

Operating System LAB 01

3.

man who

who

displays who is logged on

```
eesha  :0      2019-03-27 19:15 (:0)
```

who -a, --all

displays all information who is logged on

```
system boot 2019-03-28 01:14
```

```
eesha  ?:0      2019-03-27 19:15 ?      1920 (:0)
```

```
run-level 5 2019-03-27 19:15
```

who -b, --boot

time of last system boot

```
system boot 2019-03-28 01:14
```

who -l, --login

print system login processes

who --version, -V

display version

```
who (GNU coreutils) 8.28
```

```
Copyright (C) 2017 Free Software Foundation, Inc.
```

```
License GPLv3+: GNU GPL version 3 or later
```

```
<http://gnu.org/licenses/gpl.html>.
```

```
This is free software: you are free to change and redistribute it.
```

```
There is NO WARRANTY, to the extent permitted by law.
```

Written by Joseph Arceneaux, David MacKenzie, and Michael Stone.

who --message

user's message status

```
eesha  ?:0      2019-03-27 19:15 (:0)
```

man cat

cat actually concatenate files and print on the standard output

```
# cat myfile1 myfile2 myfile3
```

where myfile1, myfile2 and myfile3 are all files
this command prints all stuffs within the files

this is inside myfile1
this is inside myfile2
this is inside myfile3

```
# cat -A myfile1 or cat --show-all myfile1
```

this command shows all stuffs inside myfile1

this is inside myfile1\$

```
# cat -n myfile1 myfile2 myfile3
```

-n prints all output lines

```
1    this is inside myfile1
2    this is inside myfile2
3    this is inside myfile3
```

cd

cd stands for 'Change Directory'

```
# cd directory
```

it goes on that directory
directory – Documents/new/final

```
# cd..
```

goes back to previous folder

```
# cd /
```

goes back to root directory

man cp

cp - copy
cp means copy files and directories

cp hello helloCopy
where hello source file, helloCopy destination file. If helloCopy don't exists, system will create helloCopy file. If destination file exists on system and has contents on that, cp commands will replace that contents with source contents. That means all contents of destination file will be removed and then replaced.

```
# cp hello /home/eesha/Documents/helloCopy
```

cp hello.docx helloCopy.docx
if file name has extension, then it should be given.

cp -a hello.docx copy.docx
-a means all. As same as above all commands.

cp hello /home/eesha/Music
in Music Folder, system will create hello file.

cp -t /home/eesha/Music hello
-t means target. In Music folder, system will create hello if don't exists. If hello file already exists, command will abort.

man ps

ps means **Process Status**
This command will display a snapshot of current processes.

It displays (as current user) the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD). Output is unsorted by default.

To see every process on the system using standard syntax:

```
# ps -e  
displays  
PID  TTY  TIME  CMD
```

-e means Select all processes. As same as **-a**

ps -ef
displays

UID PID PPID C STIME TTY TIME CMD

PPID stands for Parent Process ID

C stands for integer value of the processor utilization percentage.

STIME means Start TIME of process.

TIME means total time of that process which utilizes processor.

CMD means Commands.

-f means do full-format listing. This option can be combined with many other UNIX-style options to add additional columns.

To see every process on the system using BSD syntax:

ps ax or ps axu

PID TTY STAT TIME COMMAND

STAT means multi-character process state.

a causes **ps** to list all processes with a terminal (tty), or to list all processes when used together with the **x** option.

x causes ps to list all processes when used together with the **a** option.

u means displaying user-oriented format.

To print a process tree:

ps -ejH

PID PGID SID TTY TIME CMD

PGID means Process Group ID.

RGID means Real Group ID.

SID means Session ID.

-j means BSD (Berkeley Software Distribution) jobs format.

-o means User defined format.

-H means show processes hierarchy (forest).

```
# ps -eZ
```

```
LABEL  PID  TTY  TIME  CMD
```

Z means add a column of security data. Where is LABEL.

To see every process running as root:

```
# ps -U root -u root u
```

displays

```
PID  TTY  TIME  CMD
```

-U means real USER ID (RUID)

-u means effective User ID (EUID)

Others:

```
# ps H
```

```
PID  TTY  STAT  TIME  COMMAND
```

H shows threads as if they were processes.

man ls

ls means list directory contents.

```
# ls
```

displays list of files and folders of current directory.

```
# ls /home/eesha/Documents or ls Documents ( iff Documents exists in  
current directory)
```

displays list of files and folders of Documents.

```
# ls -a or ls --all or ls -a /home/eesha/Documents
```

do not ignore entries starting with .

```
# ls -A or ls --almost-all or ls --almost-all /home/eesha/Documents
```

do not display which contains only - or --.

```
# ls -c or ls -t or ls -u
```

displayed list sorted by ctime, newest first.

```
# ls -c -i
```

displays list sorted by ctime also including index number of each file.

