

Practical-2

Aim: Study of Unix Architecture and the following Unix commands with option:

User Access:

Login: The login program is used to establish a new session with the system. It is normally invoked automatically by responding to the "login:" prompt on the user's terminal.

Syntax: login [-p] [-h *host*] [*username*] [*ENV=VAR...*]

Output:

```
[root@localhost ~]# adduser Yatharth
[root@localhost ~]# su Yatharth
[Yatharth@localhost root]$ exit
exit
[root@localhost ~]# login Yatharth
/sbin/init: line 53: 47 Hangup                setsid -c sh -c "exec sh -l <
/dev/$ttyname > /dev/$ttyname 2>&1"
[root@localhost ~]# █
```

Exit: When you run exit, if you have jobs running in the background, the shell will remind you that they are running and return you to the command prompt. In this case, issuing exit again will terminate those jobs and exit the shell.

Common aliases for **exit** include "**bye**", "**logout**", and "**lo**".

Syntax: exit

Output:

```
[root@localhost ~]# su Yatharth
[Yatharth@localhost root]$ exit
exit
```

logout: Exits a login shell with exit status N. Returns an error if not executed in a login shell.

Syntax: logout [n] Exit a login shell.

Output:

```
[Yatharth@localhost root]$ logout
bash: logout: not login shell: use `exit'
[Yatharth@localhost root]$ exit
exit
[root@localhost ~]#
```

passwd: The passwd command changes passwords for user accounts. A normal user may only change the password for their own account, while the super user may change the password for any account. passwd also changes the account or associated password validity period.

Syntax: passwd

Output:

```
[root@localhost ~]# passwd Yatharth
Changing password for user Yatharth.
New password:
BAD PASSWORD: The password fails the dictionary check - it is too simplistic/systematic
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost ~]#
```

Help:

Man: man, command in Linux is used to display the user manual of any command that we can run on the terminal.

Syntax: man is

Output:

```
[root@localhost ~]# man is
No manual entry for is
[root@localhost ~]#
```

Help: The help command is a shell built-in internal command. It accepts a text string as the command line argument and searches the supplied string in the shell's documents.

Syntax: help [-dms] [pattern ...]

Output:

```
[root@localhost ~]# help
GNU bash, version 5.0.17(1)-release (riscv64-redhat-linux-gnu)
These shell commands are defined internally. Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.
Use 'man -k' or 'info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

job_spec [&]
(( expression ))
. filename [arguments]
:
[ arg... ]
[[ expression ]]
alias [-p] [name[=value] ... ]
bg [job_spec ...]
bind [-lpsvPSVX] [-m keymap] [-f filename] [-q name] [-u name] [-r keys>
break [n]
builtin [shell-builtin [arg ...]]
caller [expr]
case WORD in [PATTERN [| PATTERN]...] COMMANDS ;;)... esac
cd [-L|-P [-e]] [-@]] [dir]
command [-pVv] command [arg ...]
compgen [-abdefgjklsuv] [-o option] [-A action] [-G globpat] [-W wordli>
complete [-abdefgjklsuv] [-pr] [-DEI] [-o option] [-A action] [-G globp>
comptop [-o|+o option] [-DEI] [name ...]
continue [n]
coproc [NAME] command [redirections]
declare [-aAffglnrtux] [-p] [name[=value] ...]
dirs [-clpv] [+N] [-N]
disown [-h] [-ar] [jobspec ... | pid ...]
echo [-neE] [arg ...]

history [-c] [-d offset] [n] or history -anrw [filename] or history -p>
if COMMANDS; then COMMANDS; [ elif COMMANDS; then COMMANDS; ]... [ els>
jobs [-lnprs] [jobspec ...] or jobs -x command [args]
kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l >
let arg [arg ...]
local [option] name[=value] ...
logout [n]
mapfile [-d delim] [-n count] [-O origin] [-s count] [-t] [-u fd] [-C >
popd [-n] [+N | -N]
printf [-v var] format [arguments]
pushd [-n] [+N | -N | dir]
pwd [-LP]
read [-ers] [-a array] [-d delim] [-i text] [-n nchars] [-N nchars] [->
readarray [-d delim] [-n count] [-O origin] [-s count] [-t] [-u fd] [->
readonly [-aAf] [name[=value] ...] or readonly -p
return [n]
select NAME [in WORDS ... ;] do COMMANDS; done
set [-abefhkmnptuvxBCHP] [-o option-name] [--] [arg ...]
shift [n]
shopt [-pqsu] [-o] [optname ...]
source filename [arguments]
suspend [-f]
test [expr]
time [-p] pipeline
```

Directory:

mkdir: make directories

Syntax: mkdir <name>

Output:

```
[root@localhost ~]# ls
bench.py  hello.c
[root@localhost ~]# mkdir Yatharth
[root@localhost ~]# ls
bench.py  hello.c  Yatharth
[root@localhost ~]#
```

rmdir: rmdir command is used remove empty directories from the filesystem in Linux

Syntax: rmdir <name>

Output:

```
[root@localhost ~]# rmdir Yatharth
[root@localhost ~]# ls
bench.py  hello.c
[root@localhost ~]#
```

cd : To change directory - change the current working directory to a specific Folder.

