

1. *echo hello world*

```
sowmen@Crimson-pc:~$ echo hello world
hello world
sowmen@Crimson-pc:~$
```

The “*echo*” command writes to the terminal everything that is typed after it.

2. It is the basic command to search manual.

3. Use of various linux commands :

- ***man:***

The *man* is an interface to the on-line reference manuals. *Man* is called with an argument which is any unix command or function. The *man* call returns the manual page for the argument function.

Example : *man date*. This returns the manual page for the *date* command.

```
DATE(1) User Commands
NAME
    date - print or set the system date and time
SYNOPSIS
    date [OPTION]... [+FORMAT]
    date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
DESCRIPTION
    Display the current time in the given FORMAT, or set the system date.

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

    -d, --date=STRING
        display time described by STRING, not 'now'

    --debug
Manual page date(1) line 1 (press h for help or q to quit)
```

- ***who:***

who shows the user who is logged in. Example :

```
sowmen@Crimson-pc:~$ who
sowmen    :1                2019-03-30 23:20 (:1)
sowmen    pts/0            2019-03-30 17:31 (C)
```

- **cat:**

cat concatenates files and shows on the standard output. *cat* is followed by the file names. It reads the data from the given files and writes to the console.

```
sowmen@Crimson-pc:~$ cat file1.txt file2.txt
This is the content of file 1
Content of file2
```

- **cd:**

cd changes from the current directory to the given directory. It takes as an argument the destination directory. *Cd* with no argument takes to the root directory.

```
sowmen@Crimson-pc:~$ cd Downloads/
sowmen@Crimson-pc:~/Downloads$
```

- **cp:**

cp copies files and directories. Its is used as *cp [Source] [Directory]*. If source and directory are both files then it copies the contents of the source file to directory file. If source is a file and destination is a directory, it copies the file to the directory.

```
sowmen@Crimson-pc:~$ cp file1.txt Downloads/
sowmen@Crimson-pc:~$ ls Downloads/
file1.txt
sowmen@Crimson-pc:~$
```

- **ps:**

ps returns a snapshot of the current running processes.

```
sowmen@Crimson-pc:~$ ps
  PID TTY          TIME CMD
 2537 pts/0    00:00:00 bash
 4539 pts/0    00:00:00 ps
sowmen@Crimson-pc:~$
```

- **ls:**

ls returns the list of all files and folders in the current directory.

```
sowmen@Crimson-pc:~$ ls
Desktop    Downloads  file2.txt  Music     Videos   模板
Documents  file1.txt  google-chrome  Pictures  yolo.cpp
sowmen@Crimson-pc:~$
```

- **mv:**

mv [Source] [Destination] moves the source folder/file to the destination folder/file.

```
sowmen@Crimson-pc:~$ mv file1.txt Downloads/  
sowmen@Crimson-pc:~$ ls Downloads/  
file1.txt  
sowmen@Crimson-pc:~$
```

- **rm:**

rm [File-name] deletes the file or directory of the argument.

```
sowmen@Crimson-pc:~$ ls  
Desktop    Downloads  file2.txt  google-chrome  Pictures  yolo.cpp  
Documents  file1.txt  file3.txt  Music          Videos  模板  
sowmen@Crimson-pc:~$ rm file3.txt  
sowmen@Crimson-pc:~$ ls  
Desktop    Downloads  file2.txt  Music          Videos  模板  
Documents  file1.txt  google-chrome  Pictures  yolo.cpp  
sowmen@Crimson-pc:~$
```

- **mkdir:**

mkdir [Directory-Name] creates a new directory in the current folder.

```
sowmen@Crimson-pc:~$ mkdir NewDirectory  
sowmen@Crimson-pc:~$ ls  
Desktop    Downloads  file2.txt  Music          Pictures  yolo.cpp  
Documents  file1.txt  google-chrome  NewDirectory  Videos  模板  
sowmen@Crimson-pc:~$
```

- **rmdir:**

rmdir [Directory-Name] removes the specified directory.

- **echo:**

echo [String] writes to the console everything written after it.

- **more:**

more command is used to view the text files in the command prompt, displaying one screen at a time in case the file is large.

```
[nazia@appledore ~]$ who >myfile.txt
[nazia@appledore ~]$ more myfile.txt
nazia      :1                2019-03-30 23:46 (:1)
nazia      pts/0            2019-03-30 23:46 (a)
nazia      pts/1            2019-03-31 01:28 (a)
myfile.txt (END)
```

- **less:**

less command is used to view files instead of opening the file.

```
nazia      :1                2019-03-30 23:46 (:1)
nazia      pts/0            2019-03-30 23:46 (a)
nazia      pts/1            2019-03-31 01:28 (a)
myfile.txt (END)
```

- **date:**

the **date** command displays the current date and time. It can also be used to display or calculate a date in a format specified.

```
[nazia@appledore ~]$ date
Mon Apr  1 03:18:20 +06 2019
```

- **time:**

the **time** command is used to determine how long a given command takes to run. It is useful for testing the performance of scripts and commands.

```
[nazia@appledore ~]$ time

real    0m0.000s
user    0m0.000s
sys     0m0.000s
```

- **kill :**

kill command in Linux (located in `/bin/kill`), is a built-in command which is used to terminate processes manually.

