Linux with Shell Programming, SED, AWK And Many More

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

INDEX

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 hard ware 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:

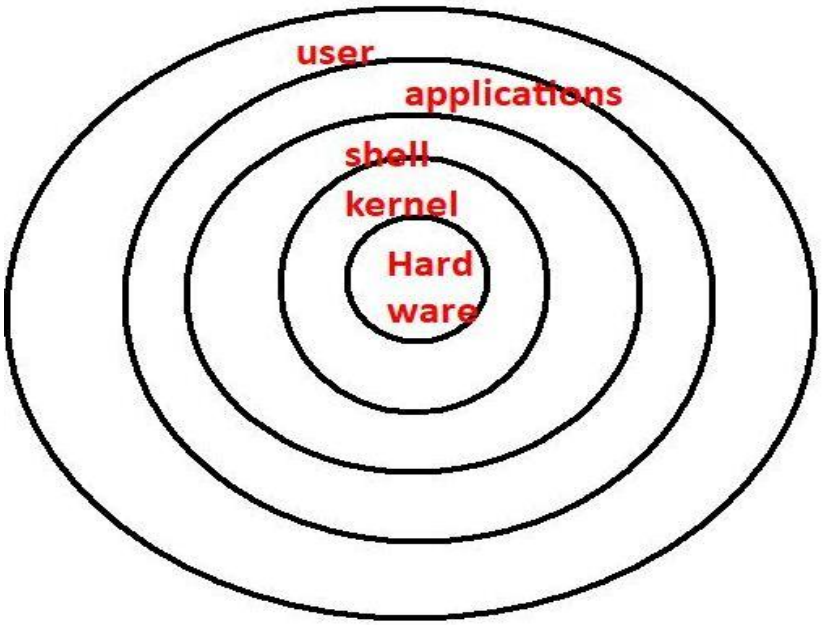
As UNIX is open source, multiple flavours are available with lot of extensions and improvements.

- Ubuntu
- RedHat
- Centos
- Fedora
- Slackware
- open solaries
- 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



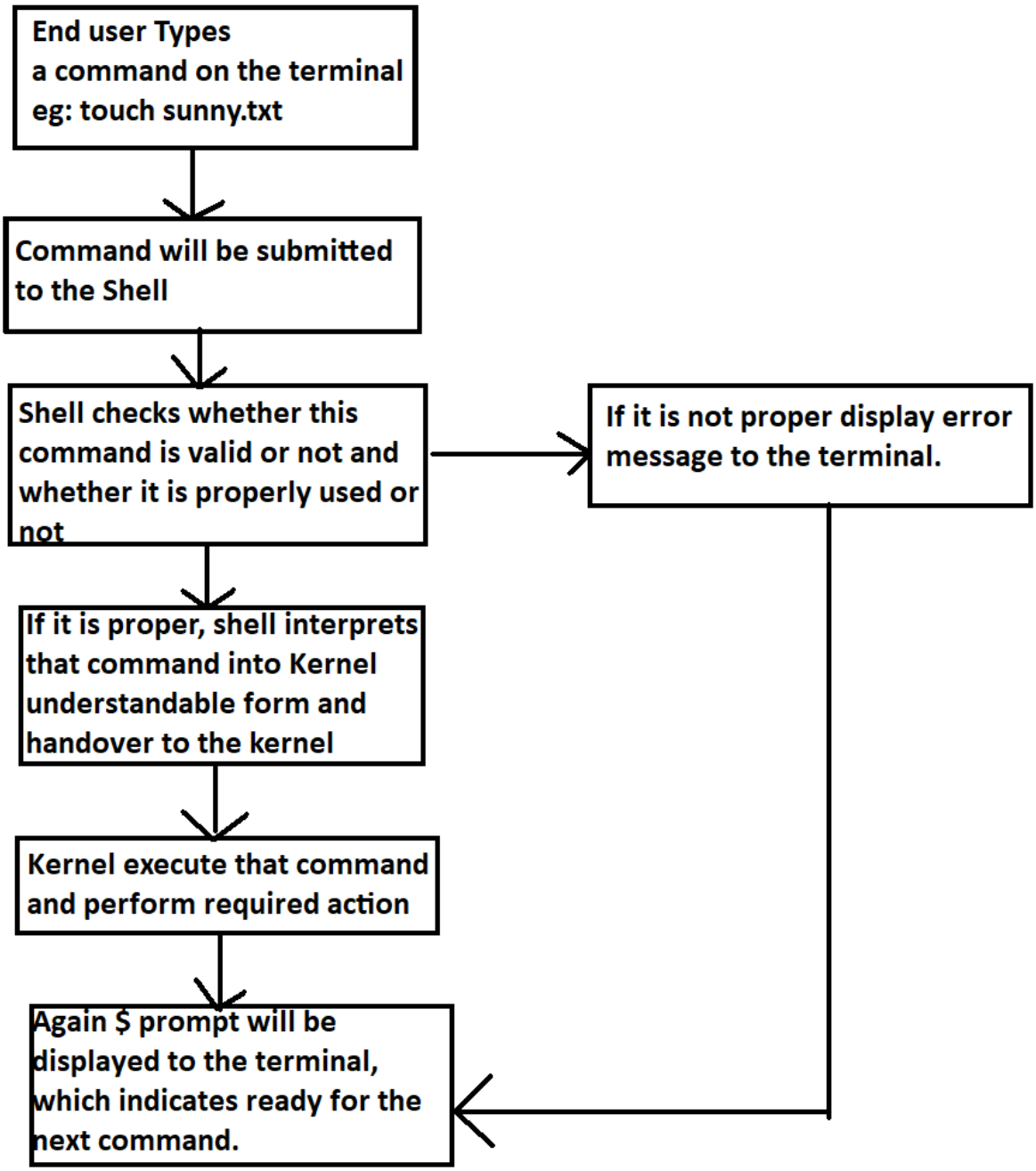
Shell:

- 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/translates that command into kernal understandable form.
3) Shell handovers that interpreted command to the kernal.
4) Kernal 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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

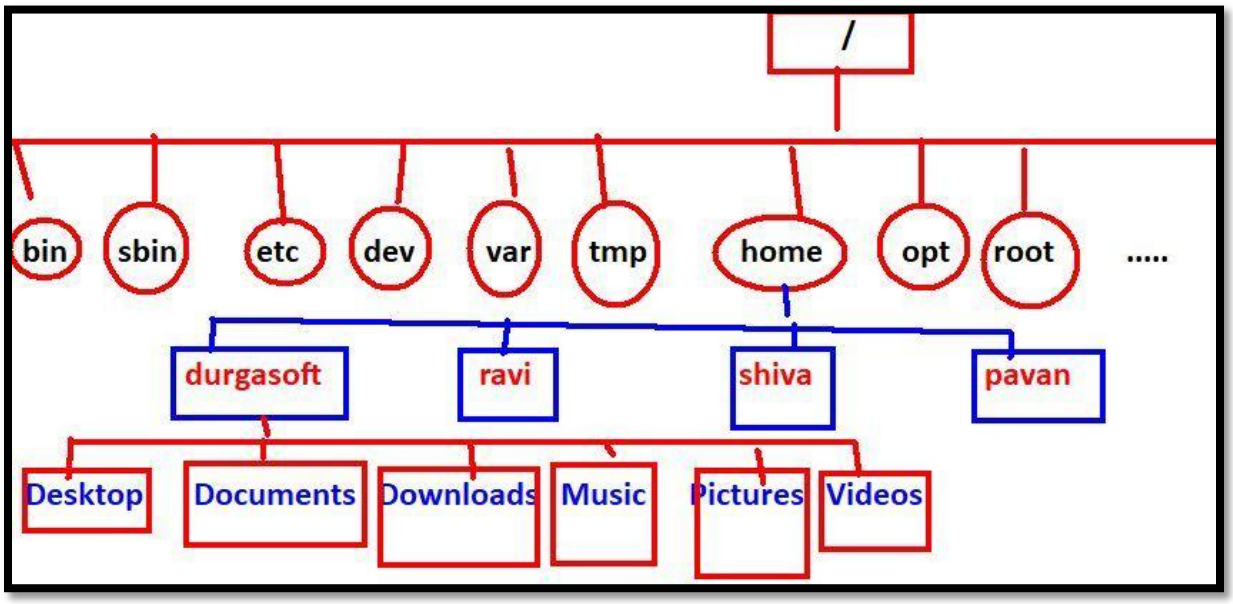
6) $ cd -
   - Means Previous Working Directory.
   It will Changes to Previous Working Directory.

```
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

Q1) 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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

Q2) 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 od this directory is same as PID.**

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Note: We can find all running proceses 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

Q3) 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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Topic-3: Linux Installation

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

1) Oracle Virtual Box Installation:

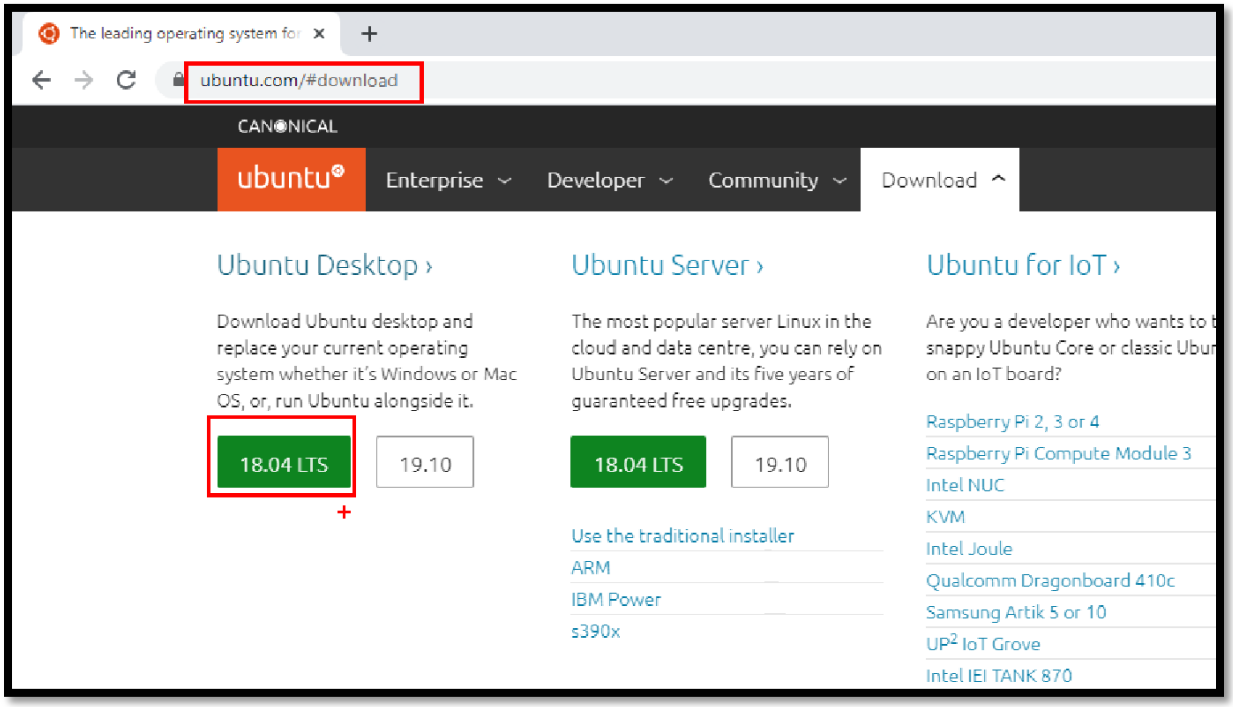
To run virtual computers in our system without effecting original computer, we should go for virtual box.
It can extends the capability of our existing computer so that we can run multiple operating systems simultaneously.

Donwload virtual box from:
virtualbox.org → Downloads → Windows hosts
VirtualBox-6.0.14-133895-Win.exe

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 decrese 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 listout 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 aplhabetical order.

3) ls | more

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

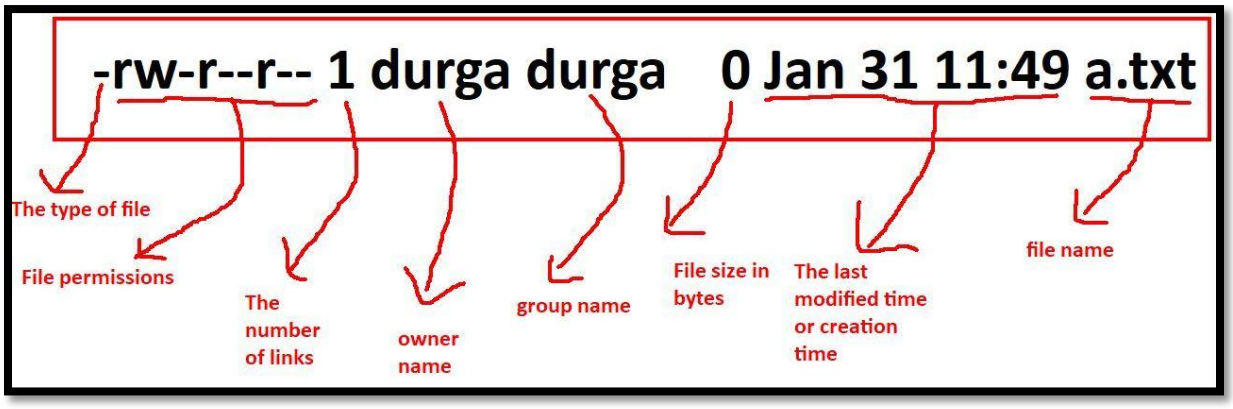
4) ls | pg

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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com



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


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 -i

- 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 time

    etc

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

Note: We can use these options simultaneously. When ever using options simultaneosuly 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 Modifiction 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  -atrisl
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) Is -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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

$ 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'
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 avilable 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 Advanatage 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?

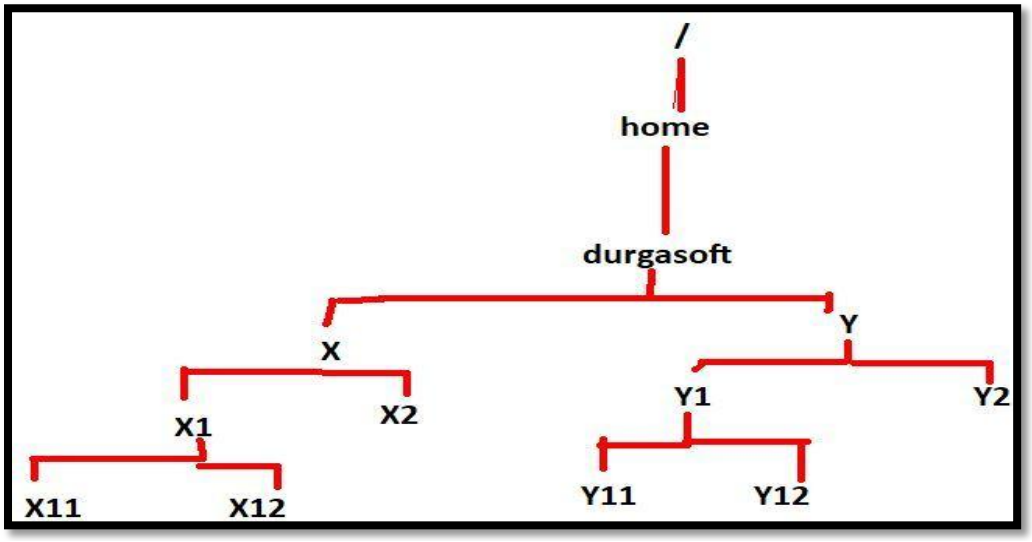
$ mkdir java classes

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}
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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*

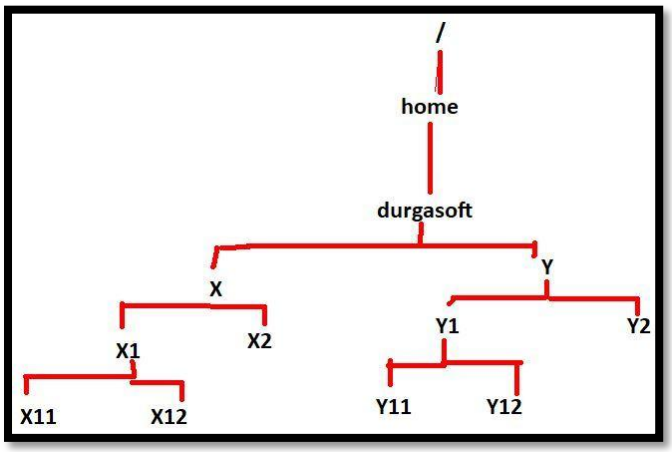
Q14) To Remove Directories where Name Starts with 'd' and Ends with 'n'

$ rmdir d*n

Q15) To Remove Directoreis 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 fo 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

Q16) 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

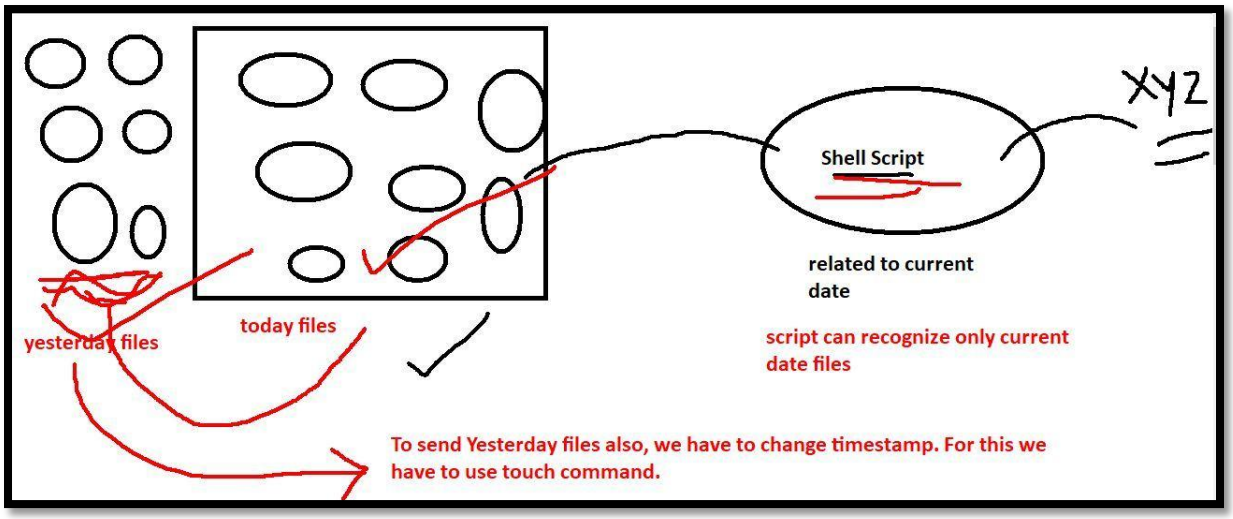
>> meant for appending/concatenation

Q3) If we are using Touch Comamnd, but the File is already available then what will happend?

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:



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 -n file1.txt
     1  This is first line
     2
     3  This is second line
     4
     5  This is third line
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
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/appending 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 -n +4 file1.txt
It will display from 4th line to last line

✻ 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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 kernal are hidden files.**
- **We can create hidden files just like normal files, only difference is file name should starts 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 contanis 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

3) sdiff Command:

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

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

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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 forcely.

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

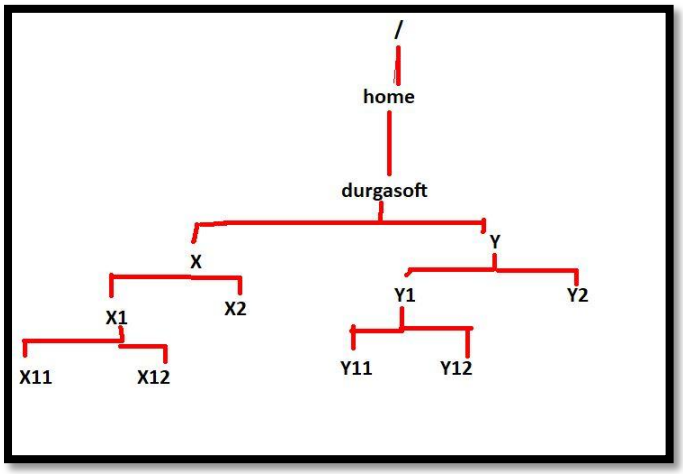
-1→ If we don't want to display column-1
-2 → If we don't want to display column-2
-3 → If we don't want to display column-3
-12 → If we don't want to display columns 1 and 2

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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

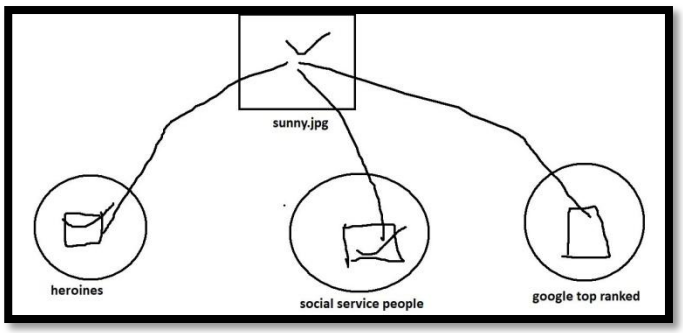
DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.**

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
ln: 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 butnot 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.

Q1) 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Q2) Which of the following is Valid Way to Create Soft Link for sunny.jpg Present in Pictures Directory?

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

Ans: B

Q3) 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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
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
```