Syntax: cd <name>

Output:

```
[root@localhost ~]# mkdir 20DCE019
[root@localhost ~]# cd 20DCE019
[root@localhost 20DCE019]#
```

pwd: pwd stands for Print Working Directory. It prints the path of the working directory, starting from the root.

Syntax: `pwd -L`

Output:

```
[root@localhost 20DCE019]# pwd
/root/20DCE019
[root@localhost 20DCE019]#
```

ls: ls List information about the FILES (the current directory by default). Sort entries alphabetically if none of -cftuvSUX nor --sort is specified. Mandatory arguments to long options are mandatory for short options too.

Syntax: `ls [OPTION]... [FILE]...`

Output:

```
[root@localhost 20DCE019]# ls
1.txt 2.txt
[root@localhost 20DCE019]#
```

mv : mv stands for move. mv is used to move one or more files or directories from one place to another in a file system like UNIX. It has two distinct functions:

- (i) It renames a file or folder.
- (ii) It moves a group of files to a different directory.

No additional space is consumed on a disk during renaming. This command normally works silently means no prompt for confirmation.

Syntax: mv <name><name>

Output:

```
[root@localhost 20DCE019]# ls
1.txt 2.txt
[root@localhost 20DCE019]# mv 1.txt 2.txt
[root@localhost 20DCE019]# ls
2.txt
[root@localhost 20DCE019]#
```

Editor:

vi: Vim is a text editor that is upwards compatible to Vi. It can be used to edit all kinds of plain text. It is especially useful for editing programs. There are a lot of enhancements above Vi: multi level undo, multi windows and buffers, syntax highlighting, command line editing, filename completion, on-line help, visual selection, etc..

Syntax: vim [options] [file ..]

Output:

```
[root@localhost 20DCE019]# vim 2.txt
[root@localhost 20DCE019]# cat 2.txt
Hi
Hello
everyone
[root@localhost 20DCE019]#
```

gedit: gedit is the default text editor of the GNOME desktop environment and part of the GNOME Core Applications. Designed as a general-purpose text editor, gedit emphasizes simplicity and ease of use, with a clean and simple GUI, according to the philosophy of the GNOME project.

Syntax: gedit filename

ed: ed is a line-oriented text editor. It is used to create, display, modify and otherwise manipulate text files. red is a restricted ed: it can only edit files in the current directory and cannot execute shell commands.

Syntax: ed [-] [-Gs] [-p *string*] [*file*]

Output:



Charotar University of Science and Technology
Devang Patel Institute of Advance Technology and Research
Department of Computer Engineering



```
[root@localhost 20DCE019]# ed  
help  
?
```


File Handling / Text Processing:

cp: To copy one or more files to another location. cp stands for copy. This command is used to copy files or group of files or directory. It creates an exact image of a file on a disk with different file name. cp command require at least two filenames in its arguments.

Syntax: cp -r

Output:

```
[root@localhost 20DCE019]# cp 2.txt 3.txt
[root@localhost 20DCE019]# cat 3.txt
Hi
Hello
everyone
[root@localhost 20DCE019]#
```

mv: Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY. Mandatory arguments to long options are mandatory for short options too.

Syntax: mv [options] [-T] source destination

Output:

```
[root@localhost 20DCE019]# ls
2.txt 3.txt
[root@localhost 20DCE019]# mv 2.txt 3.txt
[root@localhost 20DCE019]# l
sh: l: command not found
[root@localhost 20DCE019]# ls
3.txt
[root@localhost 20DCE019]#
```

rm:rm removes each specified file. By default, it does not remove directories. If the *-I* or *--interactive=once* option is given, and there are more than three files or the *-r*, *-R*, or *--recursive* are given, then rm prompts the user for whether to proceed with the entire operation. If the response is not affirmative, the entire command is aborted.

Otherwise, if a file is unwritable, standard input is a terminal, and the *-f* or *-force* option is not given, or the *-i* or *--interactive=always* option is given, **rm** prompts the user for whether to remove the file. If the response is not affirmative, the file is skipped.

Syntax: rm [OPTION]... FILE...

Output:

```
[root@localhost 20DCE019]# rm 3.txt
[root@localhost 20DCE019]# ls
[root@localhost 20DCE019]#
```

Sort: The sort command is used to sort files in alphabetical order.

Syntax: sort <file name>

Output:

```
[root@localhost 20DCE019]# cat>4.txt
Virat
Pant
Rahul
^Z
[5]+  Stopped(SIGTSTP)      cat > 4.txt
[root@localhost 20DCE019]# sort 4.txt
Pant
Rahul
Virat
[root@localhost 20DCE019]# █
```

Cat: The cat command is also used as a filter. To filter a file, it is used inside pipes.

Syntax: cat <fileName> | cat or tac | cat or tac |. . .