- **history:**

history command is used to extract or print the *commands* which was executed by users in Bash shell.

```
[nazia@appledore ~]$ history
57  reboot
58  sudo pacman install -y git curl wget zip unzip
59  pamac-manager
60  :qw
61  clear
62  apache2 --version
63  pacman -Syu
64  sudo pacman -Syu
65  pacman -S apache
66  sudo pacman -S apache
67  nano /etc/httpd/conf/httpd.conf
68  nano /etc/httpd/conf/httpd.conf
69  systemctl status httpd
70  systemctl enable httpd
71  systemctl status httpd
72  systemctl enable httpd
73  systemctl restart httpd
```

- **chmod:**

chmod command is used to change permissions of a given file according to a certain mode which might be a set of octal characters or a set of alphabetical characters.

chmod [mod] [filename]

- **chown:**

The *chown command* is used to change the owner and group of files, directories and links.

- **finger:**

finger is a program used to find information about computer users. It usually **lists** the login name, the **full** name, and possibly other details about the user .

- **pwd:**

command '**pwd**' prints the current working directory or simply the directory user is, at present. It prints the current directory name with the complete path starting from root (/)

```
[nazia@appledore ~]$ pwd
/home/nazia
```

- **cal:**

cal displays a simple calendar. If no arguments are specified, the current month is displayed.

```
[nazia@appledore ~]$ cal
  April 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
```

- **logout:**

Logout of a login shell. This command can be used by normal users to end their own session.

- **Shutdown:**

*The **shutdown** command brings the system down in a secure way*

4. **sed** stands for Stream Editor. It can perform functions like searching, find and replace, insertion, deletion on files without opening them. **grep** searches a file for a particular pattern of characters and displays the lines.

- Remove first and last character of every line

```
sed 's/^./;/s/.$//' file.txt
```

's' means substitution. 's/^./' replaces the first character of every line with " " an empty character.

's/.\$/' replaces the last character of every line with an empty character. Two commands are concatenated together with a (;).

- How many lines of a file contain a given word

grep -c 'pattern' file.txt

'-c' flag is used to get the count of the number of lines that contains the 'pattern' string.

awk is a scripting language used for manipulating data and generating reports. The *awk* command programming language requires no compiling, and allows the user to use variables, numeric functions, string functions, and logical operators.

5. This task requires us to move the output of the 'who' command into a text file called 'myfile1.txt'. The result is shown below:

```
sowmen@Crimson-pc:~/Documents$ who > myfile1.txt
sowmen@Crimson-pc:~/Documents$ more myfile1.txt
sowmen    :1                2019-03-30 23:20 (:1)
sowmen    pts/0            2019-03-30 18:31 (C)
sowmen@Crimson-pc:~/Documents$
```

6. Here we use both the 'who' directive and 'date' directive in the same line and redirect the outputs of 'who' to text file 'myfile2.txt'

```
sowmen@Crimson-pc:~/Documents$ who > myfile2.txt | date
Sat Mar 30 19:26:37 +06 2019
sowmen@Crimson-pc:~/Documents$ more myfile2.txt
sowmen    :1                2019-03-30 23:20 (:1)
sowmen    pts/0            2019-03-30 18:31 (C)
sowmen@Crimson-pc:~/Documents$
```

7. **sed** command works as a stream editor which capable of modifying a text document. The following sed command can swap the first and second words of each line in a document :

`sed -e "s/\([^]*\) \([^]*\) / \2 \1/" [filename.txt]`

```
sowmen@Crimson-pc:~$ more file.txt
is this the song that never ends
it yes, goes on and on, my friend
they'll and continue singing it forever
because... just
sowmen@Crimson-pc:~$ sed 's/\([^ ]*\) \([^ ]*\) / \2 \1/' file.txt
this is the song that never ends
yes, it goes on and on, my friend
and they'll continue singing it forever
just because...
```

8. **The runtime of a bash command is more than that of a C program.** The 'real' time shows the runtime of the command and the program based of the system clock. 'Sys' time shows the number of cpu cycles it used. In all cases a C program runs much faster than a bash command.

```
sowmen@Crimson-pc:~/Documents$ time sh world.sh
hello world

real    0m0.016s
user    0m0.015s
sys     0m0.000s
sowmen@Crimson-pc:~/Documents$ time ./world
Hello World

real    0m0.003s
user    0m0.002s
sys     0m0.001s
```

9. The following script takes input as a string and determines whether the given string leads to a executable file or a directory. It also informs in case the directory doesn't exist. The output is given in the second photo.

```
1 str=$1
2 if [ -f $str ]
3 then
4     echo $str is a file
5 elif [ -d $str ]
6 then
7     echo $str is a directory
8 else
9     echo $str is neither a file nor directory
10 fi
```

```
sowmen@Crimson-pc:~/Documents$ sh identify.sh directory/
directory/ is a directory
sowmen@Crimson-pc:~/Documents$ sh identify.sh myfile.txt
myfile.txt is a file
sowmen@Crimson-pc:~/Documents$ sh identify.sh hello/
hello/ is neither a file nor directory
sowmen@Crimson-pc:~/Documents$
```