Bydefault 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

Q1) 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.**

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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:

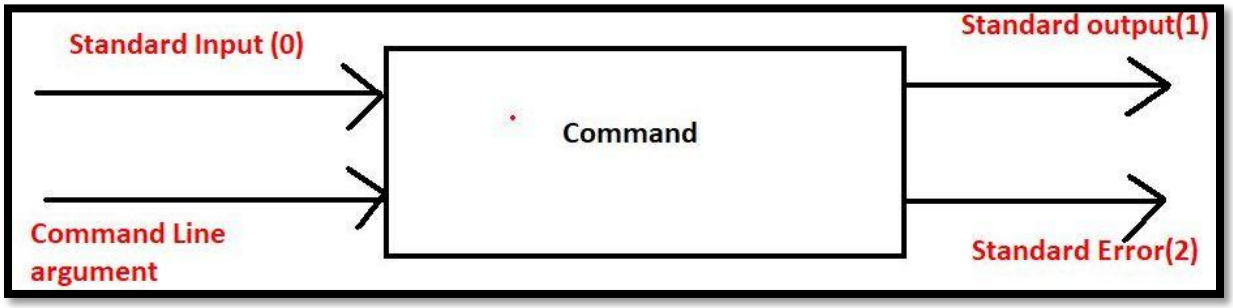
```
$ sort a.txt | uniq -i
Bunny
Chinny
sunny
$ sort a.txt | uniq -ic
    2 Bunny
    2 Chinny
    3 sunny
```

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

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


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

Bydefault 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 Ouput 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).

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

$ 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 produces 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 aceept command line arguments only.**
   echo "durgasoft"
3) **cat command can accept input either from standard input or from comamnd line arguments.**

Redirection

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

Redirecting Standard Ouput:

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 explicilty.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

$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 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.

termainal - 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

Q1) In How Many Ways Command can get Input?

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

Q2) Which of the following contains Data Streams?

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

Ans: A,B,C

Q3) By Default Standard Input connected to

A) Terminal
B) Keyboard
C) A File

Ans: B

Q4) By Default Standard Ouput connected to Terminal

Q5) By Default Standard Error connected to Terminal

Q6) What Number represents Standard Input Stream? 0

Q7) What Number represents Standard Output Stream? 1

Q8) What Number represents Standard Error? 2

Q9) 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Q10) 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

Q11) Explain the difference between <, >, >> in Redirection?

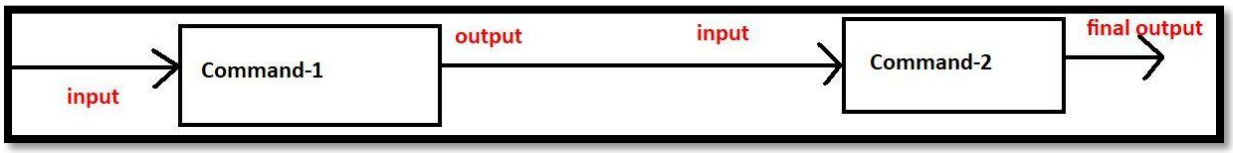
< symbol meant for input redirection

> 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:

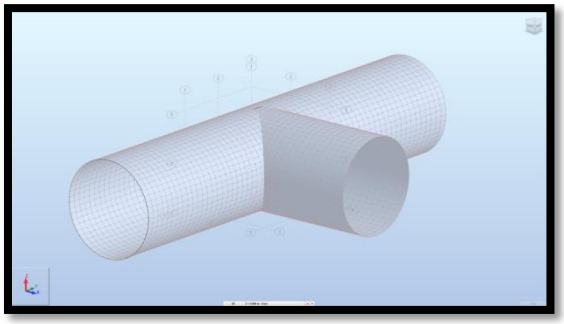
```
ls -l 1>output.txt | wc
```

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.

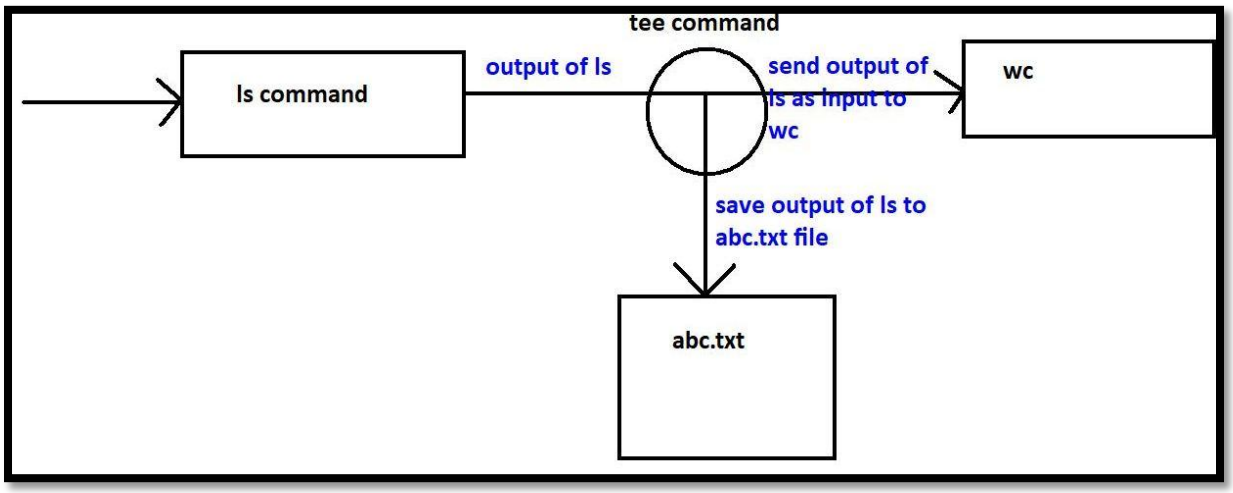
DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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



<u>Eg 1:</u> 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



<u>Eg 2:</u> 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

xargs Command:

Q1) 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
```

Q2) 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Q3) 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

Q4) 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

Q5) 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

1) To list out all files present in current working directory ➔ $ ls *

2) To list out all files with some extension ➔ $ ls *.*

3) To list out all files starts with a ➔ $ ls a*

4) To list out all files starts with a and ends with t ➔ $ ls a*t

5) To list out all .java files ➔ $ ls *.java

6) To list out all files where file name contains only 2 characters and first character should be 'a' ➔ $ ls a?

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 aphabet symbol
$ ls [a-z]* OR $ls [[:lower:]]*

12) To list out all files starts with upper case aphabet 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 3 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 3rd 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Topic-17: Command Aliasing

Alias means other alternative name or nickname.
We can give our own more convinient 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 bydefault 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.

Q1) What is the Purpose of Alias Comamnd?

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

Q2) How to Use Unalias Command?

unalias alias_name → To remove a particular alias
unalias –a → To remove all aliases

Q3) To Persist Aliases, in which File we have to define Aliases?

.bashrc OR .bash_aliases

Q4) 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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Reason:

locate command using database to find results. This database will be updated only once per day bydefault. 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

And: G

Q2) Locate Command internally Uses Database to find Results?

True
False

Ans: True

Q3) 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

Q4) 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
        |-levlel_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

Q1) 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

Q2) 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.

Q3) 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 {}  \;
```

Q4) The Find Command Uses a Database to Search Files and Directories

A) True
B) False

Ans: B

Q5) 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

Q6) 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

Q7) 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Q8) To find all Files and Directories inside /dev Directory upto Maximum of 3 Levels Deep and Size is Greater than 200 Kilo Bytes?