Output:

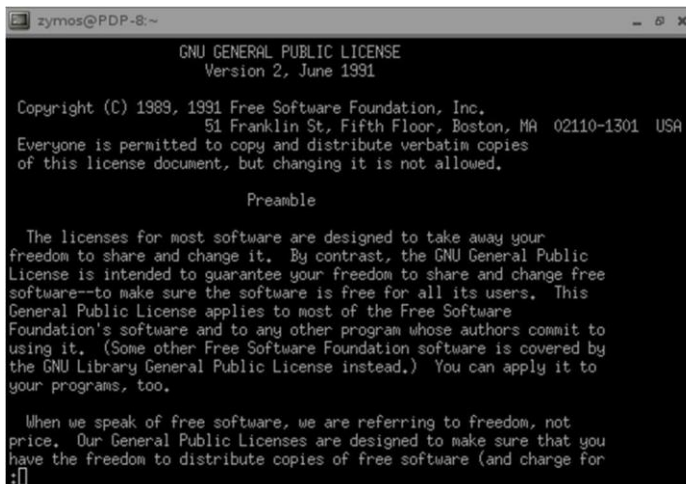
```
[root@localhost 20DCE019]# cat 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]#
```

Pg: pg displays a text file, pausing after each "page" (the height of the terminal screen). After each page, a prompt is displayed. The user may then either press the newline key to view the next page or one of the keys described below.

Syntax:

`pg [-number] [-p string] [-cefnrs] [+line] [+pattern/] [file...]`

Output:



```
zynos@PDP-8:~  
GNU GENERAL PUBLIC LICENSE  
Version 2, June 1991  
  
Copyright (C) 1989, 1991 Free Software Foundation, Inc.  
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA  
Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.  
  
Preamble  
  
The licenses for most software are designed to take away your  
freedom to share and change it. By contrast, the GNU General Public  
License is intended to guarantee your freedom to share and change free  
software--to make sure the software is free for all its users. This  
General Public License applies to most of the Free Software  
Foundation's software and to any other program whose authors commit to  
using it. (Some other Free Software Foundation software is covered by  
the GNU Library General Public License instead.) You can apply it to  
your programs, too.  
  
When we speak of free software, we are referring to freedom, not  
price. Our General Public Licenses are designed to make sure that you  
have the freedom to distribute copies of free software (and charge for  
:[]
```

lp: lp submits files for printing, or alters a pending print job. Use a file name of "-" to specify printing from the standard input.

Syntax:

`lp -i request-ID [-c] [-m] [-p] [-s] [-w] [-d destination] [-f form-name]`



Charotar University of Science and Technology
Devang Patel Institute of Advance Technology and Research
Department of Computer Engineering



`[-H special-handling] [-n number] [-o options] [-p pagenumbers]`

`[-q priority-level] [-S character-set | print-wheel] [-t title]`

`[-T content-Type [-r]] [-y mode-list]`

Output:

```
root@ubuntu:~# apt install lprng
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
dconf-gsettings-backend dconf-service glib-networking glib-networking-common
glib-networking-services gsettings-desktop-schemas libappindicator1
libcairo-gobject2 libcolor2 libdbusmenu-glib4 libdbusmenu-gtk4 libdconf1
libepoxy0 libgtk-3-0 libgtk-3-bin libgtk-3-common libindicator7
libjson-glib-1.0-0 libjson-glib-1.0-common libnautilus-extension1a
libproxy1v5 librest-0.7-0 libsoup-gnome2.4-1 libsoup2.4-1 libwayland-client0
libwayland-cursor0 libwayland-egl1 libxkbcommon0 python-cairo
python-gobject-2 python-gtk2
Use 'apt autoremove' to remove them.
Suggested packages:
magicfilter lprng-doc
The following NEW packages will be installed:
lprng
0 upgraded, 1 newly installed, 0 to remove and 20 not upgraded.
Need to get 558 kB of archives.
After this operation, 3,979 kB of additional disk space will be used.
```

pr: pr paginates or columnates *FILE(s)* for printing.

The **-t** option is implied if **PAGE_LENGTH** is less than or equal to **10**.

Syntax:

pr [*OPTION*]... [*FILE*]...

Output:



Charotar University of Science and Technology
Devang Patel Institute of Advance Technology and Research
Department of Computer Engineering



```
[root@localhost 20DCE019]# pr -2 4.txt
```

```
2022-08-01 06:40
```

```
4.txt
```

```
Page 1
```

```
Virat  
Pant
```

```
Rahul
```

head: The head command is used to display the content of a file. It displays the first 10 lines of a file.

Syntax: head <file name>

Output:

```
[root@localhost 20DCE019]# head 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]#
```

Tail: It is the complementary of head command. The tail command, as the name implies, print the last N number of data of the given input. By default it prints the last 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

Syntax: tail [OPTION]... [FILE]...

Output:

```
[root@localhost 20DCE019]# tail 4.txt
Virat
Pant
Rahul
```


cut: The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field. Basically the cut command slices a line and extracts the text. It is necessary to specify option with command otherwise it gives error. If more than one file name is provided then data from each file is not precedes by its file name.

Syntax: cut OPTION... [FILE]...

Output:

```
[root@localhost 20DCE019]# tail 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]# cut -b 2 4.txt
i
a
a
[root@localhost 20DCE019]#
```

more: The more command is quite similar to the cat command, as it is used to display the file content in the same way that the cat command does. The only difference between both commands is that, in case of larger files, the more command displays screenful output at a time.