10. Shell script to convert filenames from arguments from lowercase to Uppercase.

```
for var in "$@"
do
    if [ ! -f $var ]; then
        echo "$var is not a file"
    else
        basename=$(tr 'a-z' 'A-Z' <<< "${var%.*}")
        newname="$basename.${var#*.*}"
        echo "$var --> $newname"
        mv "$var" "$newname"
    fi
done
```

```
sowmen@Crimson-pc:~/Documents$ sh rename.sh hello.txt yolo.txt
hello.txt --> HELLO.txt
yolo.txt is not a file
```

11. The *last command* is used to show who has recently used the server and logged in and out date/time. The *last command* reads listing of *last* logged in users from the system file called */var/log/wtmp*

```
period.sh
echo ENTER USERNAME

read $username

last $username
```

Following is the output for the shell that determines the period for which a specified user is working on the system:

```
[nazia@appledore Documents]$ sh period.sh
ENTER USERNAME
appledora
nazia pts/1 a Sun Mar 31 01:28 still logged in
nazia pts/2 a Sun Mar 31 00:46 - 00:50 (00:03)
nazia pts/1 a Sun Mar 31 00:04 - 01:28 (01:23)
nazia pts/0 a Sat Mar 30 23:46 still logged in
nazia :1 :1 Sat Mar 30 23:46 gone - no logout
reboot system boot 4.14.94-1-MANJAR Sat Mar 30 23:44 still running
nazia pts/0 a Thu Mar 28 22:11 - down (23:49)
nazia :1 :1 Thu Mar 28 22:10 - 22:01 (23:50)
reboot system boot 4.14.94-1-MANJAR Thu Mar 28 22:10 - 22:01 (23:51)
nazia pts/1 a Sun Mar 24 18:01 - down (01:18)
nazia pts/0 a Sun Mar 24 17:12 - down (02:07)
nazia :1 :1 Sun Mar 24 17:12 - down (02:07)
reboot system boot 4.14.94-1-MANJAR Sun Mar 24 17:11 - 17:20 (02:09)
```

12. The following program takes 3 arguments, a Filename, starting line and ending line and outputs the range of lines from the file if the file exists.

```
1  #!/bin/bash
2  FILE=$1
3  start=$2
4  end=$3
5
6  if [ ! -f $FILE ];then
7      echo "$FILE is not a valid file"
8      exit 1
9  fi
10
11  str=`wc -l $FILE`
12  array=( $str )
13  totalLines=${array[0]}
14
15  if [[ $end -le $totalLines && $totalLines -gt 0 ]];then
16      sed -n "${start},${end}p" $FILE
17  else
18      echo "Range out of bounds"
19  fi
```

```
sowmen@Crimson-pc:~/Documents$ more file.txt
is this the song that never ends
it yes, goes on and on, my friend
they'll and continue singing it forever
because... just
sowmen@Crimson-pc:~/Documents$ sh lineCount.sh file.txt 2 3
it yes, goes on and on, my friend
they'll and continue singing it forever
sowmen@Crimson-pc:~/Documents$
```

13. The following bash scripts takes a string-pattern, filename and an output filename as arguments and produces output.

```
1  PATTERN=$1
2
3  for file in "${@:2}"
4  do
5      if [ ! -f $file ];
6      then echo $file DOES NOT EXIST
7      continue
8      fi
9      echo "Delete occurrences of '$PATTERN' in '$file'"
10     grep -v "$PATTERN" $file
11 done
12
```

```
sowmen@Crimson-pc:~/Documents$ sh deletpattern.sh on file.txt myfile.txt file1.txt appl.txt
Delete occurances of 'on' in 'file.txt'
because... just
Delete occurances of 'on' in 'myfile.txt'
sowmen :1 2019-03-24 17:36 (:1)
sowmen pts/1 2019-03-24 11:56 (C)
sowmen pts/2 2019-03-24 11:56 (C)
file1.txt DOES NOT EXIST
appl.txt DOES NOT EXIST
```

14. Following are shell scripts for :

- Counting the length of a string

```
stringlenght.sh
read -p 'Given String: ' string

Length=`expr length "$string"`
echo String lenth: $Length
```

```
[nazia@appledore Documents]$ sh stringlenght.sh
Given String: hello world
String lenth: 11
[nazia@appledore Documents]$ _
```

- Extract a substring from a given string

```
substr.sh
1 read -p "Given String: " str
2 read -p "Starting Index: " index
3 read -p "Char length: " len
4
5
6 echo "Extracted string : ${str:index:len}"
```

```
[nazia@appledore Documents]$ sh substr.sh
Given String: I find the syntax of bash scripting very confusing
Starting Index: 3
Char length: 7
Extracted string : ind the
[nazia@appledore Documents]$ _
```