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

Q9) Which of the following Command will find all Files below our Home Directory where File Size is Greater than 3 Mega Bytes 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 bydefault.
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 advanatages 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**

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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:

Bydefault grep command will consider case. If we want to ignore case then we should use 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```

Q1) 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][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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

Q2) 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.

Q3) 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
kljdslgklfjd
dsklfjakdlsfjkl
sdlkfskfjdklsjfdkl
```

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

Q4) 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
```

Q5) Write Commands to Count the Number of Files Present in the Current Working Directory?

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


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Q6) Write Commands to Count the Number of Link Files Present in the /etc Directory?

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

Q7) 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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 deverlopers 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

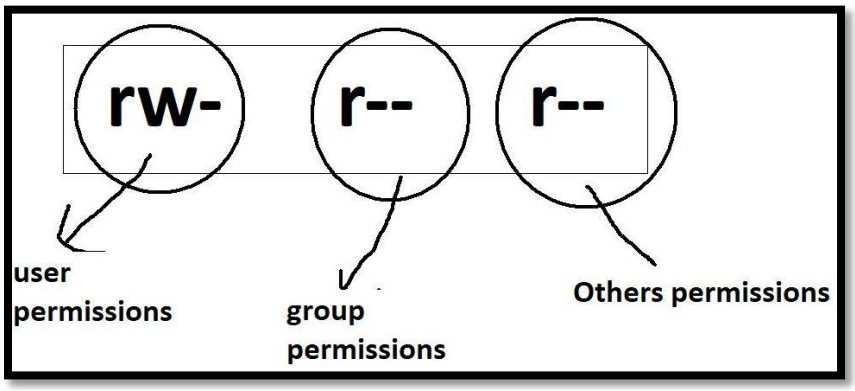
Note: Only owner and super user (root) can change file permissions.

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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
--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)

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```

Q1) 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.txtowner: 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.
duragsoft 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Topic-23: Working with Editors

We can use editors to edit file content.
There are multiple editors are available.

1) **geditor**
2) **vi editor**
3) **nano editor**
   etc

1) working with geditor:

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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 aleady 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Part-2

Shell Scripting

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

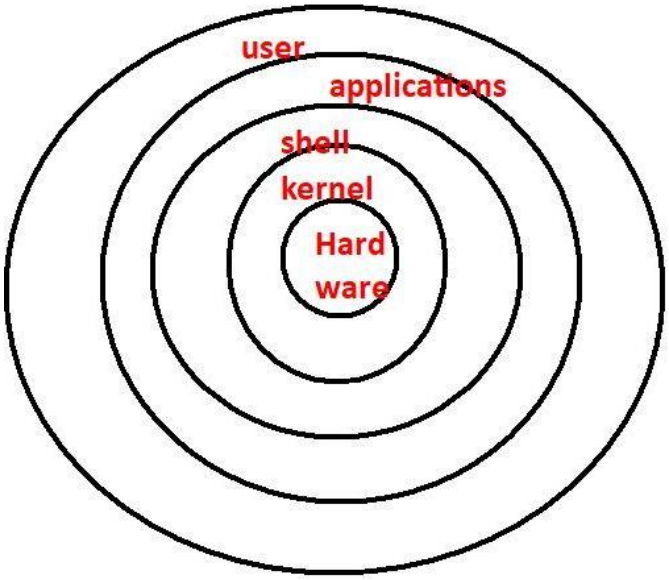
Topic-35: Stream Editor(SED)

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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

Q1) 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
```

Q2) 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.

Q3) 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
```

Q4) How to Run Our Script from any where 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**

Q5) 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**

Q6) 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.**

.bashrac

```
....
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
```

Q1) 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Note: If we use single quotes then variable substitution won't be happend. 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 butnot 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 scritp 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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  adsfkjshfdjkhsfkjhkjsfhk
The Length of given string adsfkjshfdjkhsfkjhkjsfhk : 24
```

Q1) 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
```

Q2) 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
```

Q3) 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
```

Q4) 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
```

Q5) Write a Script to Read File Name 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
----------------------------------------------
```

Q6) Write a Script to Read File Name 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"
```

Q7) Write a Script to Read File Name 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
- → Substraction
* → 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 []

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```

Q1) Write a Script to Read 2 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"
```

Q2) 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
```

Q3) 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

Q1) 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
else
   default action
fi
```

Q2) Write a Script to find Greater of 2 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
```

Q3) Write a Script to Check whether Numbers OR 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
```

Q4) 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/scrit not executed successfully.

durgasoft@durgasoft:~/scripts$ echo "Hello Friends"
Hello Friends
durgasoft@durgasoft:~/scripts$ echo "$?"
0
```

```
durgasoft@durgasoft:~/scripts$ rm sdfjkslajfdksajfjdsakjfdksajfkd
rm: cannot remove 'sdfjkslajfdksajfjdsakjfdksajfkd': 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
....

Q5) Write a Script that takes 2 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
```

Q6) 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.

Q7) 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
```

Q8) 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-dgit 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-dgit 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

-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
```

Q9) Write a Script that Reads File Name 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
```

Q10) 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.

Q11) 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
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
```

Q12) Write a Scirpt 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 has read and execute permissions on source directory
5) The user should has write and execute permissions on destination directory
6) All error messages should be send 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
```

Q13) Write a Script that Checks whether Login User is Root OR not. If Login User is Root then Display 1st 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 5 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]
```

Q14) 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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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"
      ;;
  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
```

Q15) 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"
         ;;
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
"RC")
        echo "It is too light"
        ;;
"FO")
        echo "Buy one get one FREE"
        ;;
*)
        echo "Other brands are not recommended"
esac
```

Q16) 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Q17) Write a Script that Accepts a Single Character and Checks whether it is Digit OR Special Character OR Vowel OR Consonent?

```
#! /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 Consonent"
         ;;
   *)
       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 Consonent
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
```

Q18) 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
```

Q19) Write a Script that Reads 2 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**

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```

Q4) 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$(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\t\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.

Q5) 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
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
```

Q6) 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.

Q7) 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```

Q8) 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
```

Q9) 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
```

Q10) 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"
```

Q11) 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 Employeed 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 Employeed 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)"
```

Q12) Write a Script to Display Multiple Files Content to the Terminal and all File Names 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
```

Q13) 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
```

Alternative Syntax of for Loop (Advanced for Loop):

Old Style of for Loop:

```
for variable in item1 item2 ... itemN
do
  body
done
```

Advanced for Loop:

```
for ((i=1; i<10; i++))
do
  body
done
```

Q14) Write a Script to Print Numbers from 0 to 4 by using advanced for Loop?

```
#! /bin/bash
for ((i=0;i<5;i++))
do
  echo $i
done
```

Output:

```
0
1
2
3
4
```

Q15) Write a Script to Print Numbers for Count Down from provided Number to 1 by using advanced for Loop?

```
#! /bin/bash
read -p "Enter n value:" n
for((i=n,i>=1;i--)
do
  echo $i
  sleep 1.5
done
```

Q16) Write a Script to Display n^{th} Table?

```
#! /bin/bash
read -p "Enter n value:" n
for ((i=1; i<=10; i++))
do
  echo "$n * $i = $[n*i]"
done
```

```
durga@durga-VirtualBox:~/scripts$ test.sh
Enter n value:9
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
9 * 10 = 90
```

Q17) Write a Script to generate Hotel Bill based on Customer selected Items. The Items and Price Information is as follows

A → Vadapov (Each Plate Rs 30 /-)
B → Dosa (Each Plate Rs 50 /-)
C → Poori (Each Plate Rs 40 /-)
D → Idli (Each Plate Rs 25 /-)

```
#! /bin/bash
echo "Welcome to DURGA HOTEL"
echo
amount=0
while [ true ]
do
 echo "Menu Items:"
 echo "....................."
 echo "A --->Vadapov (Each Plate Rs 30 /-)"
 echo "B --->Dosa (Each Plate Rs 50 /-)"
 echo "C --->Poori (Each Plate Rs 40 /-)"
 echo "D --->Idli (Each Plate Rs 25 /-)"
 read -p "Choose Your Required Item A|B|C|D:" item
 case $item in
   A)
    read -p "Enter the number of plates of Vadapov, you required:" quantity
    amount=$[amount+quantity*30]
    ;;
   B)
    read -p "Enter the number of plates of Dosa, you required:" quantity
    amount=$[amount+quantity*50]
    ;;
   C)
    read -p "Enter the number of plates of Poori, you required:" quantity
    amount=$[amount+quantity*40]
    ;;
   D)
    read -p "Enter the number of plates of Idli, you required:" quantity
    amount=$[amount+quantity*25]
    ;;
   *)
    echo "You entered invalid option. Choose Again"
    continue
  esac
  read -p "Do you want to order any other item[Yes|No]:" option
  case $option in
```

```
        [Yy]|[Yy][Ee][Ss])
                continue
                        ;;
        [Nn]|[Nn][Oo])
                break
                        ;;
  esac
done
echo
echo "Your Total Bill Amount: Rs $amount/-"
echo "Thanks for visiting DURGA HOTEL"

durga@durga-VirtualBox:~/scripts$ test.sh
Welcome to DURGA HOTEL

Menu Items:
……….……
A → Vadapov (Each Plate Rs 30 /-)
B → Dosa (Each Plate Rs 50 /-)
C → Poori (Each Plate Rs 40 /-)
D → Idli (Each Plate Rs 25 /-)
Choose Your Required Item A|B|C|D:D
Enter the number of plates of Idli, you required:4
Do you want to order any other item[Yes|No]:yes

Menu Items:
……….……
A → Vadapov (Each Plate Rs 30 /-)
B → Dosa (Each Plate Rs 50 /-)
C → Poori (Each Plate Rs 40 /-)
D → Idli (Each Plate Rs 25 /-)
Choose Your Required Item A|B|C|D:C
Enter the number of plates of Poori, you required:3
Do you want to order any other item[Yes|No]:Yes

Menu Items:
……….……
A → Vadapov (Each Plate Rs 30 /-)
B → Dosa (Each Plate Rs 50 /-)
C → Poori (Each Plate Rs 40 /-)
D → Idli (Each Plate Rs 25 /-)
Choose Your Required Item A|B|C|D:B
Enter the number of plates of Dosa, you required:2
Do you want to order any other item[Yes|No]:Yes
```

Menu Items:

……..
A → Vadapov (Each Plate Rs 30 /-)
B → Dosa (Each Plate Rs 50 /-)
C → Poori (Each Plate Rs 40 /-)
D → Idli (Each Plate Rs 25 /-)
Choose Your Required Item A|B|C|D:A
Enter the number of plates of Vadapov, you required:1
Do you want to order any other item[Yes|No]:No

Your Total Bill Amount: Rs 350/-
Thanks for visiting DURGA HOTEL

Q18) Write a Script to Test whether the given Number is Prime Number OR not?

If any number has two factors (1 and itself), such type of number is said to be prime number.

2 → 1, 2
3 → 1, 3
5 → 1, 5

```
#! /bin/bash
read -p "Enter Any Number to test whether it is prime or not:" n

if [ $n -le 1 ]; then
  echo "$n is not a PRIME number"
  exit 1
fi

for ((i=2;i<=n/2;i++))
do
 if [ $[n%i] -eq 0 ]; then
   echo "$n is not PRIME number"
   exit 1
 fi
done
echo "$n is a PRIME number"
```

```
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any Number to test whether it is prime or not:23
23 is PRIME number
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any Number to test whether it is prime or not:29
29 is PRIME number
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any Number to test whether it is prime or not:35
35 is not PRIME number
```

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 biginning:
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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
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
```

Q1) Write a Script to Create an Array with some Elements and Print all Elements by using while Loop, for Loop and advanced for Loop?

```
#! /bin/bash

declare -a fruits
fruits=("Apple" "Orange" "Banana" "Mango")
size=${#fruits[@]}
i=0

echo "All elements by using while loop:"
while [ $i -lt $size ]
do
  echo ${fruits[$i]}
  let i++
done

echo "All elements by using for loop:"
for fruit in ${fruits[@]}
do
   echo $fruit
done

echo "All elements by using advanced for loop:"
for (( i=0; i < ${#fruits[@]}; i++ ))
```

```
do
  echo ${fruits[$i]}
done
```

Output:

```
durga@durga-VirtualBox:~/scripts$ test.sh
All elements by using while loop:
Apple
Orange
Banana
Mango
All elements by using for loop:
Apple
Orange
Banana
Mango
All elements by using advanced for loop:
Apple
Orange
Banana
Mango
```

Note:

1) If we create an array with elements directly
   fruits=("Apple" "Orange" "Banana" "Mango")
   then the indices will be 0,1,2,3 etc

2) After creating array we can add extra elements also
   fruits=("Apple" "Orange" "Banana" "Mango")
   fruits[4]="Sapota"

Q2) Write a Script for accessing Array Elements by using for Loop if Indices are Random?

```
#! /bin/bash
declare -a fruits
fruits[10]="Apple"
fruits[20]="Banana"
fruits[30]="Orange"
fruits[40]="Mango"

echo "Accessing based on Values:"
for fruit in ${fruits[@]}
do
```

```
 echo $fruit
done
echo
echo "Accessing based on Index:"
for index in ${!fruits[@]}
do
 echo ${fruits[$index]}
done
```

Output:

```
durga@durga-VirtualBox:~/scripts$ test.sh
Accessing based on Values:
Apple
Banana
Orange
Mango

Accessing based on Index:
Apple
Banana
Orange
Mango
```

Q3) Is it Possible to Remove Array Elements?