Syntax: more <file name>

Output:

```
[root@localhost 20DCE019]# more 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]#
```

find: The find command is used to find a particular file within a directory. It also supports various options to find a file such as byname, by type, by date, and more.

The following symbols are used after the find command:

(.) : For current directory name

Syntax: find. -name "*.pdf"

Output:

```
[root@localhost 20DCE019]# find . 4.txt
.
./4.txt
4.txt
[root@localhost 20DCE019]#
```

COMM: The 'comm' command is used to compare two files or streams. By default, it displays three columns, first displays non-matching items of the first file, second indicates the non-matching item of the second file, and the third column displays the matching items of both files.

Syntax: comm <file1> <file2>

Output:

```
[root@localhost 20DCE019]# comm 4.txt 5.txt
      Virat
    Chahal
Pant
      Rahul
comm: file 2 is not in sorted order
      Hardik
comm: input is not in sorted order
[root@localhost 20DCE019]#
```

Grep: The grep is the most powerful and used filter in a Linux system. The 'grep' stands for "global regular expression print." It is useful for searching the content from a file. Generally, it is used with the pipe.

Syntax: grep <searchWord>

Output:

```
[root@localhost 20DCE019]# grep -i "a" 5.txt
Virat
Chahal
Rahul
Hardik
[root@localhost 20DCE019]#
```

Touch: Update the access and modification times of each FILE to the current time.

Mandatory arguments to long options are mandatory for short options too.

Syntax: Touch <file.name>

Output:

```
[root@localhost 20DCE019]# touch 4.txt
[root@localhost 20DCE019]# ls
4.txt  5.txt
[root@localhost 20DCE019]# rm 5.txt
[root@localhost 20DCE019]# ls
4.txt
[root@localhost 20DCE019]#
```

Tr: The tr command is used to translate the file content like from lower case to upper case.

Syntax: <'old'> <'new'>

Output:

```
[root@localhost 20DCE019]# cat 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]# cat 4.txt | tr "[a-z]" "[A-Z]"
VIRAT
PANT
RAHUL
[root@localhost 20DCE019]#
```

Uniq: The uniq command is used to form a sorted list in which every word will occur only once.

Syntax: command <fileName> | uniq

Output:

```
[root@localhost 20DCE019]# cat 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]# uniq 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]#
```

Cmp: 'cmp' reports the differences between two files character by character, instead of line by line. As a result, it is more useful than 'diff' for comparing binary files. For text files, 'cmp' is useful mainly when you want to know only whether two files are identical. For files that are identical, 'cmp' produces no output.

Syntax: \$ cmp sample.txt sample1.txt

Output:

```
[root@localhost 20DCE019]# cat 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]# cat 6.txt
Virat
Rahul
Hardik
Chahal
[root@localhost 20DCE019]# cmp 4.txt 6.txt
4.txt 6.txt differ: byte 7, line 2
[root@localhost 20DCE019]#
```

Diff: The diff software does not actually change the files it compares. However, it can optionally generate a script (if the -e option is specified) for the program ed or ex which can be used to apply the changes.

Syntax: diff file1.txt file2.txt

Output:

```
[root@localhost 20DCE019]# cat 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]# cat 6.txt
Virat
Rahul
Hardik
Chahal
[root@localhost 20DCE019]# diff 4.txt 6.txt
2d1
< Pant
3a3,4
> Hardik
> Chahal
[root@localhost 20DCE019]#
```

Grep: The grep is the most powerful and used filter in a Linux system. The 'grep' stands for "global regular expression print." It is useful for searching the content from a file. Generally, it is used with the pipe.

Syntax: grep <searchWord>

Output:

```
[root@localhost 20DCE019]# grep -i "Virat" 4.txt
Virat
[root@localhost 20DCE019]# █
```

Touch: Update the access and modification times of each FILE to the current time. Mandatory arguments to long options are mandatory for short options too.

Syntax: Touch <file.name>

Output:

```
[root@localhost 20DCE019]# touch 4.txt
[root@localhost 20DCE019]# ls -l
total 8
-rw-r--r-- 1 root root 17 Aug  1 06:51 4.txt
-rw-r--r-- 1 root root 26 Aug  1 06:49 6.txt
[root@localhost 20DCE019]#
```

Tr: The tr command is used to translate the file content like from lower case to upper case.

Syntax: tr <'old'> <'new'>

Output:

```
[root@localhost 20DCE019]# tr -d 'v'
Virat
irat
Vkohli
kohli
```

Uniq: The uniq is used to form a sorted list in which every word will occur only once.

Syntax: command <fileName> | uniq

Output:

```
[root@localhost 20DCE019]# uniq 4.txt
Virat
Pant
Rahul
[root@localhost 20DCE019]#
```

Security and Protection:

chmod: To change modes of a file.

