command:

df means disk fragmentation.

It displays file system disk space usage. It provides information about the space available on all currently mounted file systems.

```
durga@durga-VirtualBox:~$ df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
udev	1404040	0	1404040	0%	/dev
tmpfs	285584	1584	284000	1%	/run
/dev/sda1	10253588	5512804	4200216	57%	/
tmpfs	1427912	0	1427912	0%	/dev/shm
tmpfs	5120	4	5116	1%	/run/lock
tmpfs	1427912	0	1427912	0%	/sys/fs/cgroup

We can use multiple options with df command.

-h → Human Readable Format

-m → In Mega Bytes

-k → In Kilo Bytes (It is Default)

```
durga@durga-VirtualBox:~$ df -m
```

Filesystem	1M-blocks	Used	Available	Use%	Mounted on
udev	1372	0	1372	0%	/dev
tmpfs	279	2	278	1%	/run
/dev/sda1	10014	5384	4102	57%	/
tmpfs	1395	0	1395	0%	/dev/shm
tmpfs	5	1	5	1%	/run/lock

```
durga@durga-VirtualBox:~$ df -h
```

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	1.4G	0	1.4G	0%	/dev
tmpfs	279M	1.6M	278M	1%	/run
/dev/sda1	9.8G	5.3G	4.1G	57%	/
tmpfs	1.4G	0	1.4G	0%	/dev/shm
tmpfs	5.0M	4.0K	5.0M	1%	/run/lock
tmpfs	1.4G	0	1.4G	0%	/sys/fs/cgroup
/dev/loop1	43M	43M	0	100%	/snap/gtk-common-themes/1313



2) du Command:

du - estimate file space usage

It displays directory wise disk usage in the form of blocks. Each block is 1024 bytes.

```
durga@durga-VirtualBox:~$ du
4      ./dir10
4      ./Downloads/java/corejava
8      ./Downloads/java
12     ./Downloads
du: cannot read directory './.dbus': Permission denied
4      './.dbus'
4      './.config/update-notifier'
8      './.config/gedit'
12     './.config/ibus/bus'
```

With du command we can use multiple options

-k in kilobytes(It is the default value)

-h in human readable form

-m in mega bytes

```
durga@durga-VirtualBox:~$ du -k
4      ./dir10
4      ./Downloads/java/corejava
8      ./Downloads/java
12     ./Downloads
du: cannot read directory './.dbus': Permission denied
4      './.dbus'
4      './.config/update-notifier'
8      './.config/gedit'
12     './.config/ibus/bus'
16     './.config/ibus'
4      './.config/enchant'
84     './.config/pulse'
4      './.config/goa-1.0'
8      './.config/gtk-3.0'
8      './.config/evolution/sources'
12     './.config/evolution'
```

```
durga@durga-VirtualBox:~$ du -h
4.0K   ./dir10
4.0K   ./Downloads/java/corejava
8.0K   ./Downloads/java
12K    ./Downloads
du: cannot read directory './.dbus': Permission denied
4.0K   './.dbus'
```



```
4.0K  ./config/update-notifier
8.0K  ./config/gedit
12K   ./config/ibus/bus
16K   ./config/ibus
4.0K  ./config/enchant
84K   ./config/pulse
4.0K  ./config/goa-1.0
8.0K  ./config/gtk-3.0
8.0K  ./config/evolution/sources
12K   ./config/evolution
4.0K  ./config/gnome-session/saved-session
8.0K  ./config/gnome-session
28K   ./config/nautilus
20K   ./config/dconf
212K  ./config
16K   ./output5
260K  ./cache/mozilla/firefox/z901vq89.default-release/OfflineCache
12M   ./cache/mozilla/firefox/z901vq89.default-release/startupCache
```

```
durga@durga-VirtualBox:~$ du -m
```

```
1     ./dir10
1     ./Downloads/java/corejava
1     ./Downloads/java
1     ./Downloads
```

Note: By default du command will always display disk information only about directories. If we want to get files information also then we have to use -a option.
-a means all

```
durga@durga-VirtualBox:~$ du -a -h | head -10
```

```
du: cannot read directory './.dbus': Permission denied
du: cannot read directory './.cache/doc': Permission denied
du: cannot read directory './.cache/dconf': Permission denied
du: cannot read directory './.gvfs': Permission denied
0     ./dir10/c.txt
0     ./dir10/d.txt
0     ./dir10/a.txt
0     ./dir10/b.txt
4.0K  ./dir10
8.0K  ./ICEauthority
0     ./Downloads/java/corejava/collections.txt
0     ./Downloads/java/corejava/arrays.txt
4.0K  ./Downloads/java/corejava
8.0K  ./Downloads/java
```



Note: We can use -c option with du command to find grand total disk space.