Yes possible by using unset command.

Eg:

```
#! /bin/bash
declare -a fruits
fruits[10]="Apple"
fruits[20]="Banana"
fruits[30]="Orange"
fruits[40]="Mango"

echo "All Array Elements Before Removal: ${fruits[@]}"
unset fruits[20]
unset fruits[40]
echo "All Array Elements After Removal: ${fruits[@]}"
```

Output:

```
durga@durga-VirtualBox:~/scripts$ test.sh
All Array Elements Before Removal: Apple Banana Orange Mango
All Array Elements After Removal: Apple Orange
```

Q4) Write a Script to Store given n Numbers in to an Array?

```
#! /bin/bash
read -p "Enter The Number Of values:" n

for ((i=0,j=1;i<n;i++))
do
  read -p "Enter The Number-$[j++]:" NUM[${i}]
done
```

Output:

```
durga@durga-VirtualBox:~/scripts$ test.sh
Enter The Number Of values:3
Enter The Number-1:10
Enter The Number-2:20
Enter The Number-3:30
```

Q5) Write a Script to Read n Numbers and Store inside Array. Print the Sum of Even Numbers and Odd Numbers separately?

```
#! /bin/bash
read -p "Enter Number of Values:" n

for((i=0,j=1;i<n;i++))
do
  read -p "Enter The Number-$[j++]:" NUM[$i]
done

esum=0
osum=0
for val in ${NUM[@]}
do
  if [ $[val%2] -eq 0 ]; then
    let esum=esum+val
  else
    let osum=osum+val
  fi
done
echo "The Sum of Even Numbers: $esum"
echo "The Sum of Odd Numbers: $osum"
```

Output:

```
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Number of Values:5
Enter The Number-1:10
Enter The Number-2:12
Enter The Number-3:13
Enter The Number-4:15
Enter The Number-5:20
The Sum of Even Numbers: 42
The Sum of Odd Numbers: 28
```

Q6) Write a Script to Store all .txt File Names Present in Current Working Directory in to an Array and Print Permissions of every File

```
#! /bin/bash
files=($(ls *.txt))
for fname in ${files[@]}
do
  echo -ne "$fname:\t"
  if [ -r $fname ]; then
    echo -ne "READ(Y)\t"
  else
    echo -ne "READ(N)\t"
  fi
  if [ -w $fname ]; then
    echo -ne "WRITE(Y)\t"
  else
    echo -ne "WRITE(N)\t"
  fi
  if [ -x $fname ]; then
    echo  "EXECUTE(Y)"
  else
    echo  "EXECUTE(N)"
  fi
done
```

Output:

```
durga@durga-VirtualBox:~/scripts$ test.sh
abcd.txt:       READ(Y)       WRITE(Y)      EXECUTE(N)
abc.txt:        READ(Y)       WRITE(Y)      EXECUTE(N)
a.txt:          READ(Y)       WRITE(Y)      EXECUTE(N)
b.txt:          READ(Y)       WRITE(Y)      EXECUTE(N)
c.txt:          READ(Y)       WRITE(Y)      EXECUTE(N)
emp.txt:        READ(Y)       WRITE(Y)      EXECUTE(N)
hyd.txt:        READ(Y)       WRITE(Y)      EXECUTE(N)
output.txt:     READ(Y)       WRITE(Y)      EXECUTE(N)
result.txt:     READ(Y)       WRITE(Y)      EXECUTE(N)
z.txt:          READ(N)       WRITE(N)      EXECUTE(N)
```

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
}
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

How to call a Function:

function_name param1 param2 param3 ...

Q1) 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. With in the function we can access parameters as follows:

$1 → First Parameter
$2 → Second Parameter
$@ → All Paramters
$* → All parameters
$# → Total number of parameters
$0 → It is script name but not function name

Q2) Write a Function to demonstrate how to access Function Parameters?

```
#! /bin/bash
function demo()
{
  echo "First Parameter: $1"
  echo "Second Parameter: $2"
  echo "Third Parameter: $3"
  echo "Total Number of Paramerters: $#"
  echo "All Parameters with @: $@"
  echo "All Parameters with *: $*"
  echo "Script Name:$0"
}

demo 10 20 30 40 50
```

Output:

```
First Parameter: 10
Second Parameter: 20
Third Parameter: 30
Total Number of Paramerters: 5
All Parameters with @: 10 20 30 40 50
All Parameters with *: 10 20 30 40 50
Script Name:/home/durga/scripts/test.sh
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```

Q3) Write a Function that takes 2 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
```

Q4) 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Q5) Write a Function to find Maximum of 2 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
```

Q6) Write a Function to find Maximum of 3 given Integer Numbers?

```
#! /bin/bash
max()
{
  a=$1
  b=$2
  c=$3
  if [ $a -gt $b -a $a -gt $c  ]; then
    echo "Biggest Number:$a"
  elif [ $b -gt $c ]; then
    echo "Biggest Number:$b"
  else
    echo "Biggest Number:$c"
   fi
}
read -p "Enter First Number:" n1
read -p "Enter Second Number:" n2
read -p "Enter Third Number:" n3
max  $n1 $n2 $n3
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Q7) 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
```

Q8) Write a Function to find Factorial of a 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
}
```

Q9) Write a Function to find Factorial of 1st 10 Natural Numbers?

```
#! /bin/bash
factorial()
{
 original=$1
 n=$1
 fact=1
 while [ $n -gt 1 ]
 do
  let fact=fact*n
  let n--
 done
 echo "The Factorial of $original: $fact"
}
for i in {1..10}
do
 factorial $i
done

durga@durga-VirtualBox:~/scripts$ test.sh
The Factorial of 1: 1
The Factorial of 2: 2
The Factorial of 3: 6
The Factorial of 4: 24
The Factorial of 5: 120
The Factorial of 6: 720
The Factorial of 7: 5040
```

```
The Factorial of 8: 40320
The Factorial of 9: 362880
The Factorial of 10: 3628800
```

Q10) Write a Function to Check whether the given Number is Even Number OR not?

```
#! /bin/bash
even_odd()
{
  n=$1
  if [ $[n%2] -eq 0 ]; then
     echo "$n is an Even Number"
  else
     echo "$n is an Odd Number"
  fi
}

read -p "Enter Any Number to test whether it is Even or Odd:" n
even_odd  $n
```

```
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any Number to test whether it is Even or Odd:10
10 is an Even Number
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any Number to test whether it is Even or Odd:13
13 is an Odd Number
```

Q11) Write a Function to Test whether the given Number is Positive OR Negative Number?

```
#! /bin/bash
positive_negative()
{
   n=$1
   if [ $n -gt 0 ]; then
      echo "$n is a Positive Number"
   elif [ $n -lt 0 ]; then
      echo "$n is a Negative Number"
   else
      echo "It is just Zero"
   fi
}
read -p "Enter Any number to test whether it is positive or negative:" n
```

```
positive_negative  $n

durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any number to test whether it is positive or negative:10
10 is a Positive Number
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any number to test whether it is positive or negative:-20
-20 is a Negative Number
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any number to test whether it is positive or negative:0
It is just Zero
```

Prime Numbers:

Any positive number greater than 1, which has no other factors except 1 and the number itself, is called prime number.

Eg: 2, 3, 5, 7, 11, 13 etc

Any Positive Number which has more than 2 factors, such type of numbers are called Composite Numbers.

Eg: 4, 6, 8, 12 etc

1 is neither prime nor composite. It is a unique number.

Q12) Write a Function to Check whether the given Number is Prime Number OR not?

```
#! /bin/bash
prime_check()
{
  n=$1
  if [ $n -le 1 ]; then
    echo "$n is not a PRIME number"
  else
    is_prime="yes"
    for ((i=2;i<n;i++))
    do
     if [ $[n%i] -eq 0 ]; then
          echo "$n is not a PRIME Number"
          is_prime="no"
          break
        fi
```

```
    done
    if [ $is_prime = "yes" ]; then
      echo "$n is a PRIME Number"
    fi
   fi
}
read -p "Enter Any Number to test whether it is PRIME or not:" n
prime_check $n

durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any Number to test whether it is PRIME or not:129
129 is not a PRIME Number
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any Number to test whether it is PRIME or not:127
127 is a PRIME Number
durga@durga-VirtualBox:~/scripts$ test.sh
Enter Any Number to test whether it is PRIME or not:19
19 is a PRIME Number
durga@durga-VirtualBox:~/script
```

Q13) Write a Function to generate Prime Numbers which are Less than OR Equal to given Number?

```
#! /bin/bash
prime_numbers_generator()
{
  n=$1
  for((n1=2;n1<=n;n1++))  //Repeat from 2 to n every number to test
  do
    is_prime="yes"
    for((i=2;i<n1;i++)) //To test current number n1 is prime or not
    do
      if [ $[n1%i] -eq 0 ]; then
        is_prime="no"
            break
      fi
    done
    if [ $is_prime = yes ]; then
      echo $n1
    fi
  done
 }
read -p "Enter N value:" n
prime_numbers_generator $n
```

```
durga@durga-VirtualBox:~/scripts$ test.sh
Enter N value:20
2
3
5
7
11
13
17
19
```

Q14) Write a Function to generate 1st n Prime Numbers?

```
#! /bin/bash
prime_numbers_generator()
{
 n=$1
 count=0
 for((n1=2;n1>=2;n1++)) # n1=2,3,4,5,6,7,8,....
 do
  is_prime=yes
  for((i=2;i<n1;i++))
  do
   if [ $[n1%i] -eq 0 ]; then
     is_prime=no
        break
   fi
  done
  if [ $is_prime = yes ]; then
    echo $n1
    let count++
  fi
  if [ $count -eq $n ]; then
    break
  fi
 done
}
read -p "Enter n value:" n
prime_numbers_generator $n

durga@durga-VirtualBox:~/scripts$ test.sh
Enter n value:1
2
durga@durga-VirtualBox:~/scripts$ test.sh
```

```
Enter n value:2
2
3
durga@durga-VirtualBox:~/scripts$ test.sh
Enter n value:5
2
3
5
7
11
```

Variable Scope:

Bydefault 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Use Case:

```
backup()
{
   commands to take backup
}
backup
if [ $? != 0 ]; then
  something goes wrong backup failed
else
  backup successful
fi
```

break vs exit vs return:

1) break:

We can use break only inside loops to break loop exeuction.we will come out of the loop and remaining statements after loop will be executed.

2) exit:

We can use anywhere exit statement to terminate script execution.The script will be terminated. No chance of executing any other statement.

3) return:

We can use return statement only inside function to stop function execution. After return statement, the remaining statements inside function won't be exeuction. But after function call the remaining statements will be executed.

How to Call Functions Present in another Script:

util.sh

```
#! /bin/bash
x=888
y=999
add()
{
  echo "$1 + $2 = $[$1+$2]"
}
multiply()
{
  echo "$1 * $2 = $[$1*$2]"
}
subtract()
```

```
{
  echo "$1 - $2 = $[$1-$2]"
}
divide()
{
  echo "$1 / $2 = $[$1/$2]"
}
```

test.sh

```
#! /bin/bash
.  ./util.sh #This is just inclusion and we are not executing util.sh
add 10 20
subtract 20 10
multiply 10 20
divide 20 10
echo "The value of x:$x"
echo "The value of y:$y"

durga@durga-VirtualBox:~/scripts$ test.sh
10 + 20 = 30
20 - 10 = 10
10 * 20 = 200
20 / 10 = 2
The value of x:888
The value of y:999
```

Topic-34

Shell Script Projects

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Project 1: Secret Agent Application

Rules:

1) The first character of the Name should be 'd'
2) The Last character of Favourite Actor should be 'r'
3) The Lucky Number should be 7
4) The number of characters in his favourite dish should be >=6.

If the above conditions are satified then user is valid secret agent and share information about our next operation, otherwise just send thanks message.