Syntax: chmod [reference][operator][mode] file

Output:

```
[root@localhost 20DCE019]# ls -l
total 8
-rw-r--r-- 1 root root 17 Aug  1 06:51 4.txt
-rw-r--r-- 1 root root 26 Aug  1 06:49 6.txt
[root@localhost 20DCE019]# chmod +rwx 4.txt
[root@localhost 20DCE019]# ls -l
total 8
-rwxr-xr-x 1 root root 17 Aug  1 06:51 4.txt
-rw-r--r-- 1 root root 26 Aug  1 06:49 6.txt
[root@localhost 20DCE019]#
```

chown: To change group of a file.

Syntax: chgrp [user] [filename.extension]

Output:



Charotar University of Science and Technology
Devang Patel Institute of Advance Technology and Research
Department of Computer Engineering



```
[root@localhost 20DCE019]# ls -l
total 8
-rwxr-xr-x 1 root root 17 Aug  1 06:51 4.txt
-rw-r--r-- 1 root root 26 Aug  1 06:49 6.txt
[root@localhost 20DCE019]# chgrp YC 4.txt
chgrp: invalid group: 'YC'
[root@localhost 20DCE019]# chgrp 5 4.txt
[root@localhost 20DCE019]# ls -l
total 8
-rwxr-xr-x 1 root tty  17 Aug  1 06:51 4.txt
-rw-r--r-- 1 root root 26 Aug  1 06:49 6.txt
[root@localhost 20DCE019]#
```

Date: This command gives current day, date and time

Syntax: date

Output:

```
[root@localhost 20DCE019]# date
Mon Aug  1 06:55:23 AM UTC 2022
[root@localhost 20DCE019]#
```

cal: This command returns current month.

Syntax: cal

Output:

```
[root@localhost 20DCE019]# cal
      August 2022
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
```

who: who command is used to find out the following information:

1. Time of last system boot
2. Current run level of the system
3. List of logged in users and more.

Syntax: \$who [options] [filename]

Output:

```
[root@localhost 20DCE019]# who -m -H
NAME      LINE      TIME      COMMENT
[root@localhost 20DCE019]# w
06:56:12 up 35 min,  0 users,  load average: 0.04, 0.01, 0.00
USER      TTY      LOGIN@  IDLE   JCPU   PCPU   WHAT
[root@localhost 20DCE019]#
```

Tty: It displays information related to terminal. The tty command of terminal basically prints the file name of the terminal connected to standard input. tty is short of teletype, but popularly known as a terminal it allows you to interact with the system by passing on the data (you input) to the system, and displaying the output produced by the system.

Syntax: tty [OPTIONS]

Output:

```
[root@localhost 20DCE019]# tty
/dev/hvc0
[root@localhost 20DCE019]#
```

time: time command in Linux is used to execute a command and prints a summary of realtime, user CPU time and system CPU time spent by executing a command when it terminates. ‘*real*’ time is the time elapsed wall clock time taken by a command to get executed, while ‘*user*’ and ‘*sys*’ time are the number of CPU seconds that command uses in user and kernel mode respectively.

Syntax: time [OPTIONS] [COMMAND]

```
[root@localhost 20DCE019]# time
user      0m6.94s
sys       0m12.48s
[root@localhost 20DCE019]#
```

Bc: bc stands for Basic Calculations. Bc launches a prompt which allows user to perform basic computations.

Syntax: bc

Output:

```
[root@localhost 20DCE019]# 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'.
12+9
21
```

Whoami: This command returns the current username.

Syntax: whoami

Output:

```
[root@localhost 20DCE019]# whoami
root
[root@localhost 20DCE019]# █
```

Which: This command returns the whole directory of the command passed as parameter.

Syntax: which [COMMAND_NAME]

Output:

```
[root@localhost 20DCE019]# which 4.txt
/usr/bin/which: no 4.txt in (/usr/local/sbin:/bin:/sbin:/usr/bin:/usr/sbin:/usr/
local/bin)
[root@localhost 20DCE019]# █
```

Hostname: This command returns the current hostname.

Syntax: hostname

Output:

```
[root@localhost 20DCE019]# hostname
localhost
[root@localhost 20DCE019]#
```

history: history command is used to view the previously executed command. This feature was not available in the Bourne shell. Bash and Korn support this feature in which every command executed is treated as the event and is associated with an event number using which they can be recalled and changed if required. These commands are saved in a history file. In Bash shell history command shows the whole list of the command.

Syntax: \$ history

Output:

```
80 time
81 bc
82 whoami
83 su Yatharth
84 which 4.txt
85 hostname
86 history
[root@localhost 20DCE019]#
```

Wc: WC stands for Word Count. This command returns the number of lines, words and letters used in file passed as parameter.

Syntax: wc [FILENAME]

Output:



Charotar University of Science and Technology
Devang Patel Institute of Advance Technology and Research
Department of Computer Engineering



```
[root@localhost 20DCE019]# wc 4.txt
3  3 17 4.txt
[root@localhost 20DCE019]#
```

System Administrator:

Su: The Unix command su, which stands for 'substitute user', is used by a computer user to execute commands with the privileges of another user account.