-i means index number of each file.

ls -l

use a long listing format

ls -r or ls --reverse

reverse order while sorting. **Reverse of -c (kind of)**

ls -R

list sub directories recursively. Sub directories means all the files and folders of current directory, then all files and folders of each folder in current directory, and then, and then.

It is a very very fascinating command I ever have seen.

ls -R Documents

displays

Documents:

d first.sql hello users.sql

Documents/d:

dd

Documents/d/dd:

(dd have nothing)

ls -s -S

-s means print the allocated size of each file.

-S means sort by file size, descending, largest first

take a look:

total 52

12 examples.desktop	4 Downloads	4 Public	4 Videos
4 Desktop	4 Music	4 snap	4 virtualClassroom
4 Documents	4 Pictures	4 Templates	

ls --sort=none/size/time/version/extension

all in one package. Good command.
Where `ls --sort=time == ls -t` or `ls -c`

touch

`touch /home/eesha/Documents sust`
create a file named 'sust', just a text file.

man mv

`mv` means move or rename

`mv sust hello`

where `sust` is source plain text file and `hello` is destination plain text file. If destination file exists, contents of source will be overwritten into destination file. If doesn't exist, 'hello' named plain text file will be created then moved into it.

`mv sust.docx hello`

here `sust` is `docx` and `hello` is plain text. Source file's will be destroyed. Be careful with extension. It should be
`mv sust.docx hello.docx`

`mv hello /home/eesha/Documents`

here, in Documents folder 'hello' source file will be moved.

`mv -t /home/eesha/Documents hello`

As same as Above.

`mv hello /home/eesha/Documents/world`

here, `world` will be overwritten by `hello` source file. That means, all contents of `hello` will be overwritten into `world` and contents of `world` will be destroyed.

`mv -n hello Documents/world`

this command doesn't work. `-n` means don't overwrite an existing file.

But

`mv -n hello Documents`

This will work iff in Documents folder, there is no 'hello' named file.

man rm

rm means remove

rm hello
removes hello (if it is a file)

rm -f hello
-f means force, never prompt

rm -d empDirec
where empDirec is a empty directory

rm -r director or rm -R director or rm --recursive director
where director is directory will sub directories (not empty). Remove directories and their contents recursively.

rm -i -r director
where -i means prompt before every removal. And you have to give y/ yes to every removal of file or folder.

man mkdir

mkdir means make directories.

mkdir hello
which makes a folder named 'hello'.

mkdir -p hello
no error if existing

mkdir -v hellod
print a message for each created directory

man rmdir

rmdir means remove empty directories

rmdir hello
removes 'hello' directory iff it is empty.

man more

more shows what is actually exists in a file.

more hello
views what is in hello. Only plain text type file is viewed clearly. Odt, docx, xlsxx will not be viewed with this command.

man less

All manual pages are viewed according to less command.

less is opposite of more. But it has many features than more. With large input files it starts up faster than text editors like **vim**.

less hello
where hello is plain text file.

ENTER means RETURN.

For scrolling forward N lines,

1. press f or SPACE BAR or z. Default one window size.
2. e or ENTER or j, it scrolls line by line. I prefer this.
3. d, default one half of the screen.

For scrolling backward N lines,

1. b or w, default one window.
2. y, scrolls back line by line. I prefer this one.
3. u, default one half of the screen.

Right Arrow: To scroll horizontally right N characters, default half screen width.

Left Arrow: To Scroll horizontally left N characters, default half screen width.

g : Beginning of the file.

G : End of the file.

' (Single quote.) Followed by pressing it twice, returns to the position which was previously marked with that letter.

less -p hel hello

This commands will find 'hel' word in every containing line of hello and highlight.

Or

On reading screen,

type **/pattern**, which finds pattern word on the file.

Example: /hello which finds all 'hello' with highlight.

/!pattern, which will find DO NOT match stuffs.

/pattern* which will search pattern in multiple files.