```
durga@durga-VirtualBox:~$ du -a -h -c
```

```
0      ./dir10/c.txt
0      ./dir10/d.txt
0      ./dir10/a.txt
0      ./dir10/b.txt
4.0K   ./dir10
8.0K   ./ICEauthority
.....
40M    total
```

3) free Command:

It displays amount of free and used memory in the system.

```
durga@durga-VirtualBox:~$ free
```

	total	used	free	shared	buff/cache	available
Mem:	2855828	1205468	646652	51552	1003708	1434376
Swap:	483800	0	483800			

Here also we can use -k,-h,-m options.

```
durga@durga-VirtualBox:~$ free -k
```

	total	used	free	shared	buff/cache	available
Mem:	2855828	1205112	646888	51556	1003828	1434716
Swap:	483800	0	483800			

```
durga@durga-VirtualBox:~$ free -h
```

	total	used	free	shared	buff/cache	available
Mem:	2.7G	1.1G	631M	50M	980M	1.4G
Swap:	472M	0B	472M			

```
durga@durga-VirtualBox:~$ free -m
```

	total	used	free	shared	buff/cache	available
Mem:	2788	1176	631	50	980	1401
Swap:	472	0	472			

Note: We can use -l and -t options also.

-l, --lohi

Show detailed low and high memory statistics.

-t, --total

Display a line showing the column totals.

```
durga@durga-VirtualBox:~$ free -l
```

	total	used	free	shared	buff/cache	available
Mem:	2855828	1205100	646660	51552	1004068	1434736
Low:	2855828	2209168	646660			
High:	0	0	0			
Swap:	483800	0	483800			



UNIX/LINUX



```
durga@durga-VirtualBox:~$ free -t
```

	total	used	free	shared	buff/cache	available
Mem:	2855828	1206092	645620	51556	1004116	1433732
Swap:	483800	0	483800			
Total:	3339628	1206092	1129420			



Topic-51: Networking Commands

telnet:

telnet protocol can be used to connect to remote servers.

Syntax:

telnet ipaddress

login:XXXX

password:XXXX

Note: By default telnet is not supported by ubuntu.

```
C:\Users\lenovo>telnet 192.168.100.3
```

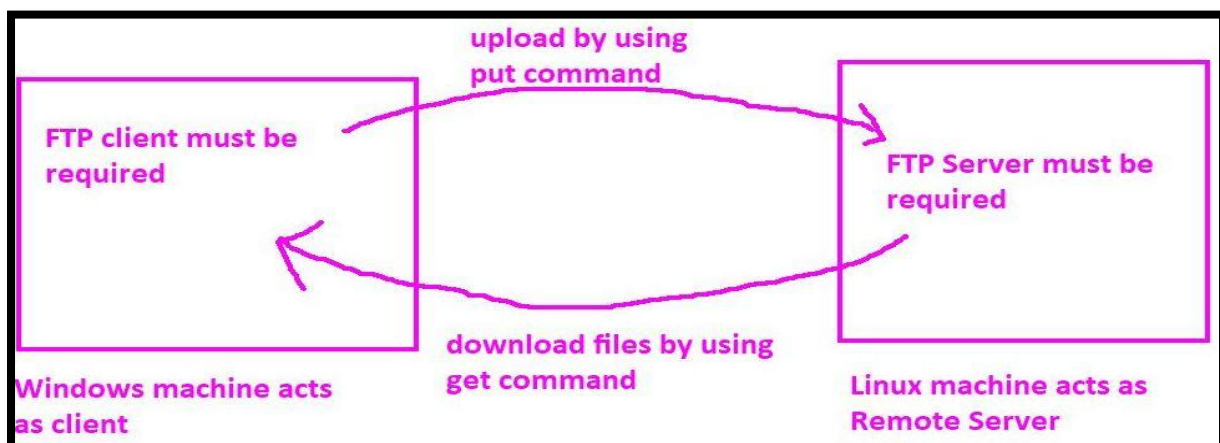
Connecting To 192.168.100.3...Could not open connection to the host, on port 23: Connect failed

Note: telnet is not recommended to use because it is very less secure. Instead of telnet the people are using ssh(Secure Shell), which is more secured. SSH is applicable only for *UX Systems (like Linux, UNIX etc).

putty, mobaxterm, xshell can be used to connect with remote servers by using ssh.
By default ssh server is running on ubuntu.

FTP (File Transfer Protocol):

To upload files to the remote server and to download files from the remote server we have to use FTP.



On windows, by default ftp client is available.

But in ubuntu by default ftp server won't be available, we have to install separately.



How to install FTP Server on ubuntu:

