**Assignment 1**

**Aim: Study of Important Linux Commands**

**Objective:** To study the frequently used Linux commands

**File Commands**

**1) man :**

man - an interface to the on-line reference manuals

**Description:**

man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed.

**Example**:

man ls :- Display the manual page for the item (program) ls.

man cat :- Display the manual page for the item (program) cat.

man touch :- Display the manual page for the item (program) touch.

man grep: Display the manual page for the item (program) grep.

man mkdir :- Display the manual page for the item (program) mkdir.

man cd : Display the manual page for the item (program) cd.

**2) ls :**

ls - list directory contents

**Description:**

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.

-a, --all

do not ignore entries starting with .

-A, --almost-all

do not list implied . and etc..

Exit status:

0 if OK,

1 if minor problems (e.g., cannot access subdirectory),

2 if serious trouble (e.g., cannot access command-line argument).

**Examples**:

1) ls :-

**ls** with no option list files and directories in bare format where we won’t be able to view details like file types, size, modified date and time, permission and links etc.

2) ls -l

Here, **ls -l** (**-l** is character not one) shows file or directory, size, modified date and time, file or folder name and owner of file and its permission.

3) ls -a

List all files including hidden file starting with ‘**.**‘. it will lsit hidden files.

4) ls -lh

With combination of **-lh** option, shows sizes in human readable format.

5) ls -F

Using **-F** option with **ls** command, will add the **‘/’** Character at the end each directory.

6) ls -ltr

With combination of **-ltr** will shows latest modification file or directory date as last.

7) ls -i

With **-i** options list file **/** directory with inode number.

8) ls -n

To display **UID** and **GID** of files and directories. use option **-n** with ls command.

**3) cd**

**Description:**

Change the current directory to DIR. The default DIR is the value of the

HOME shell variable.

The variable CDPATH defines the search path for the directory containing

DIR. Alternative directory names in CDPATH are separated by a colon (:).

A null directory name is the same as the current directory. If DIR begins

with a slash (/), then CDPATH is not used.

If the directory is not found, and the shell option `cdable\_vars' is set,

the word is assumed to be a variable name. If that variable has a value,

its value is used for DIR.

**Options:**

-L force symbolic links to be followed: resolve symbolic

links in DIR after processing instances of `..'

-P use the physical directory structure without following

symbolic links: resolve symbolic links in DIR before

processing instances of `..'

-e if the -P option is supplied, and the current working

directory cannot be determined successfully, exit with

a non-zero status

-@ on systems that support it, present a file with extended

attributes as a directory containing the file attributes

The default is to follow symbolic links, as if `-L' were specified.

`..' is processed by removing the immediately previous pathname component

back to a slash or the beginning of DIR.

Exit Status:

Returns 0 if the directory is changed, and if $PWD is set successfully when

-P is used; non-zero otherwise.

**Examples:**

* cd /: this command is used to change directory to the root directory, The root directory in the first directory in your file system hierarchy.
* cd dir\_1/dir\_2/dir\_3: This command is used to move inside a directory from a directory
* cd ~ : this command is used to change directory to the home directory.
* Cd .. : this command is used to move to the parent directory of current directory, or the directory one level up from the current directory. “..” represents parent directory.
* cd “dir name”: This command is used to navigate to a directory with white spaces. Instead of using double quotes, single quotes can be used too.

**4) pwd:**

**Description:**

Print the name of the current working directory.

**Options:**

-L print the value of $PWD if it names the current working directory

-P print the physical directory, without any symbolic links

By default, `pwd' behaves as if `-L' were specified.

**Exit Status:**

Returns 0 unless an invalid option is given or the current directory

cannot be read.

**5) mkdir**

**Usage:**

mkdir [OPTION]... DIRECTORY...

Create the DIRECTORY(ies), if they do not already exist.

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

-m, --mode=MODE set file mode (as in chmod), not a=rwx - umask

-p, --parents no error if existing, make parent directories as needed

-v, --verbose print a message for each created directory

-Z set SELinux security context of each created directory

to the default type

--context[=CTX] like -Z, or if CTX is specified then set the SELinux

or SMACK security context to CTX

--help display this help and exit

--version output version information and exit

**6) cat**

**Usage:**

cat [OPTION]... [FILE]...

Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

-A, --show-all equivalent to -vET

-b, --number-nonblank number nonempty output lines, overrides -n

-e equivalent to -vE

-E, --show-ends display $ at end of each line

-n, --number number all output lines

-s, --squeeze-blank suppress repeated empty output lines

-t equivalent to -vT

-T, --show-tabs display TAB characters as ^I

-u (ignored)

-v, --show-nonprinting use ^ and M- notation, except for LFD and TAB

--help display this help and exit

--version output version information and exit

**Examples:**

cat f - g Output f's contents, then standard input, then g's contents.

cat Copy standard input to standard output.

**9) more**

**Usage**:

more [options] <file>...

A file perusal filter for CRT viewing.

**Options:**

-d display help instead of ringing bell

-f count logical rather than screen lines

-l suppress pause after form feed

-c do not scroll, display text and clean line ends

-p do not scroll, clean screen and display text

-s squeeze multiple blank lines into one

-u suppress underlining

-<number> the number of lines per screen

+<number> display file beginning from line number

+/<string> display file beginning from search string match

--help display this help

-V, --version display version

**10) head**

**Description:**

Print the first 10 lines of each FILE to standard output.

With more than one FILE, precede each with a header giving the file name.

With no FILE, or when FILE is -, read standard input.

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

-c, --bytes=[-]NUM print the first NUM bytes of each file;

with the leading '-', print all but the last

NUM bytes of each file

-n, --lines=[-]NUM print the first NUM lines instead of the first 10;

with the leading '-', print all but the last

NUM lines of each file

-q, --quiet, --silent never print headers giving file names

-v, --verbose always print headers giving file names

-z, --zero-terminated line delimiter is NUL, not newline

--help display this help and exit

--version output version information and exit

NUM may have a multiplier suffix:

b 512, kB 1000, K 1024, MB 1000\*1000, M 1024\*1024,

GB 1000\*1000\*1000, G 1024\*1024\*1024, and so on for T, P, E, Z, Y.

**11) tail**

**Description:**

Print the last 10 lines of each FILE to standard output.

With more than one FILE, precede each with a header giving the file name.

With no FILE, or when FILE is -, read standard input.

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

-c, --bytes=[+]NUM output the last NUM bytes; or use -c +NUM to

output starting with byte NUM of each file

-f, --follow[={name | descriptor}]

output appended data as the file grows;

an absent option argument means 'descriptor'

-F same as --follow=name --retry

-n, --lines=[+]NUM output the last NUM lines, instead of the last 10;

or use -n +NUM to output starting with line NUM

--max-unchanged-stats=N

with --follow=name, reopen a FILE which has not

changed size after N (default 5) iterations

to see if it has been unlinked or renamed

(this is the usual case of rotated log files);

with inotify, this option is rarely useful

--pid=PID with -f, terminate after process ID, PID dies

-q, --quiet, --silent never output headers giving file names

--retry keep trying to open a file if it is inaccessible

-s, --sleep-interval=N with -f, sleep for approximately N seconds

(default 1.0) between iterations;

with inotify and --pid=P, check process P at

least once every N seconds

-v, --verbose always output headers giving file names

-z, --zero-terminated line delimiter is NUL, not newline

--help display this help and exit

--version output version information and exit

NUM may have a multiplier suffix:

b 512, kB 1000, K 1024, MB 1000\*1000, M 1024\*1024,

GB 1000\*1000\*1000, G 1024\*1024\*1024, and so on for T, P, E, Z, Y.

With --follow (-f), tail defaults to following the file descriptor, which

means that even if a tailed file is renamed, tail will continue to track

its end. This default behavior is not desirable when you really want to

track the actual name of the file, not the file descriptor (e.g., log

rotation). Use --follow=name in that case. That causes tail to track the

named file in a way that accommodates renaming, removal and creation.

**12) touch**

**Description:**

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

A FILE argument that does not exist is created empty, unless -c or -h

is supplied.

A FILE argument string of - is handled specially and causes touch to

change the times of the file associated with standard output.