Version-1 (Performance wise not upto the Mark)

```
1)  #! /bin/bash
2)  echo "Welcome to Secret Agent Application"
3)  echo "#####################################"
4)
5)  read -p "Enter Your Name:" name
6)  read -p "Enter Your Favourite Actor:" actor
7)  read -p "Enter Your Lucky Number:" lucky
8)  read -p "Enter Your Favourite Dish:" dish
9)
10) first_char_name=$(echo -n $name | cut -c 1)
11) len=$(echo -n $actor | wc -c)
12) last_char_actor=$(echo -n $actor | cut -c $len)
13) no_of_characters_dish=$(echo -n $dish | wc -c)
14) if [ $first_char_name = "d" -a $last_char_actor = "r" -a $lucky -eq 7 -a
    $no_of_characters_dish -ge 6 ]; then
15)   echo "Hello Secret Agent...Our Next Operation is"
16)   echo "We have to kill atleast 10 sleeping students of the class room because they
    are burden to country"
17) else
18)   echo "Hello $name, thanks for your information"
19) fi
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Version-2 (Improved Performance wise)

```
1)  #! /bin/bash
2)  echo "Welcome to Secret Agent Application"
3)  echo "#####################################"
4)
5)  read -p "Enter Your Name:" name
6)  first_char_name=$(echo -n $name | cut -c 1)
7)  if [ $first_char_name != "d" ]; then
8)     echo "Hello $name, thanks for your information"
9)     exit 1
10) fi
11)
12) read -p "Enter Your Favourite Actor:" actor
13) len=$(echo -n $actor | wc -c)
14) last_char_actor=$(echo -n $actor | cut -c $len)
15) if [ $last_char_actor != "r" ]; then
16)    echo "Hello $name, thanks for your information"
17)    exit 1
18) fi
19)
20) read -p "Enter Your Lucky Number:" lucky
21) if [ $lucky -ne 7 ]; then
22) echo "Hello $name, thanks for your information"
23)    exit 1
24) fi
25)
26) read -p "Enter Your Favourite Dish:" dish
27) no_of_characters_dish=$(echo -n $dish | wc -c)
28) if [ $no_of_characters_dish -lt 6 ]; then
29) echo "Hello $name, thanks for your information"
30)    exit 1
31) fi
32) echo "Hello Secret Agent...Our Next Operation is"
33) echo "We have to kill atleast 10 sleeping students of the class room because they a
    re burden to country"
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Output:

```
durga@durga-VirtualBox:~/scripts$ test.sh
Welcome to Secret Agent Application
#####################################
Enter Your Name:durga
Enter Your Favourite Actor:Ameer
Enter Your Lucky Number:7
Enter Your Favourite Dish:Fish
Hello durga, thanks for your information
durga@durga-VirtualBox:~/scripts$ test.sh
Welcome to Secret Agent Application
#####################################
Enter Your Name:durga
Enter Your Favourite Actor:Ameer
Enter Your Lucky Number:7
Enter Your Favourite Dish:Mutton
Hello Secret Agent...Our Next Operation is
We have to kill atleast 10 sleeping students of the class room because they are burden to
country
```

Project 2: Book Rental Application

The DURGASOFT Book Company needs a way to determine the cost that a student has to pay for renting a Book.

The cost is dependent on the time of the Book is returned.
However there are also special rates on Saturday and Sundays.
The Fee Structure is shown in the following list:

The cost is Rs 30 /- Per Day.
If the Book is returned after 9 PM,the student will be charged an extra day.
If the Book is rented on a Sunday, the student will get 50% off for as long as they keep the book.
If the Book is rented on a Saturday, the student will get 30% off as long as they keep the book.

We need to write the code to meet this requiement?

Version-1 (We are not considering System Time)

```
1)  #! /bin/bash
2)  echo "Welcome to DURGASOFT BOOK RENTAL APPLICATION"
3)  echo "################################################"
4)
5)  cost_per_day=30
6)
7)  read -p "Was Book returned before 9 PM [Yes|No]:" ontime
8)  read -p "How many days was Book Rented:" days_rented
9)  read -p "What day the Book rented:" day_rented
10)
11) if [ $ontime = "No" ]; then
12)    days_rented=$[days_rented+1]
13) fi
14)
15) if [ $day_rented = "Sunday" ]; then
16)    cost=$(echo "$days_rented * $cost_per_day * 0.5" | bc )
17) elif [ $day_rented = "Saturday" ]; then
18)    cost=$(echo "$days_rented * $cost_per_day * 0.7" | bc )
19) else
20)    cost=$(echo "$days_rented * $cost_per_day " | bc )
21) fi
22)
23) echo "The  Cost Of Book Rental: Rs $cost"
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
24) original_cost=$[days_rented * cost_per_day]
25) discountF=$(echo "$original_cost - $cost" | bc)
26) discountI=$(echo "$discountF" | cut -d "." -f1)
27) if [ $discountI -gt 0 ]; then
28)    echo "Hello, You got Rs $discountF Discount, Enjoy.."
29) fi
30) echo "Visit Again..."
```

Version -2 (Ontime will be generated automatically based on System Time)

```
1)  #! /bin/bash
2)  echo "Welcome to DURGASOFT BOOK RENTAL APPLICATION"
3)  echo "################################################"
4)
5)  cost_per_day=30
6)
7)  read -p "How many days was Book Rented:" days_rented
8)  read -p "What day the Book rented:" day_rented
9)
10) h=$(date +%H)
11) if [ $h -ge 21 ]; then
12)    days_rented=$[days_rented+1]
13) fi
14)
15) if [ $day_rented = "Sunday" ]; then
16)    cost=$(echo "$days_rented * $cost_per_day * 0.5" | bc )
17) elif [ $day_rented = "Saturday" ]; then
18)    cost=$(echo "$days_rented * $cost_per_day * 0.7" | bc )
19) else
20)    cost=$(echo "$days_rented * $cost_per_day " | bc )
21) fi
22)
23) echo "The  Cost Of Book Rental: Rs $cost"
24) original_cost=$[days_rented * cost_per_day]
25) discountF=$(echo "$original_cost - $cost" | bc)
26) discountI=$(echo "$discountF" | cut -d "." -f1)
27)
28) if [ $discountI -gt 0 ]; then
29)    echo "Hello, You got Rs $discountF Discount, Enjoy.."
30) fi
31) echo "Visit Again..."
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Output:

```
durga@durga-VirtualBox:~/scripts$ test.sh
Welcome to DURGASOFT BOOK RENTAL APPLICATION
##############################################
Was Book returned before 9 PM [Yes|No]:Yes
How many days was Book Rented:4
What day the Book rented:Sunday
The  Cost Of Book Rental: Rs 60.0
Hello, You got Rs 60.0 Discount, Enjoy..
Visit Again...

durga@durga-VirtualBox:~/scripts$ test.sh
Welcome to DURGASOFT BOOK RENTAL APPLICATION
##############################################
Was Book returned before 9 PM [Yes|No]:Yes
How many days was Book Rented:6
What day the Book rented:Saturday
The  Cost Of Book Rental: Rs 126.0
Hello, You got Rs 54.0 Discount, Enjoy..
Visit Again...

durga@durga-VirtualBox:~/scripts$ test.sh
Welcome to DURGASOFT BOOK RENTAL APPLICATION
##############################################
Was Book returned before 9 PM [Yes|No]:No
How many days was Book Rented:6
What day the Book rented:Monday
The  Cost Of Book Rental: Rs 210
Visit Again...
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Project-3: Book Management Application

```
1)  #! /bin/bash
2)  add_book()
3)  {
4)    read -p "Enter Name of the Book to Add:" newBook
5)    books[$i]=$newBook
6)    let i++
7)    echo "$newBook Book Added Successfully"
8)    echo
9)  }
10) delete_book()
11) {
12)   read -p "Enter Name of the Book to delete:" book_to_delete
13)   available="no"
14)   for index in ${!books[@]}
15)   do
16)     if [ ${books[$index]} = $book_to_delete ]; then
17)       available="yes"
18)     unset books[$index]
19)     echo "$book_to_delete Book Deleted Successfully"
20)       echo
21)     break
22)     fi
23)   done
24)   if [ $available = "no" ]; then
25)     echo "$book_to_delete Book is not available, cannot be deleted"
26)     echo
27)   fi
28)
29) }
30) list_books()
31) {
32)  if [ ${#books[@]} -eq 0 ]; then
33)    echo "No Books are Available"
34)    echo
35)    return 1
36)  fi
37)  echo "List Of All Available Books:"
38)  echo "----------------------------"
39)  for book in ${books[@]}
40)  do
41)    echo $book
42)  done
```

```
43)   echo
44) }
45) echo "Welcome to DURGASOFT Book Management Application"
46) echo "################################################"
47) declare -a books
48) i=0
49) while [ true ]
50) do
51)   read -p "Which Operation you want to perform [add|delete|list|exit]:" option
52)   case $option in
53)     add)
54)       add_book
55)     ;;
56)     delete)
57)       delete_book
58)     ;;
59)     list)
60)       list_books
61)     ;;
62)     exit)
63)       echo "Thanks for using Our Application"
64)     exit 0
65)     ;;
66)     *)
67)       echo "Wrong Option, Try Again"
68)   esac
69) done
```

Project-4: User Management Application

```
1)  #! /bin/bash
2)  add_user()
3)  {
4)    if [ ! -e users.dat ]; then
5)      touch users.dat
6)    fi
7)
8)    echo "Please Provide details to add user:"
9)    read -p "First Name:" fname
10)   read -p "Last Name:" lname
11)   read -p "User Id:" uid
12)   count=$(cat users.dat | cut -d ":"  -f 1 | grep -w $uid | wc -l)
13)   if [ $count -ne 0 ]; then
14)     echo "User Id:$uid already exists, We cannot add user"
15)     echo
16)     return 1
17)   fi
18)   read -s -p "Password:" pwd
19)   echo
20)   read -p "Retype Password:" cpwd
21)   if [ $pwd != $cpwd ]; then
22)     echo "Passwords are not matching, We cannot add user"
23)     echo
24)     return 2
25)   fi
26)   read -p "ZipCode:" zipcode
27)   echo "$uid:$pwd:$fname:$lname:$zipcode" >> users.dat
28)   echo "User Added Successfully"
29)   echo
30)   echo
31) }
32) search_user()
33) {
34)  read -p "Enter User Id: " uid
35)  count=$(cat users.dat | cut -d ":"  -f1  | grep -w $uid | wc -l)
36)  if [ $count -eq 0 ]; then
37)    echo "User Id: $uid does not exist, Cannot Search for this user"
38)    echo
39)    return 3
40)  fi
41)  read -p "Enter Password: " pwd
42)  count=$(cat users.dat | grep -w $uid | cut -d ":" -f 2 | grep -w $pwd |wc -l)
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
43)  if [ $count -eq 0 ]; then
44)    echo "Invalid Password,, Cannot Search for this user"
45)    echo
46)    return 4
47)  fi
48)
49)  while read line
50)  do
51)    fuid=$(echo $line | cut -d ":" -f1)
52)    fpwd=$(echo $line | cut -d ":" -f2)
53)    if [ $uid = $fuid -a $pwd = $fpwd ]; then
54)      echo "The Complete information of the user is:"
55)      echo "User Id:$(echo $line | cut -d ":" -f 1)"
56)      echo "Password:$(echo $line | cut -d ":" -f 2)"
57)      echo "First Name:$(echo $line | cut -d ":" -f 3)"
58)      echo "Last Name:$(echo $line | cut -d ":" -f 4)"
59)      echo "Zip Code:$(echo $line | cut -d ":" -f 5)"
60)      echo
61)      echo
62)      break
63)    fi
64)    done < users.dat
65) }
66) change_password()
67) {
68)  read -p "Enter User Id: " uid
69)  count=$(cat users.dat | cut -d ":"  -f1  | grep -w $uid | wc -l)
70)  if [ $count -eq 0 ]; then
71)    echo "User Id: $uid does not exist, Cannot Change Password"
72)    echo
73)    return 3
74)  fi
75)  read -p "Enter Password: " pwd
76)  count=$(cat users.dat | grep -w $uid | cut -d ":" -f 2 | grep -w $pwd |wc -l)
77)  if [ $count -eq 0 ]; then
78)    echo "Invalid Password, Cannot Change Password"
79)    echo
80)    return 4
81)  fi
82)  while read line
83)  do
84)    fuid=$(echo $line | cut -d ":" -f1)
85)    fpwd=$(echo $line | cut -d ":" -f2)
86)    if [ $uid = $fuid -a $pwd = $fpwd ]; then
87)      grep -v $line users.dat  > temp.dat
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
88)    record=$line
89)    break
90)    fi
91)  done < users.dat
92)  mv temp.dat users.dat
93)  read -p "Enter New Password:" npwd
94)  uid=$(echo $record | cut -d ":" -f 1)
95)  fname=$(echo $record | cut -d ":" -f 3)
96)  lname=$(echo $record | cut -d ":" -f 4)
97)  zipcode=$(echo $record | cut -d ":" -f 5)
98)  echo "$uid:$npwd:$fname:$lname:$zipcode" >> users.dat
99)  echo "Password updated successfully"
100)          echo
101)          echo
102)        }
103)        delete_user()
104)        {
105)          read -p "Enter User Id: " uid
106)          count=$(cat users.dat | cut -d ":"  -f1  | grep -w $uid | wc -l)
107)          if [ $count -eq 0 ]; then
108)            echo "User Id: $uid does not exist, Cannot Delete User"
109)            echo
110)            return 3
111)          fi
112)          read -p "Enter Password: " pwd
113)        count=$(cat users.dat | grep -w $uid | cut -d ":" -f 2 | grep -w $pwd |wc -l)
114)          if [ $count -eq 0 ]; then
115)            echo "Invalid Password, Cannot Delete User"
116)            echo
117)            return 4
118)          fi
119)          while read line
120)          do
121)            fuid=$(echo $line | cut -d ":" -f1)
122)            fpwd=$(echo $line | cut -d ":" -f2)
123)            if [ $uid = $fuid -a $pwd = $fpwd ]; then
124)               grep -v $line users.dat  > temp.dat
125)            break
126)            fi
127)          done < users.dat
128)          mv temp.dat users.dat
129)          echo "User Deleted Successfully"
130)          echo
131)          echo
132)        }
```

```
133)    show_all_users()
134)    {
135)      echo "All Users Information:"
136)      echo "----------------------"
137)      cat users.dat
138)      echo
139)      echo
140)    }
141)    users_count()
142)    {
143)      count=$(cat users.dat | wc -l)
144)      echo "The Total Number of users:$count"
145)      echo
146)      echo
147)    }
148)    echo "Welcome to User Management Application"
149)    echo "########################################"
150)    while [ true ]
151)    do
152)      echo "1. Add User"
153)      echo "2. Search For User"
154)      echo "3. Change Password"
155)      echo "4. Delete User"
156)      echo "5. Show All Users"
157)      echo "6. Users Count"
158)      echo "7. Exit"
159)      read -p "Enter Your Choice [1|2|3|4|5|6|7]:" choice
160)      case $choice in
161)        1)
162)          add_user
163)          ;;
164)        2)
165)          search_user
166)          ;;
167)        3)
168)          change_password
169)          ;;
170)        4)
171)          delete_user
172)          ;;
173)        5)
174)          show_all_users
175)          ;;
176)        6)
177)          users_count
```

```
178)             ;;
179)          7)
180)             echo "Thanks for using application"
181)             exit 0
182)             ;;
183)          *)
184)             echo "Wrong choice...Try again"
185)       esac
186)    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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

4A) Display only specific Independent Lines:

```
sed -n '2p
> 4p' emp.dat
```

After 2p we have to enter new line.
It will delete only 2nd and 4th lines.
Alternatively we can use the following commands also.