```
$ sudo apt update  
$ sudo apt install vsftpd
```

After installation, by default ftp server won't run. We have to start manually.

```
$ sudo service vsftpd status  
$ sudo service vsftpd start
```

By default, we can download(get) files from remote server. But we cannot upload(put). To enable upload operation we have to edit the following file:
/etc/vsftpd.conf

In this file we have to remove comment for the following line:

```
# Uncomment this to enable any form of FTP write command.  
write_enable=YES
```

On the client: !command
On the server: command

Eg: ls command will list all files in the server

!ls command will list all files in the client

```
C:\Users\lenovo>ftp  
ftp> open 192.168.100.3  
Connected to 192.168.100.3.  
220 (vsFTPd 3.0.3)  
200 Always in UTF8 mode.  
User (192.168.100.3:(none)): durga  
331 Please specify the password.  
Password:  
230 Login successful.
```

ftp Commands:

```
ftp>ls → It display server side files  
ftp>!ls → It display client side files  
ftp>pwd → It display server side present working directory.  
ftp>!pwd → It display client side present working directory.  
ftp>cd dirname → To change to the specified directory on server machine.  
ftp>lcd dirname → To change to the specified directory on client machine.  
ftp>get filename → To download a file from the server.  
ftp>mget file1 file2 file3 → To download multiple files from the server.  
ftp>mget * → To download all files of current working directory from the server.  
ftp>mget *.log → To download all log files of current working directory from the server.
```



UNIX/LINUX



ftp>put filename → To upload file to the server.
ftp>mput file1 file2 file3 → To upload multiple files to the server.
ftp>? → To list all ftp commands
ftp>bye → To quit from ftp
ftp>hash → To on/off hash. If it is on then it will be off and if it is off then it will be on.
If it is on, then it shows status of download and upload.



Topic-52: Working with winscp

winscp meant for file transfer from windows to any unix based system.

scp → Means secure copy

How to download and Install:

<https://winscp.net/eng/download.php>

DOWNLOAD WINSCP 5.15.9 → WinSCP-5.15.9-Setup.exe

It is simply GUI Based application over FTP,SFTP for file transfer.

By using this we can transfer files from windows to linux system and from linux system to windows system.

Even we can get terminal, to execute some commands on the linux system.



Topic-53: Working with Filezilla

It is GUI Based Application over FTP. We can use this for file transfer from windows to linux and linux to windows.

How to download and Install:

<https://filezilla-project.org/index.php>

Download FileZilla Client for Windows (64bit)

FileZilla_3.46.3_win64_sponsored-setup.exe

Note: FileZilla won't be used commonly because it is based on FTP which is not secured.



Topic-54: init command

init is the parent of all processes with a PID "1".

```
durga@durga-VirtualBox:~$ ps -ef | head -5
```

UID	PID	PPID	C	STIME	TTY	TIME	CMD
root	1	0	0	21:06	?	00:00:01	/sbin/init splash
root	2	0	0	21:06	?	00:00:00	[kthreadd]
root	3	2	0	21:06	?	00:00:00	[rcu_gp]
root	4	2	0	21:06	?	00:00:00	[rcu_par_gp]

After loading Kernel into RAM, init will be started.
All remaining processes will be started after init process.

Runlevel of init Command:

Runlevels are state of init, where group of processes are defined at the time of OS startup.
The allowed runlevels are: 0 to 6

- 0 → System Halt or shutdown or poweroff the machine
- 1 → Single User { mostly root user for recover if system crashes }
- 2 → Multi user without NFS
- 3 → Multi user, command line (Mostly used for servers)
- 4 → Not used
- 5 → Multi user mode with GUI (used for Desktop)
- 6 → Reboot the OS

We can find current system runlevel by using runlevel command.
durga@durga-VirtualBox:~\$ runlevel
N 5

Note:

- \$ init 0 → To shutdown the system
- \$ init 6 → To reboot the system



Topic-55: TopMost important Linux Questions

Q1) How to check current operating system version?

By using uname command.

uname command can be used to print system information.