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

-a change only the access time

-c, --no-create do not create any files

-d, --date=STRING parse STRING and use it instead of current time

-f (ignored)

-h, --no-dereference affect each symbolic link instead of any referenced

file (useful only on systems that can change the

timestamps of a symlink)

-m change only the modification time

-r, --reference=FILE use this file's times instead of current time

-t STAMP use [[CC]YY]MMDDhhmm[.ss] instead of current time

--time=WORD change the specified time:

WORD is access, atime, or use: equivalent to -a

WORD is modify or mtime: equivalent to -m

--help display this help and exit

--version output version information and exit

Note that the -d and -t options accept different time-date formats.

**13) rm**

**Usage:**

rm [OPTION]... [FILE]...

Remove (unlink) the FILE(s).

-f, --force ignore nonexistent files and arguments, never prompt

-i prompt before every removal

-I prompt once before removing more than three files, or

when removing recursively; less intrusive than -i,

while still giving protection against most mistakes

--interactive[=WHEN] prompt according to WHEN: never, once (-I), or

always (-i); without WHEN, prompt always

--one-file-system when removing a hierarchy recursively, skip any

directory that is on a file system different from

that of the corresponding command line argument

--no-preserve-root do not treat '/' specially

--preserve-root[=all] do not remove '/' (default);

with 'all', reject any command line argument

on a separate device from its parent

-r, -R, --recursive remove directories and their contents recursively

-d, --dir remove empty directories

-v, --verbose explain what is being done

--help display this help and exit

--version output version information and exit

By default, rm does not remove directories. Use the --recursive (-r or -R)

option to remove each listed directory, too, along with all of its contents.

To remove a file whose name starts with a '-', for example '-foo',

use one of these commands:

rm -- -foo

rm ./-foo

Note that if you use rm to remove a file, it might be possible to recover

some of its contents, given sufficient expertise and/or time. For greater

assurance that the contents are truly unrecoverable, consider using shred

**14) cp**

**Usage:**

cp [OPTION]... [-T] SOURCE DEST

or: cp [OPTION]... SOURCE... DIRECTORY

or: cp [OPTION]... -t DIRECTORY SOURCE...

Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

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

-a, --archive same as -dR --preserve=all

--attributes-only don't copy the file data, just the attributes

--backup[=CONTROL] make a backup of each existing destination file

-b like --backup but does not accept an argument

--copy-contents copy contents of special files when recursive

-d same as --no-dereference --preserve=links

-f, --force if an existing destination file cannot be

opened, remove it and try again (this option

is ignored when the -n option is also used)

-i, --interactive prompt before overwrite (overrides a previous -n

option)

-H follow command-line symbolic links in SOURCE

-l, --link hard link files instead of copying

-L, --dereference always follow symbolic links in SOURCE

-n, --no-clobber do not overwrite an existing file (overrides

a previous -i option)

-P, --no-dereference never follow symbolic links in SOURCE

-p same as --preserve=mode,ownership,timestamps

--preserve[=ATTR\_LIST] preserve the specified attributes (default:

mode,ownership,timestamps), if possible

additional attributes: context, links, xattr,

all

--no-preserve=ATTR\_LIST don't preserve the specified attributes

--parents use full source file name under DIRECTORY

-R, -r, --recursive copy directories recursively

--reflink[=WHEN] control clone/CoW copies. See below

--remove-destination remove each existing destination file before

attempting to open it (contrast with --force)

--sparse=WHEN control creation of sparse files. See below

--strip-trailing-slashes remove any trailing slashes from each SOURCE

argument

-s, --symbolic-link make symbolic links instead of copying

-S, --suffix=SUFFIX override the usual backup suffix

-t, --target-directory=DIRECTORY copy all SOURCE arguments into DIRECTORY

-T, --no-target-directory treat DEST as a normal file

-u, --update copy only when the SOURCE file is newer

than the destination file or when the

destination file is missing

-v, --verbose explain what is being done

-x, --one-file-system stay on this file system

-Z set SELinux security context of destination

file to default type

--context[=CTX] like -Z, or if CTX is specified then set the

SELinux or SMACK security context to CTX

--help display this help and exit

--version output version information and exit

By default, sparse SOURCE files are detected by a crude heuristic and the

corresponding DEST file is made sparse as well. That is the behavior

selected by --sparse=auto. Specify --sparse=always to create a sparse DEST

file whenever the SOURCE file contains a long enough sequence of zero bytes.