```
sed -n -e '2p' -e '4p' emp.dat
     or
sed -n -e '2p;4p' emp.dat
```

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:

A. To Delete a Particular Line:

```
$ 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.

B) To Delete a Permanently 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.

C) To Delete Range of Lines:

$ 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)

D) To Delete specified Lines:

$ sed -i '1d
> $d' emp.dat

After 1d we are using enter key. It will delete first and last records.

9) 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 convinient 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;
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
    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: Instead of c=c+1, we can use the following lines also.

```
c++;
c+=1
```

Note:

1. Normal Bash commands like echo won't work in AWK.

2. $0 represents total line/record.

Processing multiple files data at a time:

We can use awk to process multiple files also

```
awk 'BEGIN{print "Beginning of the file";print "..............................";c=0;}{print
$0;c++;}END{print "..............................";print "The Number of Lines:" c}' file1.txt
file2.txt
```

Beginning of the file

```
File1 content
File1 content
File1 content
File2 content
File2 content
File2 content
File2 content
..............................
The Number of Lines:7
```

Q) Can we restrict on no of times action block should be executed?

Yes. We can define some constraints. will be discussed in next level

Processing tabular data by using awk:

While procesing 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.

Instead of tab, if we use any other separator then we can handle by using FS variable or -F option. Will be discussed in next level

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 $3}' emp.txt
enoesal
1001000
2002000
3003000
4004000
5005000
durgasoft@durgasoft:~/scripts$ 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

Bydefault 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

durgasoft@durgasoft:~/scripts$ awk -F "," '{print $2" "$4}' emp.csv
ename eaddr
Sunny Mumbai
Bunny Hyderabad
Chinny Hyderabad
Vinny Chennai
Pinny Mumbai
```

Note:

$ awk -F "," '{print $2" "$4}' emp.csv → Valid
$ awk '{print $2" "$4}' FS="," emp.csv → Valid
$ awk FS="," '{print $2" "$4}' emp.csv → Invalid

```
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 prededined 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

awk 'NR>2{print $0}' FS="," emp.csv
awk 'NR>2{print $1}' FS="," emp.csv
awk 'NR==2 || NR==4 {print $0}' FS="," emp.csv
It will display only 2nd and 4th records
awk 'NR==2 || NR==4 {print $1}' FS="," emp.csv
It will display only 2nd and 4th records first field value.
awk 'NR>2 && NR<5 {print $0}' FS="," emp.csv
It will display only 3rd and 4th records
awk 'NR>1 && NR<5 {print $0}' FS="," emp.csv
It will display from 2nd record to 4th record.
awk 'NR==2,NR==4{print $0}' FS=',' emp.csv
It will display from 2nd record to 4th record.

Q) Consider the File

xy.txt
1.abc
2.def
3.ghi
4.jkl5.mno

What is the output of the following commands:
1. awk '{print $0;}' xy.txt
2. awk '{print $1;}' xy.txt
3. awk '{print $2;}' xy.txt

1. awk '{print $0;}' xy.txt
It will print total file
2. awk '{print $1;}' xy.txt
It will print total file, because file contains only one column.
3. awk '{print $2;}' xy.txt
Nothing will be printed, because 2nd column is not available.

Note: awk will always treat every file as tabular file only.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

AWK Code into a separate File:

We can separate awk code into a file and we can provide that file data by using -f option with awk command.

$ awk -f <programfile> <datafile>

x.txt

```
BEGIN {print "BEGIN"}
{print $0}
END { print "End"}
```

```
$ awk -f x.txt  FS="," emp.csv
durga@durga-VirtualBox:~/Desktop$ awk -f x.txt FS="," emp.csv
BEGIN
eno,ename,esal,eaddr
100,Sunny,1000,Mumbai
200,Bunny,1000,Hyderabad
300,Chinny,3000,Hyderabad
400,Vinny,4000,Chennai
500,Pinny,5000,Mumbai
End
```

Note: **We can configure FS="," in awk code itself.**

x.txt

```
BEGIN {FS=",";print "BEGIN"}
{print $0}
END { print "End"}
```

```
$ awk -f x.txt emp.csv
```

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

durga@durga-VirtualBox:~/Desktop$ awk 'NR!=1 && $3>2500{print $0}' FS="," emp.csv
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
```

2nd Way

x.txt

```
BEGIN{
   FS=",";
   c=0;
}
NR!=1{
 if($3>3000)
 {
   c=c+1;
   print $0;
  }
}
END{
 print "The Total Number of Employees where salary > 3000 :" c
}
```

```
durga@durga-VirtualBox:~/Desktop$ awk -f x.txt  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
```

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
```

2nd Way with File:

```
durga@durga-VirtualBox:~/Desktop$ cat 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
20,Durga,7000,Hyderabad
```

x.txt

```
BEGIN{
      FS=",";
}
NR!=1{
          if($3>3000)
          {
            print $2" is very costly employee";
          }
          else
          {
            print $2" is not costly employee";
          }
}
```

```
awk -f x.txt emp.csv
durga@durga-VirtualBox:~/Desktop$ awk -f x.txt emp.csv
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
Durga is very costly employee
```

for Loop Syntax in AWK:

```
for(initialization section;conditional check;increment/decrement)
{
   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

echo "CAT  RAT  DOG  TIGER" | awk '{for(i=1;i<=NF;i++){print "Field:" i":" $i}}'
cat emp.txt | awk '{for(i=1;i<=NF;i++){print "Field:" i":" $i;}print "";}'

durga@durga-VirtualBox:~/Desktop$ cat emp.txt | awk '{for(i=1;i<=NF;i++){print "Field:"
i":" $i;}print "";}'
Field:1:eno
Field:2:ename
Field:3:esal
Field:4:eaddr

Field:1:100
Field:2:Sunny
Field:3:1000
Field:4:Mumbai

Field:1:200
Field:2:Bunny
Field:3:1000
Field:4:Hyderabad
```

Field:1:300
Field:2:Chinny
Field:3:3000
Field:4:Hyderabad

Field:1:400
Field:2:Vinny
Field:3:4000
Field:4:Chennai

Field:1:500
Field:2:Pinny
Field:3:5000
Field:4:Mumbai

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
```

Note: In BEGIN Block, NR and NF representing zero. Hence we cannot use these inside BEGIN Block.

```
durga@durga-VirtualBox:~/Desktop$ awk 'BEGIN{print "The Number of Records:" NR;print "The Number of Fields:" NF;}' emp.txt
The Number of Records:0
The Number of Fields:0
```

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
```

Assume : is Record Separator and , is Field Separator

```
awk 'BEGIN{FS=",";RS=":"}{print "Record Number:" NR;print $1"...>"$2;print ""} END{print "The Number of Records:" NR;print "The Number of Fields:" NF;}' demo.txt
```

```
durga@durga-VirtualBox:~/Desktop$ awk 'BEGIN{FS=",";RS=":"}{print "Record Number:" NR;print $1"--->"$2;print ""} END{print "The Number of Records:" NR;print "The Number of Fields:" NF;}' demo.txt
Record Number:1
APPLE--->BANANA

Record Number:2
CAT--->DOG

Record Number:3
JAVA--->PYTHON

The Number of Records:3
The Number of Fields:2
```

Note: We can change order of BEGIN,ACTION and END blocks but not recommended.

```
durga@durga-VirtualBox:~/Desktop$ awk '{print "ACTION"}BEGIN{print "BEGIN"}END{print "END"}' demo.txt
o/p:
BEGIN
ACTION
END
```

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

Topic-37: Project on awk Programming

Five Most Beautiful Scripts by using awk:

shift Comamnd:

We can use shift command to move the command line arguments to one position left.
The value of $2 moves to $1, and the value of $3 moves to $2 etc

Q1) Write a Script to iterate Command Line Arguments, where Order is not Important and the Number of Arguments is not Important?

```
$ test.sh
    -l location | --location location
    -e extension | --extension extension
    -s | --stats
    -h | --help

 test.sh  -l  /etc
 test.sh  --location  /etc  -e conf
 test.sh  --location  /etc  -e conf  -s
 test.sh  --location  /etc  -e conf  --stats
 test.sh -e txt
```

test.sh

```
#! /bin/bash

while [ $# != 0 ]
do
  case $1 in
    -l|--location)
        echo "-l | --location option provided"
            LOCATION=$2
            echo "$LOCATION provided as argument"
            shift
            shift
            ;;
    -e|--extension)
        echo "-e | --extension option provided"
            EXTENSION=$2
            echo "$EXTENSION provided as argument"
```

```
            shift
            shift
            ;;
    -s|--stats)
        echo "-s|--stats option provided"
            shift
            ;;
    -h|--help)
        echo "-h|--help option provided"
            shift
            ;;
    *)
        echo "Invalid Option"
            shift
  esac
done
```

Output

```
durga@durga-VirtualBox:~/scripts$ test.sh
durga@durga-VirtualBox:~/scripts$ test.sh -l /etc
-l | --location option provided
/etc provided as argument
durga@durga-VirtualBox:~/scripts$ test.sh -l /etc --extension txt
-l | --location option provided
/etc provided as argument
-e | --extension option provided
txt provided as argument
durga@durga-VirtualBox:~/scripts$ test.sh -e txt -l /etc
-e | --extension option provided
txt provided as argument
-l | --location option provided
/etc provided as argument

durga@durga-VirtualBox:~/scripts$ test.sh -e txt -l /etc --stats
-e | --extension option provided
txt provided as argument
-l | --location option provided
/etc provided as argument
-s|--stats option provided
durga@durga-VirtualBox:~/scripts$ test.sh -h -e txt -l /etc --stats
-h|--help option provided
-e | --extension option provided
txt provided as argument
-l | --location option provided
/etc provided as argument
```

```
-s|--stats option provided
durga@durga-VirtualBox:~/scripts$ test.sh durga
Invalid Option
durga@durga-VirtualBox:~/scripts$ test.sh durga -l /etc
Invalid Option
-l | --location option provided
/etc provided as argument
```

Q2) Write a Script to find the Number of Files Present in the given Location?

```
test.sh -l /etc
test.sh -l /bin
```

test.sh

```
#! /bin/bash

LOCATION=$2

if [ ! -d $LOCATION ]; then
  echo "You should provide valid directory as Location"
  exit 1
fi

ls -l $LOCATION | awk '/^-/' | awk 'END{print "The Number of Files:" NR}'
```

Output

```
durga@durga-VirtualBox:~/scripts$ test.sh -l /etc
The Number of Files:93
durga@durga-VirtualBox:~/scripts$ test.sh -l /bin
The Number of Files:140
durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts
The Number of Files:19
durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts/dir99
The Number of Files:0
durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts/dir9999
You should provide valid directory as Location
```

Q3) Write a Script to find Total Number of Files based on given Location and Extension? Assume the following is Sample Script Invocation

test.sh -l /etc -e conf
We have to find the number of files with .conf extension in /etc directory.

test.sh

```
#! /bin/bash

LOCATION=$2
EXTENSION=$4

if [ ! -d $LOCATION ]; then
  echo "You should provide valid directory as Location"
  exit 1
fi

ls -l $LOCATION | awk '/^-/' | awk "/.$EXTENSION$/" | awk 'END{print "The Number of Files:" NR}'
```

Output

```
durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts -e txt
The Number of Files:10
durga@durga-VirtualBox:~/scripts$ test.sh -l /etc -e conf
The Number of Files:31
durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts/dir99 -e txt
The Number of Files:0
durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts -e conf
The Number of Files:0
durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts -e sh
The Number of Files:7
```

Q4) Write a Script to find Total Size of all Files based on given Location and Extension?

test.sh

```
#! /bin/bash

LOCATION=$2
EXTENSION=$4

if [ ! -d $LOCATION ]; then
  echo "You should provide valid directory as Location"
  exit 1
fi

ls -l $LOCATION | awk '/^-/' | awk "/.$EXTENSION$/" | awk
'BEGIN{sum=0}{sum=sum+$5}END{print "The Number of Files:" NR;if(NR > 0){print "The
Total Size of all files:" sum " Bytes";print "The Total Size of all files:" sum/1024/1024 "
MB";}}'

durga@durga-VirtualBox:~/scripts$ test.sh -l /etc  -e txt
The Number of Files:0
durga@durga-VirtualBox:~/scripts$ test.sh -l /etc  -e conf
The Number of Files:31
The Total Size of all files:98086 Bytes
The Total Size of all files:0.0935421 MB
```

Q5) Write a Script to find Largest File and Smallest File based on given Location and Extension?

test.sh

```
#! /bin/bash

LOCATION=$2
EXTENSION=$4

if [ ! -d $LOCATION ]; then
  echo "You should provide valid directory as Location"
  exit 1
fi

ls -l $LOCATION | awk '/^-/' | grep ".$EXTENSION$" &> /dev/null
if [ $? -ne 0 ]; then
  echo "No Files Found In $LOCATION location with $EXTENSION extension"
```

```
  exit 2
fi

ls -l $LOCATION | awk '/^-/' | awk "/.$EXTENSION$/" | awk
'{if(NR==1){min=$5;min_file=$9;max=$5;max_file=$9;}if($5<min){min=$5;min_file=$9}if($
5>max){max=$5;max_file=$9;}}END{print "The Smallest File:" min_file;print "The Smallest
File Size:" min " Bytes";print "The Largest File:" max_file;print "The Largest File Size:" max
" Bytes";}'

durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts  -e conf
No Files Found In /home/durga/scripts location with conf extension
durga@durga-VirtualBox:~/scripts$ test.sh -l ~/scripts  -e txt
The Smallest File:abc.txt
The Smallest File Size:16 Bytes
The Largest File:output.txt
The Largest File Size:838 Bytes

durga@durga-VirtualBox:~/scripts$ test.sh -l /etc  -e conf
The Smallest File:libao.conf
The Smallest File Size:27 Bytes
The Largest File:brltty.conf
The Largest File Size:25341 Bytes
durga@durga-VirtualBox:~/scripts$ test.sh -l /etc  -e txt
No Files Found In /etc location with txt extension
```