```
durga@durga-VirtualBox:~$ uname
Linux
durga@durga-VirtualBox:~$ uname -v
#24~18.04.1-Ubuntu SMP Mon Jul 29 16:12:28 UTC 2019
durga@durga-VirtualBox:~$ uname -a
Linux durga-VirtualBox 5.0.0-23-generic #24~18.04.1-Ubuntu SMP Mon
Jul 29 16:12:28 UTC 2019 x86_64 x86_64 x86_64 GNU/Linux
```

Various allowed options for uname command are:

- a, --all
print all information
- s, --kernel-name
print the kernel name
- p, --processor
print the processor type (non-portable)
- o, --operating-system
print the operating system
- version
output version information and exit

Q2) How to check current system's ipaddress?

From the terminal we can use any of the following commands

- 1) ifconfig
- 2) ip -a
- 3) ip -br a

```
durga@durga-VirtualBox:~$ ip -br a
lo                UNKNOWN      127.0.0.1/8 ::1/128
enp0s3            UP          192.168.100.4/24 fe80::3eda:ca33:380d:c44d/64
```



Q3) How to check available free disk space?

By using df command

```
durga@durga-VirtualBox:~$ df
Filesystem      1K-blocks    Used Available Use% Mounted on
udev            1404164         0   1404164   0% /dev
tmpfs           285608      1580    284028   1% /run
/dev/sda1       10253588 6158636   3554384  64% /
tmpfs           1428040         0   1428040   0% /dev/shm
tmpfs           5120         4      5116   1% /run/lock
tmpfs           1428040         0   1428040   0% /sys/fs/cgroup
/dev/loop0       90624     90624         0 100% /snap/core/7270
```

Q4) How to manage services in our system?

By using either service or systemctl.

A) To check status of a particular service:

\$ sudo service mysql status

```
durga@durga-VirtualBox:~$ sudo service mysql status
```

[sudo] password for durga:

- mysql.service - MySQL Community Server
Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
Active: active (running) since Sun 2020-01-12 13:07:49 IST; 9min ago

\$ sudo systemctl status mysql

```
durga@durga-VirtualBox:~$ sudo systemctl status mysql
```

- mysql.service - MySQL Community Server
Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
Active: active (running) since Sun 2020-01-12 13:07:49 IST; 10min ago

B) How to stop a particular service:

\$ sudo service mysql stop

\$ sudo systemctl stop mysql

```
durga@durga-VirtualBox:~$ sudo systemctl status mysql
```

- mysql.service - MySQL Community Server
Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
Active: inactive (dead) since Sun 2020-01-12 13:20:13 IST; 7s ago



C) How to start a particular service:

```
$ sudo service mysql start
$ sudo systemctl start mysql
```

```
durga@durga-VirtualBox:~$ sudo service mysql start
durga@durga-VirtualBox:~$ sudo systemctl status mysql
```

- mysql.service - MySQL Community Server
Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
Active: active (running) since Sun 2020-01-12 13:21:47 IST; 2s ago

D) How to restart a particular service:

```
$ sudo service mysql restart
$ sudo systemctl restart mysql
```

Q5) How to check the size of a particular file?

```
$ ls -s filename
$ ls -l filename
```

Q6) How to check the size of a directory?

By using du command

```
durga@durga-VirtualBox:~$ du -sh dir1
28K   dir1
```

Q7) How to check open ports available in linux?