?pattern works like /pattern

&pattern displays only lines which match the pattern.

q or Q or :q or :Q or ZZ that exits less.

Line Editing on VIM or LESS:

Vim is an editor to create or edit a text file.

There are two modes in vim. One is the command mode and another is the insert mode.

In the command mode, user can move around the file, delete text, etc.

In the insert mode, user can insert text.

From command mode **to** insert mode type **a/A/i/I/o/O.**

From insert mode **to** command mode type **Esc**

Some Commands:

a Append text following current cursor position

A Append text to the end of current line

i Insert text before the current cursor position

I Insert text at the beginning of the cursor line

o/O Open up a new line following the current line and add text there

h Moves the cursor one character to the left

l Moves the cursor one character to the right

k Moves the cursor up one line

j Moves the cursor down one line

^U Up half screenful

^D Down half screenful

x Delete character

man date

date print the system's date and time

date

Sun Mar 31 05:33:10 +06 2019

date -d now or today or tomorrow or Sunday or last-sunday
-d display time described by STRING.

date -d "1997-09-12"
Fri Sep 12 00:00:00 +06 1997

date +%s
%s stands for seconds since **1970-01-01 00:00:00 UTC**
1553989228 (it is total seconds since 1970-01-01 to current time)

date -d "1974-01-04" +%s (no space after +)
126468000

man time

time means run programs and summarize system resource usage.

type time
time is a shell keyword

We will work where time is a shell keyword.

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

where real time is the time from start to finish of the call. It is the time from the moment you hit the Enter key until the moment the **task** command is completed.

user - amount of CPU time spent in **user mode**.
system or sys - amount of CPU time spent in **kernel mode**.

time options command arguments

```
# time -p mkdir newfile
```

-p means time format will be printed in POSIX format.
How much time a CPU takes to create a new directory.

```
real 0.00
```

```
user 0.00
```

```
sys 0.00
```

```
# time df
```

Calculates free disk space using the **df** command and reports how long it took for the command to complete.

```
Filesystem 1K-blocks Used Available Use% Mounted on
```

```
# time wc hello
```

```
70 255 1389 hello
```

where 70 is number of lines.

255 is the number of words.

1389 is the number of bytes.

@ **wc** prints newline, word and byte counts for each file.

man kill

kill command actually sends a signal to a process of kernel

kill options PID

```
# kill -9 -1
```

which **kill** all processes you can kill.

-9 is alternate signal. (option)

PID -1 indicates all processes except the kill process itself and initial.

Negative PID values may be used to choose whole process groups.

```
# kill -l 11
```

Translate number 11 into a signal name.

```
SEGV
```

```
# kill -L
```

List signal names in a nice table.

`#kill 123`

Send the default signal to that process.

man history

history command prints all stuffs that I wrote on terminal from begin to current.

`# history`

```
1 sudo apt-get update
2 sudo apt-get upgrade
3 ls
4 tar xvjf firefox-64.0.2.tar.bz2
5 ./configure
6 sudo apt-get install tweak
7 rm /var/lib/dpkg/lock
8 sudo bash
9 rm /var/lib/dpkg/lock
10 apt-get install tweak
11 apt-get install gnome-tweak
```

and so on..

`# !1`

this command will work as history's 1 command.

`sudo apt-get update`

`# history | grep update`

This pipe command will find and show all commands with 'update'.

`# history | tail -n 3` or `history | tail -n (number)`

This command will show last executed 3 commands.

man chmod

It is a very important command in Linux.

chmod stands for change mode, change file mode bits, change permissions of files or directories.

We have :

1. (u) owner of the file
2. (g) group who owns the file
3. (o) anyone others
4. (a) all

Two ways to represent these permissions.

1. Alphabetical way
2. Octal way

rwX stands for read, write, **execute**.

4 2 1 stands for read, write, execute.

7 stands for 4+2+1 which means read, write and execute.

6 stands for 4+2 which means read and write.

5 stands for 4+1 which means read and execute.

0 stands for no permission.

And so on.

chmod **options permissions file_name**

chmod **u=rwx,g=rw,o=r hellofile** or chmod **764 hellofile**

This command means:

Owner of the file can read, write and execute.

Group who owns the file can read and write.

Anyone others can only read.

@ To see a file or directory's permission :

ls **-l filename/folder**

ls **-l hellofile**

-rw-r--r-- 1 eesha eesha 1389 Mar 31 17:26 hellofile

where

1. first '-' represents the file type. '-' for a regular file, 'd' for a directory, 'l' for a symbolic link.