Project on AWK Programming:

```
1)  #! /bin/bash
2)  help_function()
3)  {
4)     echo "$0 can provide information based on location,extension including memory
    info"
5)     echo
6)     echo "$0 [-l location][--location location][-e extension][--extension extension][-
    s|--stats][-h|--help]"
7)     echo
8)     echo "Examples:"
9)     echo "$0  -l /etc"
10)    echo "$0  -l /etc  -e conf"
11)    echo "$0  -l /etc  -e conf --stats"
12)    exit 2
13) }
14)
15) LOCATION=$(pwd)
```

```
16) EXTENSION=""
17) STATS=0
18)
19) while [ $# != 0 ]
20) do
21)   case $1 in
22)     -l|--location)
23)         LOCATION=$2
24)       if [ ! -d $LOCATION ]; then
25)         echo "Your provided location either not a valid directory or does not exit"
26)         exit 1
27)       fi
28)       shift
29)       shift
30)       ;;
31)     -e|--extension)
32)         EXTENSION=$2
33)       shift
34)       shift
35)       ;;
36)      -s|--stats)
37)         STATS=1
38)       shift
39)       ;;
40)      -h|--help)
41)         help_function
42)       ;;
43)      *)
44)         echo "You are providing invalid option, please check help doc below"
45)         help_function
46)
47)   esac
48) done
49)
50) echo -n "This script finding all files with "
51) if [ "$EXTENSION" = "" ]; then
52)    echo -n "all extensions in "
53) else
54)    echo -n ".$EXTENSION extension in "
55) fi
56) echo "$LOCATION location"
57) echo
58)
59) ls -l $LOCATION | awk '/^-/' | grep ".$EXTENSION$" &> /dev/null
60) if [ $? -ne 0 ]; then
```

```
61)   echo "But, No files Found in $LOCATION location with .$EXTENSION extension"
62)   exit 3
63) fi
64)
65) ls -l $LOCATION | awk '/^-/' | grep ".$EXTENSION$" |
66) awk 'BEGIN{
67)  sum=0;
68) }
69) {
70)  sum=sum+$5;
71) }
72) END{
73)  print "The Number of Files:" NR;
74)  print "The Total Size of all files: " sum " Bytes";
75)  print "The Total Size of all files: " sum/1024 " KB";
76)  print "The Total Size of all files: " sum/1024/1024 " MB";
77)  print ""
78)  }'
79)
80) if [ $STATS -eq 1 ]; then
81)   echo "The Statistics Are:"
82)   ls -l $LOCATION | awk '/^-/' | grep ".$EXTENSION$" |
83)   awk '{
84)   if(NR==1){
85)     min_file_size=$5;
86)     min_file_name=$9;
87)     max_file_size=$5;
88)     max_file_name=$9;
89)   }
90)   else{
91)     if($5<min_file_size){
92)       min_file_size=$5;
93)    min_file_name=$9;
94)     }
95)     if($5>max_file_size){
96)       max_file_size=$5;
97)    max_file_name=$9;
98)     }
99)   }}
100)          END{
101)            print "The Smallest File Name: " min_file_name;
102)            print "The Smallest File Size: " min_file_size;
103)            print "The Largest File Name: " max_file_name;
104)            print "The Largest File Size: " max_file_size;
105)          }'
```

```
106)        fi
```

Separating AWK Code into other Files:

test.sh

```
1)  #! /bin/bash
2)  help_function()
3)  {
4)     echo "$0 can provide information based on location,extension including memory
    info"
5)     echo
6)     echo "$0 [-l location][--location location][-e extension][--extension extension][-
    s|--stats][-h|--help]"
7)     echo
8)     echo "Examples:"
9)     echo "$0  -l /etc"
10)    echo "$0  -l /etc  -e conf"
11)    echo "$0  -l /etc  -e conf --stats"
12)    exit 2
13) }
14)
15) LOCATION=$(pwd)
16) EXTENSION=""
17) STATS=0
18)
19) while [ $# != 0 ]
20) do
21)   case $1 in
22)     -l|--location)
23)          LOCATION=$2
24)       if [ ! -d $LOCATION ]; then
25)          echo "Your provided location either not a valid directory or does not exit"
26)          exit 1
27)       fi
28)       shift
29)       shift
30)       ;;
31)     -e|--extension)
32)          EXTENSION=$2
33)       shift
34)       shift
35)       ;;
36)      -s|--stats)
```

```
37)         STATS=1
38)       shift
39)       ;;
40)      -h|--help)
41)          help_function
42)       ;;
43)      *)
44)         echo "You are providing invalid option, please check help doc below"
45)         help_function
46)
47)   esac
48) done
49)
50) echo -n "This script finding all files with "
51) if [ "$EXTENSION" = "" ]; then
52)    echo -n "all extensions in "
53) else
54)    echo -n ".$EXTENSION extension in "
55) fi
56) echo "$LOCATION location"
57) echo
58)
59) ls -l $LOCATION | awk '/^-/' | grep ".$EXTENSION$" &> /dev/null
60) if [ $? -ne 0 ]; then
61)    echo "But, No files Found in $LOCATION location with .$EXTENSION extension"
62)    exit 3
63) fi
64)
65) ls -l $LOCATION | awk '/^-/' | grep ".$EXTENSION$" |
66) awk -f awk_code_1
67)
68) if [ $STATS -eq 1 ]; then
69)    echo "The Statistics Are:"
70)    ls -l $LOCATION | awk '/^-/' | grep ".$EXTENSION$" |
71)    awk -f awk_code_2
72) fi
73)
74) awk_code_1:
75) -----------------
76) BEGIN{
77)   sum=0;
78) }
79) {
80)  sum=sum+$5;
81) }
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
82) END{
83)  print "The Number of Files:" NR;
84)  print "The Total Size of all files: " sum " Bytes";
85)  print "The Total Size of all files: " sum/1024 " KB";
86)  print "The Total Size of all files: " sum/1024/1024 " MB";
87)  print ""
88)  }
89)
90) awk_code_2:
91) -----------
92) {
93)   if(NR==1){
94)     min_file_size=$5;
95)     min_file_name=$9;
96)     max_file_size=$5;
97)     max_file_name=$9;
98)   }
99)   else{
100)            if($5<min_file_size){
101)              min_file_size=$5;
102)           min_file_name=$9;
103)            }
104)            if($5>max_file_size){
105)              max_file_size=$5;
106)           max_file_name=$9;
107)            }
108)          }
109)        }
110)        END{
111)            print "The Smallest File Name: " min_file_name;
112)            print "The Smallest File Size: " min_file_size;
113)            print "The Largest File Name: " max_file_name;
114)            print "The Largest File Size: " max_file_size;
115)        }
```

Output

```
durga@durga-VirtualBox:~/scripts$ test.sh -l /etc -e conf -s
This script finding all files with .conf extension in /etc location

The Number of Files:31
The Total Size of all files: 98086 Bytes
The Total Size of all files: 95.7871 KB
The Total Size of all files: 0.0935421 MB
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

```
The Statistics Are:
The Smallest File Name: libao.conf
The Smallest File Size: 27
The Largest File Name: brltty.conf
The Largest File Size: 25341

durga@durga-VirtualBox:~/scripts$ ls -l *.txt
-rw-r--r-- 1 durga newgrp 217 Apr  2 19:17 abcd.txt
-rw-r--r-- 1 durga newgrp  16 Apr  9 13:10 abc.txt
-rw-r--r-- 1 durga newgrp 243 Apr  2 19:17 a.txt
-rw-r--r-- 1 durga newgrp 243 Apr  2 19:17 b.txt
-rw-r--r-- 1 durga newgrp 254 Apr  2 19:17 c.txt
-rw-r--r-- 1 durga newgrp 216 Apr  2 20:04 emp.txt
-rw-r--r-- 1 durga newgrp  51 Apr  2 19:41 hyd.txt
-rw-r--r-- 1 durga newgrp 838 Apr  5 20:16 output.txt
-rw-r--r-- 1 durga newgrp 696 Apr  2 20:31 result.txt
----r--r-- 1 durga newgrp  26 Mar 30 11:53 z.txt

durga@durga-VirtualBox:~/scripts$ test.sh -e txt -s
This script finding all files with .txt extension in /home/durga/scripts location

The Number of Files:10
The Total Size of all files: 2800 Bytes
The Total Size of all files: 2.73438 KB
The Total Size of all files: 0.00267029 MB

The Statistics Are:
The Smallest File Name: abc.txt
The Smallest File Size: 16
The Largest File Name: output.txt
The Largest File Size: 838
durga@durga-VirtualBox:~/scripts$ test.sh -s -e conf -l /etc
This script finding all files with .conf extension in /etc location

The Number of Files:31
The Total Size of all files: 98086 Bytes
The Total Size of all files: 95.7871 KB
The Total Size of all files: 0.0935421 MB

The Statistics Are:
The Smallest File Name: libao.conf
The Smallest File Size: 27
The Largest File Name: brltty.conf
The Largest File Size: 25341
```

Part-3

Linux Administration Basics

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 reange : 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 * * *

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 1 */2 *
```

23) Every 6 months once

```
0 0 1 */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 *
```

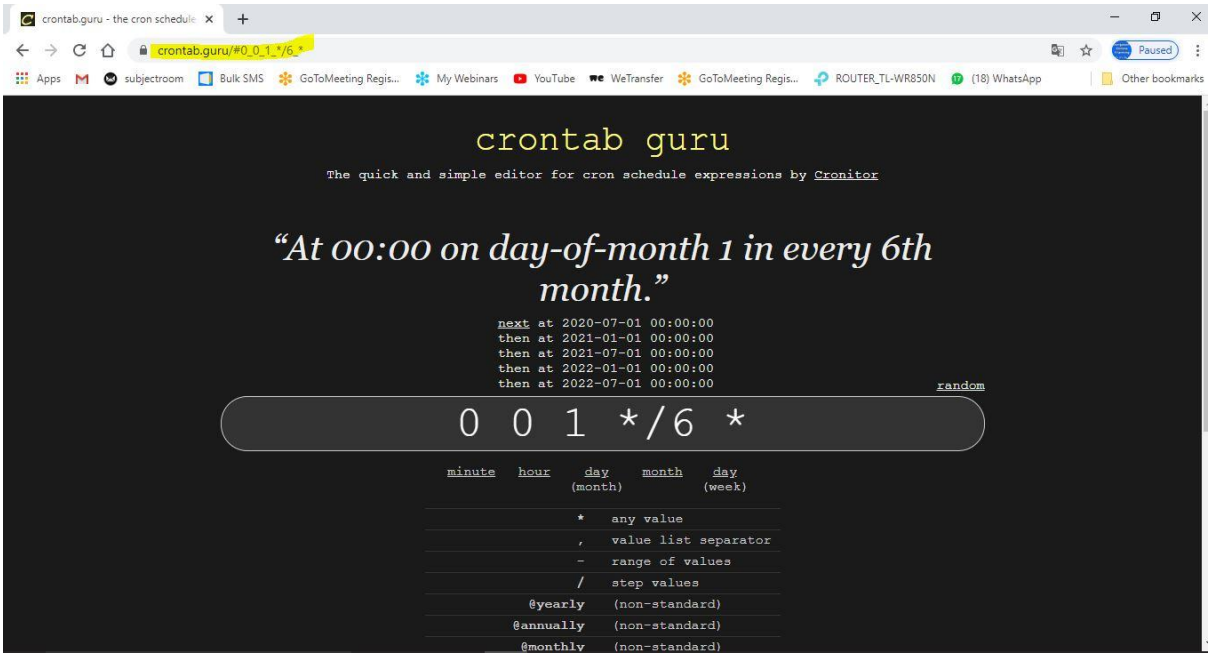
Crontab Shortcuts:

```
@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.


crontab.guru - the cron schedule
crontab.guru/#0_0_1_*/6_*
Paused
Apps
subjectroom
Bulk SMS
GoToMeeting Regis...
My Webinars
YouTube
WeTransfer
GoToMeeting Regis...
ROUTER_TL-WR850N
(18) WhatsApp
Other bookmarks
crontab guru
The quick and simple editor for cron schedule expressions by Cronitor
"At 00:00 on day-of-month 1 in every 6th month."
next at 2020-07-01 00:00:00
then at 2021-01-01 00:00:00
then at 2021-07-01 00:00:00
then at 2022-01-01 00:00:00
then at 2022-07-01 00:00:00
random
0 0 1 */6 *
minute
hour
day (month)
month
day (week)
* any value
, value list separator
- range of values
/ step values
@yearly (non-standard)
@annually (non-standard)
@monthly (non-standard)

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
11) How to check available groups?
12) How to change password of a user?
13) What is the difference between adduser and useradd in Linux?
14) sudo command
15) sudoers file
16) sudo command
17) sudoers file
18) How to check the allowed commands by sudo for a particular user?

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 user finger command. But bydefault 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
  libmspub-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]
Fetched 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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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
```

11) 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]

12) How to Change Password of User:

To change a password for user named durga:
$ sudo passwd durga

To change a password for root user:
$ sudo passwd root

To change your own password:
$ passwd

13) 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 creater 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

14) sudo Command:

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

15) 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/bin"

# 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
```

16) 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
```