```
$ sudo netstat -tulpn
durga@durga-VirtualBox:~$ sudo netstat -tulpn
Active Internet connections (only servers)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
PID/Program name					
tcp	0	0	127.0.0.1:3306	0.0.0.0:*	LISTEN
2261/mysqld					
tcp	0	0	127.0.0.53:53	0.0.0.0:*	LISTEN
423/systemd-resolve					
tcp	0	0	0.0.0.0:22	0.0.0.0:*	LISTEN
818/sshd					
tcp	0	0	127.0.0.1:631	0.0.0.0:*	LISTEN
581/cupsd					
tcp6	0	0	:::21	:::*	LISTEN
793/vsftpd					
tcp6	0	0	:::22	:::*	LISTEN
818/sshd					
tcp6	0	0	:::1:631	:::*	LISTEN
581/cupsd					
udp	0	0	127.0.0.53:53	0.0.0.0:*	
423/systemd-resolve					



```
udp      0      0 0.0.0.0:68          0.0.0.0:*
830/dhclient
udp      0      0 0.0.0.0:59952        0.0.0.0:*
589/avahi-daemon: r
udp      0      0 0.0.0.0:631          0.0.0.0:*
703/cups-browsed
udp      0      0 0.0.0.0:5353         0.0.0.0:*
589/avahi-daemon: r
udp6     0      0 :::46241             :::*
589/avahi-daemon: r
udp6     0      0 :::5353              :::*
589/avahi-daemon: r
```

Q8. How to check CPU Usage for a process?

By using either top or htop command we can get this information.

```
top - 13:35:29 up 28 min,  1 user,  load average: 0.02, 0.04, 0.09
Tasks: 228 total,   1 running, 190 sleeping,   0 stopped,   0 zombie
%Cpu(s):  3.6 us,  0.8 sy,  0.0 ni, 95.6 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 2856080 total,  835336 free, 1103628 used,  917116 buff/cache
KiB Swap: 483800 total,  483800 free,    0 used. 1574360 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1546	durga	20	0	3457844	263976	105168	S	5.0	9.2	0:37.42	gnome-shell
1353	durga	20	0	539524	85868	45024	S	3.3	3.0	0:16.08	Xorg
1888	durga	20	0	805412	38672	28744	S	2.6	1.4	0:09.40	gnome-terminal-
2324	durga	20	0	49008	3772	3080	R	1.0	0.1	0:00.35	top
1	root	20	0	225472	9444	6912	S	0.3	0.3	0:02.36	systemd
1483	durga	20	0	126384	2140	1772	S	0.3	0.1	0:03.38	VBoxClient
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0H-kb
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0.0	0.0	0:00.06	ksoftirqd/0

Q. Give an example for monitoring commands?

du,df,free for memory monitoring

top and htop mainly for cpu monitoring

Q. How to get complete information about a command?

By using man pages.

Q. How to check history of all commands used in our system?

By using history command



Q. How to check file type in Linux?

Extension is not applicable in linux. We can check file type by using file command.

```
durga@durga-VirtualBox:~$ file abc.txt
abc.txt: ASCII text
durga@durga-VirtualBox:~$ mv abc.txt abc.mp4
durga@durga-VirtualBox:~$ file abc.mp4
abc.mp4: ASCII text
```

Q. Explain differences between Linux and UNIX?

Q. Explain about Linux File System?

Q. Explain about components of Linux operating system?

Q. Why linux is more secured than other operating systems?

Linux follows 2 level security:

1. login
2. file permissions

Most of the users don't have root access. Hence they cannot copy new files in our directory. No chance of spreading viruses.

All linux activities are recorded in log files.

Iptables to manage incoming and outgoing traffic.

Q. Explain about grep command?

Q. What is the difference between root directory and / ?

/ is the root of linux file system.

Inside / there is a subdirectory named with root, which acts as superuser home directory

/home/durga → Durga user home directory

/root → Root user(super user) home directory

Q. Explain about types of users and types of permissions?

Q. What is the difference between more and less?

Both can be used to view large files page by page.

BY using more we can move only in forward direction. But by using less we can move either in the forward direction or in the backward direction.

Q. Explain difference between find and locate?



Q. How to get information about which command executing?

By using which command.

```
durga@durga-VirtualBox:~$ which cat
/bin/cat
durga@durga-VirtualBox:~$ which ls
/bin/ls
```

Q. What is the purpose of ping command?

It pings hosts and outputs the results.

```
durga@durga-VirtualBox:~$ ping www.google.com
PING www.google.com (216.58.200.132) 56(84) bytes of data.
64 bytes from maa05s10-in-f4.1e100.net (216.58.200.132): icmp_seq=1 ttl=56 time=21.5
ms
64 bytes from maa05s10-in-f4.1e100.net (216.58.200.132): icmp_seq=2 ttl=56 time=21.4
ms
```

Q. What is the purpose of whois command?

To get whois information of the given domain.