2. rw- represents user's permissions. Read, write, But no execution.

3. r - represents group's permissions. Read Only.
4. r - represents other's permissions. Read Only.
5. 1, number of hard links for the file.
6. eesha user.
7. eesha group.
8. 1389, size of the file in blocks. 1389 **byte** actually.
9. Mar 31 17:26 (last modified time)
10. hellofile, file name

```
# chmod -R 755 hellofolder
```

-R changes files and directories recursively.

```
# chmod u+s hello (-rwsr- -r- -)
```

where u+s means, user(owner)= Set-User-ID. That means anyone who attempts to access that file does so as if they are the owner of the file.

Where '+' causes the selected file mode bits to be added to the existing file mode.

```
# chmod u-s hello (-rwxr- -r- -)
```

Opposite of above.

```
# chmod a=rw hello
```

read and write by everyone.

man chown

chown command changes **ownership** of files and directories in a Linux file system.

```
# chown options ugo(who will own the file) filename
```

':' means omission.

@ user (user will own, others be unchanged)

@ user: (user own, group is omitted)

@ user:group (user,group both will own)

@ : (nothing changes)

@ Only **root** can change the owner of a file. Use **sudo** for it.

```
# sudo chown root: hello.sql
```

```
or, sudo chown root:root hello.sql
```

eesha (normal user) owned hello.sql before. But now, **root** and **root_group** (because of :) will own hello.sql file. That's all

```
# sudo chown -R root: Documents
```

-R changes files and directories recursively.

@ Group

In Linux, a user can be a member of multiple groups, but it has only one "current group". The user's current group is the user's group identity, or **GID**.

```
# id -g
```

-g to check your current group.

```
1000
```

```
# id -ng
```

-n name it.

```
eesha
```

```
# id -nG
```

-G all of the groups existing in OS.

```
eesha adm cdrom sudo dip plugdev lpadmin sambashare
```

```
# sudo chown :dip hellofile
```

```
-rwxrw-r-- 1 eesha dip 1389 Mar 31 17:26 hello
```

```
# sudo chown 1000:1000 hellofile
```

```
or sudo chown eesha:eesha hellofile
```

where 1000:1000 is **UID:GID**.

man finger

finger looks up and displays information about system **users**.

finger **options** users

finger -s eesha

-s displays the user's login name, real name, terminal name and write status, idle time, login time, office location and office phone number.

Login	Name	Tty	Idle	Login Time	Office	Office Phone
eesha	eesha	*:0		Mar 31 20:59	(:0)	

finger -l eesha

-l displays user's home directory, home phone number, login shell, mail status, and the contents of the files ".plan".

Login: eesha Name: eesha
Directory: /home/eesha Shell: /bin/bash
On since Sun Mar 31 20:59 (+06) on :0 from :0 (messages off)
No mail.
No Plan.

finger -p root

-p prevents from ".plan", ".project" .

Login: root Name: root
Directory: /root Shell: /bin/bash
Never logged in.
No mail.

man pwd

pwd means print the name of working/current directory.

pwd or pwd -P

prints current directory.

/home/eesha/Documents

avoid all symlinks

pwd -L
use PWD from environment, even if it contains symlinks

man cal

cal command displays a simple, formatted calendar in your terminal.
ncal provides the same functions of cal, but it can display the calendar vertically and some extra options.

cal or ncal
displays present date with highlight and month.

cal -3
Display last month, this month, and next month.

cal -m February
or cal -m Feb
or cal -m 2
Displays Feb. As same as ncal

cal -y 1996
or cal 1996
Entire calendar of 1996

cal -d 1996-09 (yyyy-mm)
or cal 09 1996
displays September of 1996

ncal -w 2008
-w means week numbers. Displays week number .

cal -j 2008
or ncal -J 2008
Julian Calendar.

man logout

exit/logout/bye/lo commands at the shell prompt will cause the shell to exit.

@ write utmp and wtmp entries.

@ The utmp file records who is currently using the system. The wtmp file records all logins and logouts.

exit

man shutdown

Shut down means Halt, power-off or reboot the machine.

shutdown **options** TIME (24 CLOCK)

shutdown or shutdown -P or **shutdown now**

Power-off PC immediate

shutdown -r or shutdown -r now

reboot the system immediate.

shutdown - -reboot 10:10

reboots at 10:10 AM (24 hr)