Use --sparse=never to inhibit creation of sparse files.

When --reflink[=always] is specified, perform a lightweight copy, where the

data blocks are copied only when modified. If this is not possible the copy

fails, or if --reflink=auto is specified, fall back to a standard copy.

Use --reflink=never to ensure a standard copy is performed.

The backup suffix is '~', unless set with --suffix or SIMPLE\_BACKUP\_SUFFIX.

The version control method may be selected via the --backup option or through

the VERSION\_CONTROL environment variable. Here are the values:

none, off never make backups (even if --backup is given)

numbered, t make numbered backups

existing, nil numbered if numbered backups exist, simple otherwise

simple, never always make simple backups

As a special case, cp makes a backup of SOURCE when the force and backup

options are given and SOURCE and DEST are the same name for an existing,

regular file.

**15) mv**

**Usage:**

mv [OPTION]... [-T] SOURCE DEST

or: mv [OPTION]... SOURCE... DIRECTORY

or: mv [OPTION]... -t DIRECTORY SOURCE...

Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY.

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

--backup[=CONTROL] make a backup of each existing destination file

-b like --backup but does not accept an argument

-f, --force do not prompt before overwriting

-i, --interactive prompt before overwrite

-n, --no-clobber do not overwrite an existing file

If you specify more than one of -i, -f, -n, only the final one takes effect.

--strip-trailing-slashes remove any trailing slashes from each SOURCE

argument

-S, --suffix=SUFFIX override the usual backup suffix

-t, --target-directory=DIRECTORY move all SOURCE arguments into DIRECTORY

-T, --no-target-directory treat DEST as a normal file

-u, --update move only when the SOURCE file is newer

than the destination file or when the

destination file is missing

-v, --verbose explain what is being done

-Z, --context set SELinux security context of destination

file to default type

--help display this help and exit

--version output version information and exit

The backup suffix is '~', unless set with --suffix or SIMPLE\_BACKUP\_SUFFIX.

The version control method may be selected via the --backup option or through

the VERSION\_CONTROL environment variable. Here are the values:

none, off never make backups (even if --backup is given)

numbered, t make numbered backups

existing, nil numbered if numbered backups exist, simple otherwise

simple, never always make simple backups

**16) ln**

**Usage:**

ln [OPTION]... [-T] TARGET LINK\_NAME

or: ln [OPTION]... TARGET

or: ln [OPTION]... TARGET... DIRECTORY

or: ln [OPTION]... -t DIRECTORY TARGET...

In the 1st form, create a link to TARGET with the name LINK\_NAME.

In the 2nd form, create a link to TARGET in the current directory.

In the 3rd and 4th forms, create links to each TARGET in DIRECTORY.

Create hard links by default, symbolic links with --symbolic.

By default, each destination (name of new link) should not already exist.

When creating hard links, each TARGET must exist. Symbolic links

can hold arbitrary text; if later resolved, a relative link is

interpreted in relation to its parent directory.

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

--backup[=CONTROL] make a backup of each existing destination file

-b like --backup but does not accept an argument

-d, -F, --directory allow the superuser to attempt to hard link

directories (note: will probably fail due to

system restrictions, even for the superuser)

-f, --force remove existing destination files

-i, --interactive prompt whether to remove destinations

-L, --logical dereference TARGETs that are symbolic links

-n, --no-dereference treat LINK\_NAME as a normal file if

it is a symbolic link to a directory

-P, --physical make hard links directly to symbolic links

-r, --relative create symbolic links relative to link location

-s, --symbolic make symbolic links instead of hard links

-S, --suffix=SUFFIX override the usual backup suffix

-t, --target-directory=DIRECTORY specify the DIRECTORY in which to create

the links

-T, --no-target-directory treat LINK\_NAME as a normal file always

-v, --verbose print name of each linked file

--help display this help and exit

--version output version information and exit

The backup suffix is '~', unless set with --suffix or SIMPLE\_BACKUP\_SUFFIX.