Syntax: su [USERNAME]

Output:

```
[root@localhost 20DCE019]# adduser Yatharth
adduser: user 'Yatharth' already exists
[root@localhost 20DCE019]# su Yatharth
[Yatharth@localhost 20DCE019]$ exit
exit
[root@localhost 20DCE019]# login Yatharth
/sbin/init: line 53: 135 Hangup                setsid -c sh -c "exec sh -l <
/dev/$ttyname > /dev/$ttyname 2>&1"
[root@localhost ~]#
```

Adduser: This command allows root user to create another user with restricted privileges.

Syntax: adduser [username]

Output:

```
[root@localhost 20DCE019]# adduser Yatharth
adduser: user 'Yatharth' already exists
[root@localhost 20DCE019]# su Yatharth
[Yatharth@localhost 20DCE019]$ exit
exit
[root@localhost 20DCE019]# login Yatharth
/sbin/init: line 53: 135 Hangup                setsid -c sh -c "exec sh -l <
/dev/$ttyname > /dev/$ttyname 2>&1"
[root@localhost ~]#
```

Rmuser: This command allows root user to delete another user with restricted privileges.

Syntax: rmuser [username]

Output:

```
vivek@nixcraft-asus:~$ id ashish
uid=1002(ashish) gid=1002(ashish) groups=1002(ashish)
vivek@nixcraft-asus:~$ grep '^ashish' /etc/passwd
ashish:x:1002:1002:,,,:/home/ashish:/bin/bash
vivek@nixcraft-asus:~$ sudo userdel -r ashish
userdel: ashish mail spool (/var/mail/ashish) not found
vivek@nixcraft-asus:~$ grep '^ashish' /etc/passwd
vivek@nixcraft-asus:~$ id ashish
id: 'ashish': no such user
vivek@nixcraft-asus:~$
```

© www.cyberciti.biz

fsck: The fsck (File System Consistency Check) Linux utility checks filesystems for errors or outstanding issues. The tool is used to fix potential errors and generate reports.

Syntax: fsck options

<filesystem>

Output:

```
[root@localhost 20DCE019]# fsck
fsck from util-linux 2.36
[root@localhost 20DCE019]#
```


init: init is the parent of all Linux processes. It is the first process to start when a computer boots up and it runs until the system shuts down. It is the ancestor of all other processes.

```
[root@localhost 20DCE019]# init
mount: /proc: none already mounted on /proc.
mount: /sys: none already mounted on /proc.
mkdir: cannot create directory '/dev/pts': File exists
RTNETLINK answers: File exists

Welcome to Fedora 33 (riscv64)

[root@localhost ~]#
```

wall: wall command in Linux system is used to write a message to all users. This command displays a message, or the contents of a file, or otherwise its standard input, on the terminals of all currently logged in users.

Syntax: initwhoma [-n] [-t

timeout] [message | file]

Output:

```
[root@localhost ~]# wall --v
wall from util-linux 2.36
[root@localhost ~]#
```

shutdown: The shutdown command in Linux is used to shutdown the system in a safe way. You can shutdown the machine immediately, or schedule a shutdown using 24-hour format. It brings the system down in a secure way.

Syntax: shutdown [OPTIONS] [TIME] [MESSAGE]

Output:

```
WORD OF THE SESSION: FELICITOUS
MEANING               : SUITABLY EXPRESSED; APPROPRIATE; WELL-CHOSEN

sourcedigit@50:~$ sudo shutdown -h 20:20
[sudo] password for sourcedigit:
Shutdown scheduled for Wed 2018-03-07 20:26:00 IST, use 'shutdown -c' to cancel.
sourcedigit@50:~$
```

mkfs: The mkfs command stands for “make file system” is utilized to make a file system (which is, a system for organizing a hierarchy of directories, subdirectories, and files).

Syntax: mkfs [-V] [-t fstype] [fs-options] filesystem [blocks]

Output:

```
[root@localhost ~]# mkfs
mkfs: no device specified
Try 'mkfs --help' for more information.
[root@localhost ~]# ll /usr/sbin/mkfs*
-rwxr-xr-x 1 root root 12008 Nov 26 2020 /usr/sbin/mkfs
-rwxr-xr-x 1 root root 556896 Nov 28 2020 /usr/sbin/mkfs.btrfs
-rwxr-xr-x 1 root root 24760 Nov 26 2020 /usr/sbin/mkfs.cramfs
-rwxr-xr-x 1 root root 108904 Aug 13 2020 /usr/sbin/mkfs.ext2
-rwxr-xr-x 1 root root 108904 Aug 13 2020 /usr/sbin/mkfs.ext3
-rwxr-xr-x 1 root root 108904 Aug 13 2020 /usr/sbin/mkfs.ext4
-rwxr-xr-x 1 root root 33344 Sep 12 2018 /usr/sbin/mkfs.f2fs
-rwxr-xr-x 1 root root 29152 Aug 14 2020 /usr/sbin/mkfs.fat
-rwxr-xr-x 1 root root 46688 Aug 14 2020 /usr/sbin/mkfs.jffs2
-rwxr-xr-x 1 root root 49544 Aug 15 2020 /usr/sbin/mkfs.jfs
-rwxr-xr-x 1 root root 37440 Nov 26 2020 /usr/sbin/mkfs.minix
lrwxrwxrwx 1 root root 8 Aug 14 2020 /usr/sbin/mkfs.msdos -> mkfs.fat
lrwxrwxrwx 1 root root 16 Aug 13 2020 /usr/sbin/mkfs.ntfs -> /usr/sbin/mkntfs
-rwxr-xr-x 1 root root 70368 Aug 14 2020 /usr/sbin/mkfs.ubifs
lrwxrwxrwx 1 root root 7 Aug 14 2020 /usr/sbin/mkfs.udf -> mkudffs
lrwxrwxrwx 1 root root 8 Aug 14 2020 /usr/sbin/mkfs.vfat -> mkfs.fat
-rwxr-xr-x 1 root root 342016 Aug 15 2020 /usr/sbin/mkfs.xfs
```