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

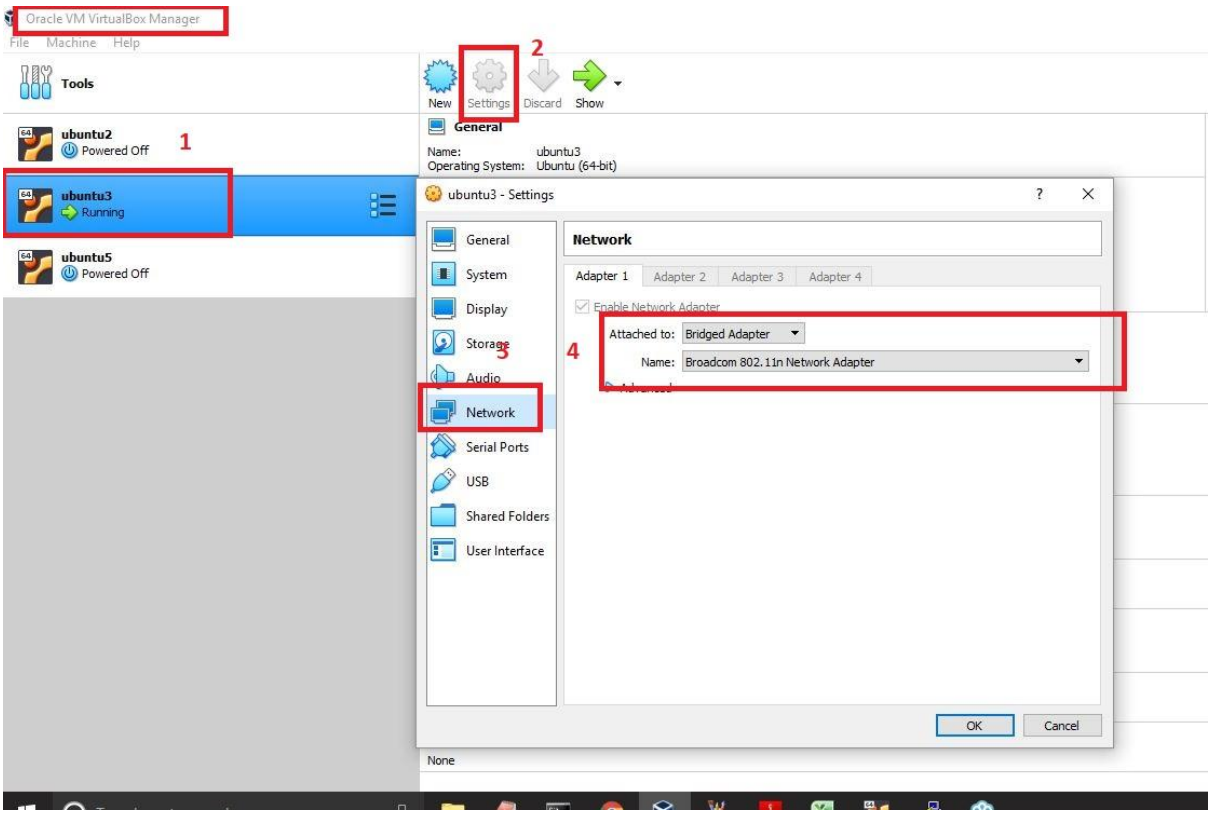
Requirements for connecting with Putty:

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:

Bydefault 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)
RealTekPCIe... (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:

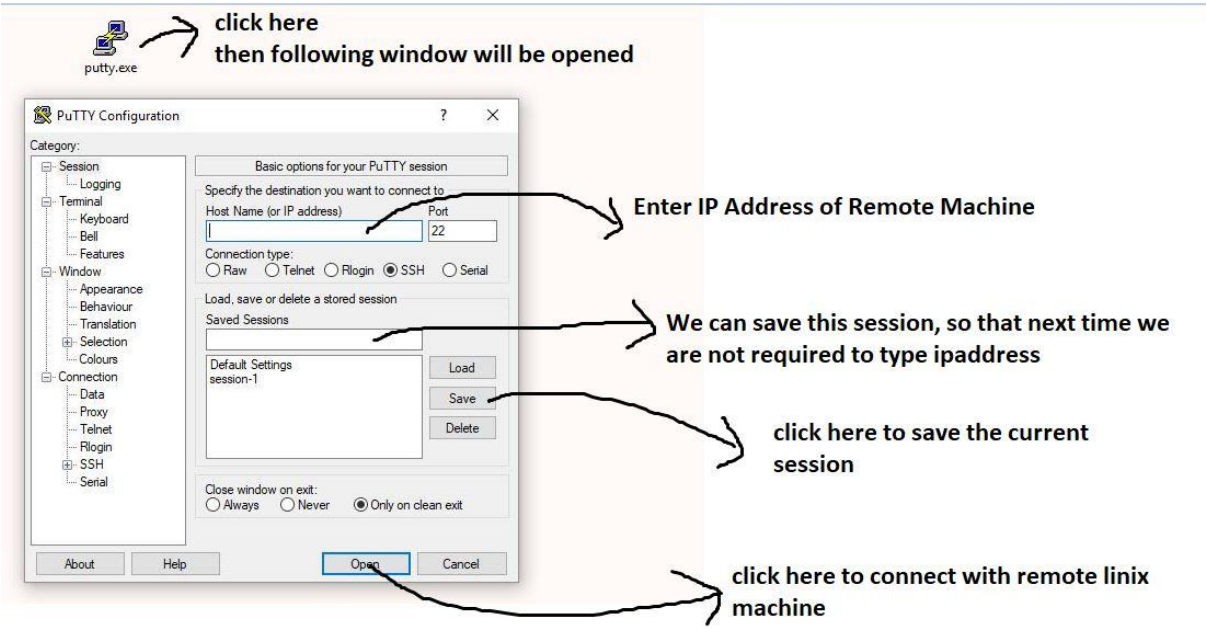
Bydefault in ubuntu ssh server is not available and we have to install separately.

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

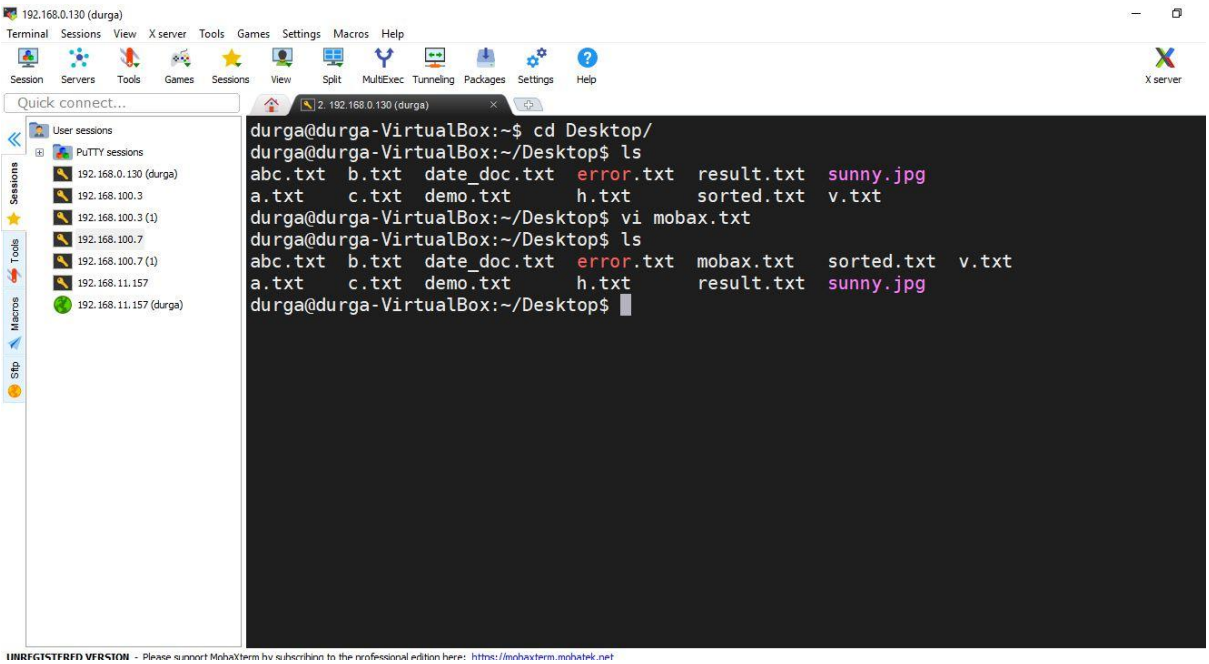
Open putty,provide ipaddress, click Open



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

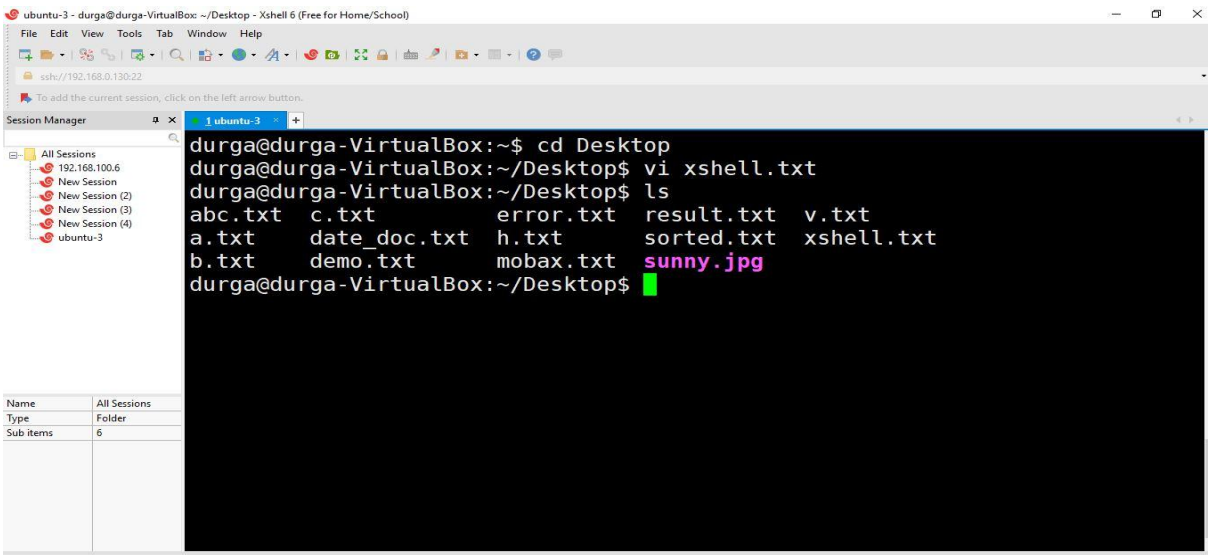
Open Mobaxterm → Right click on User sessions → New Session → SSH → provide IP Address → OK



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.



DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 responible 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, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 bydefault 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 coretutils package from gnu.org
2) uncompress and extract tar file
3) Perform required modifictions 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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-udevd
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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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 snapd
```

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, mesg

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 Communcation):

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
```


DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

wall Command (For Online Communcation):

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 ~/msmtp.log
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 is 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 installs, 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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

```
__        __  _
\ \      / /__| | ___ ___  _ __ ___   ___
 \ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \
  \ V  V /  __/ | (_| (_) | | | | | |  __/
   \_/\_/ \___|_|\___\___/|_| |_| |_|\___|
```

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    mmmmm    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 bydefault 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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. Donwload 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

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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.

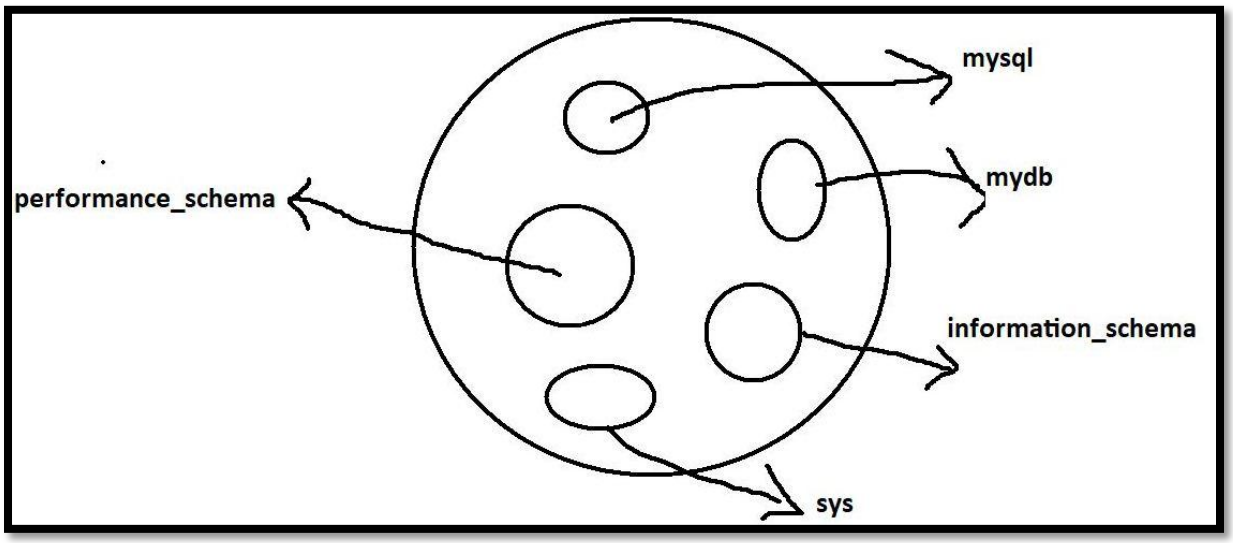
```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydb               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.05 sec)

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.

DURGASOFT, # 202, 2nd Floor, HUDA Maitrivanam, Ameerpet, Hyderabad - 500038,
☎ 88 85 25 26 27, 72 07 21 24 27/28 | www.durgasoftonline.com
Maii: durgasoftonline@gmail.com

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)
```