The version control method may be selected via the --backup option or through

the VERSION\_CONTROL environment variable. Here are the values:

none, off never make backups (even if --backup is given)

numbered, t make numbered backups

existing, nil numbered if numbered backups exist, simple otherwise

simple, never always make simple backups

Using -s ignores -L and -P. Otherwise, the last option specified controls

behavior when a TARGET is a symbolic link, defaulting to -P.

**Process Management**

**1) ps (Process Status)**

**Description:**

Report a snapshot of the current processes. ps displays information about a selection of the active processes.

This version of **ps** accepts several kinds of options:

* UNIX options, which may be grouped and must be preceded by a dash.
* BSD options, which may be grouped and must not be used with a dash.
* GNU long options, which are preceded by two dashes.

Result contains four columns of information.

Where,  
PID – the unique process ID  
TTY – terminal type that the user is logged into  
TIME – amount of CPU in minutes and seconds that the process has been running  
CMD – name of the command that launched the process.

**Options:**

* ps -A or ps -E: View all the running processes
* ps -a :View Processes not associated with a terminal.
* Ps -d :View all the processes except session leaders
* ps -a -N :View all processes except those that fulfill the specified conditions (negates the selection)
* ps -T :View all processes associated with this terminal
* ps -r :View all the running processes
* ps -x :View all processes owned by you

**Process selection by list:**

Here we will discuss how to get the specific processes list with the help of ps command. These options accept a single argument in the form of a blank-separated or comma-separated list. They can be used multiple times.

**Example:**

ps -p “1 2” -p 3,4

* Select the process by the command name. This selects the processes whose executable name is given in cmdlist. There may be a chance you won’t know the process ID and with this command it is easier to search.  
  **Syntax:** ps -C command\_name
* Select by group ID or name. The group ID identifies the group of the user who created the process.

**Syntax:** ps -G group\_name

* View by group id :

**Syntax:** ps -g group\_id

* View process by process ID.

**Syntax:** ps p process\_name

* Select by parent process ID. By using this command we can view all the processes owned by parent process except the parent process.

**Syntax:** ps -p process\_id

* View all the processes belongs to any session ID.

**Syntax:** ps -s session\_id

* Select by tty. This selects the processes associated with the mentioned tty :

**Syntax:** ps t tty

ps -t tty

ps --t tty

* Select by effective user ID or name.  
  **Syntax:**  
  ps U user\_name/ID  
  ps -U user\_name/ID  
  ps -u user\_name/ID  
  ps –User user\_name/ID  
  ps –user user\_name/ID

**2) top**

**Description:**

top command is used to show the Linux processes. It provides a dynamic real-time view of the running system. Usually, this command shows the summary information of the system and the list of processes or threads which are currently managed by the Linux Kernel.  
As soon as you will run this command it will open an interactive command mode where the top half portion will contain the statistics of processes and resource usage. And Lower half contains a list of the currently running processes. Pressing *q*will simply exit the command mode.

It displays following:

* PID: Shows task’s unique process id.
* PR: Stands for priority of the task.
* SHR: Represents the amount of shared memory used by a task.
* VIRT: Total virtual memory used by the task.
* USER: User name of owner of task.
* %CPU: Represents the CPU usage.
* TIME+: CPU Time, the same as ‘TIME’, but reflecting more granularity through hundredths of a second.
* SHR: Represents the Shared Memory size (kb) used by a task.
* NI: Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priority.
* **s:**This is the process status. Processes are defined by a functioning state. It can have one of the following values:
* D - uninterruptible sleep: a sleep state where the process is waiting for something to happen. It cannot be interrupted by a signal; it is usually seen when the process is waiting for the disk.
* R - running: the process is ready to run, and will run whenever its turn to use the CPU comes
* S - sleeping: a sleep state where the process is waiting for something to happen. It can be interrupted by a signal
* T - traced or stopped: it is a state where the process is stopped, usually via SIGSTOP or SIGTSTP. It can also be stopped by a debugger (ptrace). When you see that state, it is usually because you used Ctrl-Z to put a command in the background.
* Z - zombie: is a state where the process is dead (it has finished its execution), and the only thing left is the structure describing it on the kernel. It is waiting for its parent process to retrieve its exit code, and not much more. After its parent process is finished with it, it will disappear.