mount: mount command is used to mount the filesystem found on a device to big tree structure(Linux filesystem) rooted at '/'. Conversely, another command umount can be used to detach these devices from the Tree.

Syntax: mount -t type device dir

Output:

```
[root@localhost ~]# mount
root on / type 9p (rw,relatime,dirsync,mmap,access=client,trans=virtio)
devtmpfs on /dev type devtmpfs (rw,relatime,size=93120k,nr_inodes=23280,mode=755)
none on /proc type proc (rw,relatime)
none on /sys type sysfs (rw,relatime)
devpts on /dev/pts type devpts (rw,relatime,mode=600,ptmxmode=000)
devpts on /dev/pts type devpts (rw,relatime,mode=600,ptmxmode=000)
[root@localhost ~]#
```

umount: This command enables you to remove a remote file system that is currently mounted. The umount command supports the -V option to allow for testing.

Syntax: umount [options] filesystem

Output:

```
dave@howtogeek:~$ sudo umount /media/dave/isomnt
dave@howtogeek:~$
dave@howtogeek:~$ ls iso
boot  cde
dave@howtogeek:~$ sudo umount iso
dave@howtogeek:~$
dave@howtogeek:~$ ls iso
dave@howtogeek:~$
```

dump: dump command in Linux is used for backup the filesystem to some storage device. It backs up the complete file system and not the individual files. In other words, it backups the required files to tape, disk or any other storage device for safe storage.

Syntax:

```
dump [-level#] [-a autosize] [-A file] [-B records] [-b blocksize] [-  
d density] [-D file] [-e inode numbers] [-E file] [-f file]  
[-F script] [-h level] [-I nr errors] [-jcompression level] [-L label] [-  
Q file] [-s feet] [-T date] [-y] [-zcompression level] files-to-dump  
dump [-W | -w]
```

Output:

```
anubhav@anubhav-ThinkPad-Yoga-460:~$ dump  
dump 0.4b44 (using libext2fs 1.42.13 of 17-May-2015)  
usage: dump [-level#] [-acmMnqSuv] [-A file] [-B records] [-b blocksize]  
[-d density] [-D file] [-e inode#,inode#,...] [-E file]  
[-f file] [-h level] [-I nr errors] [-j zlevel] [-Q file]  
[-s feet] [-T date] [-y] [-z zlevel] filesystem  
dump [-W | -w]  
anubhav@anubhav-ThinkPad-Yoga-460:~$
```

restore: restore command in Linux system is used for restoring files from a backup created using dump. The restore command performs the exact inverse function of dump. A full backup of a file system is being restored and subsequent incremental backups layered is being kept on top of it.

Syntax:

```
restore -C [-cdHklMvVy] [-b blocksize] [-D filesystem] [-f file] [-F script] [-L limit] [-s fileno]  
[T directory]
```

Output:

```
rahul@rahul-SVF15318SNB:~/Desktop/prashant/Chrome-Extensions$ restore
restore 0.4b46 (using libext2fs 1.44.1 of 24-Mar-2018)
usage: restore -C [-cdeHLMuvVy] [-b blocksizes] [-D filesystem] [-E mls]
        [-f file] [-F script] [-L limit] [-s fileno]
restore -i [-acdehHLMuvVy] [-A file] [-b blocksizes] [-E mls]
        [-f file] [-F script] [-Q file] [-s fileno]
restore -P file [-acdHLMuvVy] [-b blocksizes]
        [-f file] [-F script] [-s fileno] [-X filelist] [file ...]
restore -r [-cdeHLMuvVy] [-b blocksizes] [-E mls]
        [-f file] [-F script] [-s fileno] [-T directory]
restore -R [-cdeHLMuvVy] [-b blocksizes] [-E mls]
        [-f file] [-F script] [-s fileno] [-T directory]
restore -t [-cdhHLMuvVy0] [-A file] [-b blocksizes]
        [-f file] [-F script] [-Q file] [-s fileno] [-X filelist] [file ...]
restore -x [-acdehHLMuvVy] [-A file] [-b blocksizes] [-E mls]
        [-f file] [-F script] [-Q file] [-s fileno] [-X filelist] [file ...]
```

tar: The Linux ‘tar’ stands for tape archive, is used to create Archive and extract the Archive files. tar command in Linux is one of the important command which provides archiving functionality in Linux. We can use Linux tar command to create compressed or uncompressed Archive files and also maintain and modify them.