**3) kill pid**

**Description:**

***kill*** command in Linux (located in /bin/kill), is a built-in command which is used to terminate processes manually. ***kill*** command sends a signal to a process which terminates the process. If the user doesn’t specify any signal which is to be sent along with kill command then default ***TERM***signal is sent that terminates the process.

**4)** **killproc**

**Usage:**

killproc [-v] [-q] [-L] [-g|-G] [-p *pid*\_*file*] [-c *root*]

[-t<sec>] [-SIG] */full/path/to/executable*

killproc [-v] [-q] [-g|-G] [-n] [-t<sec>] [-SIG]

*name*\_*of*\_*kernel*\_*thread*

killproc [-v] [-q] [-g|-G] [-n] [-t<sec>] [-SIG] *base­*

*name*\_*of*\_*executable*

killproc -l

**Description:**

killproc sends signals to all processes that use the specified executable. If no signal name is specified, the signal SIGTERM is sent. If this program is not called with the name killproc then SIGHUP is used. Note that if SIGTERM is used and does not terminate a process the signal SIGKILL is send after a few seconds . If a program has been terminate successfully and a verified pid file was found, this pid file will be removed if the terminated process didn't already do so.

**Options:**

-G Sends the signal to all session followers (chil­

dren) of the identified process.

-g Sends the signal to all members of the session

including the identified process. Note that usu­

ally the option -G should be used.

-L This option causes symlinks to be followed, as the

like-named option in [ls(1)](http://www.linux-tutorial.info/modules.php?name=ManPage&sec=1&manpage=ls). Note: for the file

name the original name of the program is used

instead of the name of the symbolic link.

-p *pid*\_*file*

(Former option -f changed due to the LSB specifica­

tion.) Use an alternate pid file instead of the

default (/var/run/<basename>.pid).

-c *root*

Change root directory to *root* for services which

have been started with this option by [startproc(8)](http://www.linux-tutorial.info/modules.php?name=ManPage&sec=8&manpage=startproc).

-n This option indicates that a kernel thread should

be signaled. In this case not the executable with

its full path name is required but the name of the

kernel thread.

-SIG Signals can be specified either by name (e.g.

-HUP, -SIGHUP) or by number (e.g. -1).

-t<sec>

The number <sec> specifies the seconds to wait

between the sent signal SIGTERM and the subsequen­

tially signal SIGKILL if the first SIGTERM does not

show any result within the first few milli seconds.

This defaults to 5 seconds.

-q This option is ignored.

-v Be more verbose.

-l This option list all available signals and some of

their synonyms by their number and signal names to

standard out. and exits.

**Examples:**

killproc -TERM /usr/sbin/sendmail

sends the signal SIGTERM to the running sendmail

no signal was specified and no program was there

for Termination because it is already terminated.

**Exit Codes:**

The exit codes have the following LSB conform conditions:

0 Success or program was not running (no signal specified)

1 Generic or unspecified error

2 Invalid or excess argument(s)

4 Insufficient privilege(s)

5 Program is not installed

7 Program was not running to receive the specified signal

In some error cases, diagnostic output is sent to standard

error, or, if standard error is not available, is being used.

**5) killall proc**

**Description:**

Instead of specifying a process by its PID, you can specify the name of the process. If more than one process runs with that name, all of them will be killed.

**Example:**

To kill the Firefox web-browser process, enter:

killall -9 firefox

**6) pkill pattern:**

**Description:**

The pkill command in Linux is basically an easier way to kill processes.

pkill command basically sends a signal to the process. By default, it's the SIGTERM signal that gets sent, but if you want, you can change the signal using the --signal command line

**Option:**

d *delimiter*

Sets the string used to delimit each process ID in the output (by default a newline). (**pgrep** only.)

-f

The *pattern* is normally only matched against the process name. When -f is set, the full command line is used.

-g *pgrp*,...

Only match processes in the process group IDs listed. Process group 0 is translated into **pgrep**'s or **pkill**'s own process group.

-G *gid*,...

Only match processes whose real group ID is listed. Either the numerical or symbolical value may be used.

-l

List the process name as well as the process ID. (**pgrep** only.)

-n

Select only the newest (most recently started) of the matching processes.

-o

Select only the oldest (least recently started) of the matching processes.

-P *ppid*,...

Only match processes whose parent process ID is listed.

-s *sid*,...

Only match processes whose process session ID is listed. Session ID 0 is translated into **pgrep**'s or **pkill**'s own session ID.

-t *term*,...

Only match processes whose controlling terminal is listed. The terminal name should be specified without the "/dev/" prefix.

-u *euid*,...

Only match processes whose effective user ID is listed. Either the numerical or symbolical value may be used.

-U *uid*,...

Only match processes whose real user ID is listed. Either the numerical or symbolical value may be used.

-v

Negates the matching.

-x

Only match processes whose name (or command line if -f is specified) **exactly** match the *pattern*.

-*signal*

Defines the signal to send to each matched process. Either the numeric or the symbolic signal name can be used. (**pkill** only.)

**Operands:**

*pattern*

Specifies an Extended Regular Expression for matching against the process names or command lines.

**Exit Status:**

* One or more processes matched the criteria.
* No processes matched.
* Syntax error in the command line.
* Fatal error: out of memory etc.

**7) bg**

**Description:**

Move jobs to the background. Place the jobs identified by each JOB\_SPEC in the background, as if they had been started with `&'. If JOB\_SPEC is not present, the shell's notion of the current job is used.

By using this command We use [jobs command](https://www.geeksforgeeks.org/process-control-commands-unixlinux/) to list all jobs, we create a process using sleep command, we get its ID as 1,We put it in background by providing its ID to bg.

**Exit Status:**

Returns success unless job control is not enabled or an error occurs.

**8)** **fg**

**Description:**

Move job to the foreground.

Place the job identified by JOB\_SPEC in the foreground, making it the

current job. If JOB\_SPEC is not present, the shell's notion of the

current job is used.

**Exit Status:**

Status of command placed in foreground, or failure if an error occurs.

**File Permissions**

**1) chmod**

**Usage:**

chmod [OPTION]... MODE[,MODE]... FILE...

or: chmod [OPTION]... OCTAL-MODE FILE...

or: chmod [OPTION]... --reference=RFILE FILE...

**Description:**

On Unix-like [operating systems](https://www.computerhope.com/os.htm), a set of [flags](https://www.computerhope.com/jargon/f/flag.htm) associated with each file determines who can access that file, and how they can access it. These flags are called file *permissions* or *modes*, as in "mode of access." The command name chmod stands for "change mode." It restricts the way a file can be accessed.

**Option:**

If no options are specified, chmod modifies the permissions of the file specified by file name to the permissions specified by permissions.

Permission: *permissions* defines the permissions for the owner of the file (the "user"), members of the group who owns the file (the "group"), and anyone else ("others"). There are two ways to represent these permissions: with symbols ([alphanumeric](https://www.computerhope.com/jargon/a/alphanum.htm) [characters](https://www.computerhope.com/jargon/c/charact.htm)), or with [octal](https://www.computerhope.com/jargon/o/octal.htm) numbers (the digits **0** through **7**).

Let's say you are the owner of a file named **myfile**, and you want to set its permissions so that:

* the **u**ser can **r**ead, **w**rite, and e**x**ecute it;
* members of your **g**roup can **r**ead and e**x**ecute it; and
* **o**thers may only **r**ead it.

This command will do the trick:

**chomd u=rwx,g=rx,o=r file\_name**

This example uses symbolic permissions notation. The letters **u**, **g**, and **o** stand for "**user**", "**group**", and "**other**". The equals sign ("**=**") means "set the permissions exactly like this," and the letters "**r**", "**w**", and "**x**" stand for "read", "write", and "execute", respectively. The commas separate the different classes of permissions, and there are no spaces in between them.

Here is the equivalent command using octal permissions notation:

chmod 754 File\_name

Here the digits **7**, **5**, and **4** each individually represent the permissions for the user, group, and others, in that order. Each digit is a combination of the numbers **4**, **2**, **1**, and **0**:

* **4** stands for "read",
* **2** stands for "write",
* **1** stands for "execute", and
* **0** stands for "no permission."