Syntax: tar [options] [archive-file] [file or directory to be archived]

Output:

```
[root@localhost 20DCE019]# tar cvf txt.tar *.txt
4.txt
6.txt
[root@localhost 20DCE019]#
```

Terminal:

Echo: This command allows user to print the string (passed as parameter) with new line character on the console.

Syntax: echo [STRING]

Output:

```
[root@localhost 20DCE019]# echo Yatharth
Yatharth
[root@localhost 20DCE019]#
```

printf: This command allows user to print the string (passed as parameter) without new line character on the console.

Syntax: printf [STRING]

Output:

```
[root@localhost 20DCE019]# printf "Yatharth Chauhan - 20DCE019\n"
Yatharth Chauhan - 20DCE019
```

Clear: This command clears the terminal screen and provides blank screen to user.

Syntax: clear

Output:

```
82 whoami
83 su Yatharth
84 which 4.txt
85 hostname
86 history
87 wc 4.txt
88 adduser Yatharth
89 su Yatharth
90 login Yatharth
91 wall --v
92 mount
93 cd 20DCE019
94 MOUNT
95 tar cvf txt.tar *.txt
96 echo Yatharth
97 printf "Yatharth Chauhan - 20DCE019\n"
98 history
[root@localhost 20DCE019]#
```

```
[root@localhost 20DCE019]#
```


Process:

ps: Linux provides us a utility called **ps** for viewing information related with the processes on a system which stands as abbreviation for “**Process Status**”. ps command is used to list the currently running processes and their PIDs along with some other information depends on different options.

Syntax: ps

[options]

Output:

```
[root@localhost 20DCE019]# ps
  PID TTY          TIME CMD
   350 hvc0      00:00:01 sh
   378 hvc0      00:00:00 ps
[root@localhost 20DCE019]#
```

Pipe (|): This command executes multiple commands simultaneously.

Syntax: [COMMAND1] | [COMMAND2] | ...

Output:

```
[root@localhost 20DCE019]# ls -l | more
total 20
-rwxr-xr-x 1 root tty      17 Aug  1 06:51 4.txt
-rw-r--r-- 1 root root    26 Aug  1 06:49 6.txt
-rw-r--r-- 1 root root 10240 Aug  1 07:06 txt.tar
[root@localhost 20DCE019]#
```

kill -l :To display all the available signals you can use below command option:

Syntax: \$kill -l

Output:

```
[root@localhost 20DCE019]# kill -l
HUP INT QUIT ILL TRAP ABRT BUS FPE KILL USR1 SEGV USR2 PIPE ALRM TERM STKFLT CHLD
CONT STOP TSTP TTIN TTOU URG XCPU XFSZ VTALRM PROF WINCH IO PWR SYS RTMIN RTMI
N+1 RTMIN+2 RTMIN+3 RTMIN+4 RTMIN+5 RTMIN+6 RTMIN+7 RTMIN+8 RTMIN+9 RTMIN+10 RTM
IN+11 RTMIN+12 RTMIN+13 RTMIN+14 RTMIN+15 RTMAX-14 RTMAX-13 RTMAX-12 RTMAX-11 RT
MAX-10 RTMAX-9 RTMAX-8 RTMAX-7 RTMAX-6 RTMAX-5 RTMAX-4 RTMAX-3 RTMAX-2 RTMAX-1 R
TMAX
[root@localhost 20DCE019]# █
```

exec: exec command in Linux is used to execute a command from the bash itself. This command does not create a new process it just replaces the bash with the command to be executed. If the exec command is successful, it does not return to the calling process.

Syntax: exec [-cl] [-a name] [command [arguments]] [redirection ...]

Output:

```
[root@localhost 20DCE019]# exec ls
4.txt 6.txt txt.tar
█
```

I/O Redirection (<,>,>>): Redirection can be defined as changing the way from where commands read input to where commands sends output. You can redirect input and output of a command.

Overwrite: Commands with a single bracket '>' overwrite existing file content.

> : standard output

< : standard input

2> : standard error

Syntax: cat > <fileName>

Output:

```
[root@localhost ~]# mkdir 20DCE019
[root@localhost ~]# cd 20DCE019
[root@localhost 20DCE019]# cat>a.txt
hello
everyone
```

Append: Commands with a double bracket '>>' do not overwrite the existing file content.

>> - standard output

<< - standard input

2>> - standard error

Syntax: cat >> <fileName>

Output:

```
[root@localhost 20DCE019]# cat >> a.txt
hello
guys
^Z
[2]+  Stopped(SIGTSTP)      cat >> a.txt
[root@localhost 20DCE019]# cat a.txt
hello
everyone
hello
guys
[root@localhost 20DCE019]#
```

Gcc: GCC stands for GNU Compiler Collections which is used to compile mainly C and C++ language. It can also be used to compile Objective C and Objective C++. The most important option required while compiling a source code file is the name of the source program, rest every argument is optional like a warning, debugging, linking libraries, object file etc. The different options of gcc command allow the user to stop the compilation process at different stages.

Syntax: gcc [C_Source] [OPTION] [COMMAND]

Output:

```
localhost:~# gcc hello.c -o hello
localhost:~# ./hello
hello world
```