So **7** is the combination of permissions **4**+**2**+**1** (read, write, and execute), **5** is **4**+**0**+**1** (read, no write, and execute), and **4** is **4**+**0**+**0** (read, no write, and no execute).

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Permission** | **rwx** | **Binary** |
| 7 | read, write and execute | rwx | 111 |
| 6 | read and write | rw- | 110 |
| 5 | read and execute | r-x | 101 |
| 4 | read only | r-- | 100 |
| 3 | write and execute | -wx | 011 |
| 2 | write only | -w- | 010 |
| 1 | execute only | --x | 001 |
| 0 | none | --- | 000 |

Options

|  |  |
| --- | --- |
| **Tag** | **Description:** |
| -f, --silent, --quiet | suppress most error messages |
| -v, --verbose | output a diagnostic for every file processed |
| -c, --changes | like verbose but report only when a change is made |
| -c, --reference=RFile | use RFile's mode instead of MODE values |
| -R, --recursive | change files and directories recursively |
| --help | display help and exit |
| --version | output version information and exit |

**Searching**

**1)** **grep pattern**

**Description:**

Search for PATTERNS in each FILE.

**Example:**

grep -i 'hello world' menu.h main.c

PATTERNS can contain multiple patterns separated by newlines.

Pattern selection and interpretation:

-E, --extended-regexp PATTERNS are extended regular expressions

-F, --fixed-strings PATTERNS are strings

-G, --basic-regexp PATTERNS are basic regular expressions

-P, --perl-regexp PATTERNS are Perl regular expressions

-e, --regexp=PATTERNS use PATTERNS for matching

-f, --file=FILE take PATTERNS from FILE

-i, --ignore-case ignore case distinctions

-w, --word-regexp match only whole words

-x, --line-regexp match only whole lines

-z, --null-data a data line ends in 0 byte, not newline

**Miscellaneous:**

-s, --no-messages suppress error messages

-v, --invert-match select non-matching lines

-V, --version display version information and exit

--help display this help text and exit

**Output control:**

-m, --max-count=NUM stop after NUM selected lines

-b, --byte-offset print the byte offset with output lines

-n, --line-number print line number with output lines

--line-buffered flush output on every line

-H, --with-filename print file name with output lines

-h, --no-filename suppress the file name prefix on output

--label=LABEL use LABEL as the standard input file name prefix

-o, --only-matching show only nonempty parts of lines that match

-q, --quiet, --silent suppress all normal output

--binary-files=TYPE assume that binary files are TYPE;

TYPE is 'binary', 'text', or 'without-match'

-a, --text equivalent to --binary-files=text

-I equivalent to --binary-files=without-match

-d, --directories=ACTION how to handle directories;

ACTION is 'read', 'recurse', or 'skip'

-D, --devices=ACTION how to handle devices, FIFOs and sockets;

ACTION is 'read' or 'skip'

-r, --recursive like --directories=recurse

-R, --dereference-recursive likewise, but follow all symlinks

--include=GLOB search only files that match GLOB (a file pattern)

--exclude=GLOB skip files and directories matching GLOB

--exclude-from=FILE skip files matching any file pattern from FILE

--exclude-dir=GLOB skip directories that match GLOB

-L, --files-without-match print only names of FILEs with no selected lines

-l, --files-with-matches print only names of FILEs with selected lines

-c, --count print only a count of selected lines per FILE

-T, --initial-tab make tabs line up (if needed)

-Z, --null print 0 byte after FILE name

**Context control:**

-B, --before-context=NUM print NUM lines of leading context

-A, --after-context=NUM print NUM lines of trailing context

-C, --context=NUM print NUM lines of output context

-NUM same as --context=NUM

--color[=WHEN],

--colour[=WHEN] use markers to highlight the matching strings;

WHEN is 'always', 'never', or 'auto'

-U, --binary do not strip CR characters at EOL (MSDOS/Windows)

When FILE is '-', read standard input. With no FILE, read '.' if

recursive, '-' otherwise. With fewer than two FILEs, assume -h.

Exit status is 0 if any line (or file if -L) is selected, 1 otherwise;

if any error occurs and -q is not given, the exit status is 2.

**2)** **command | grep [pattern]**

**Description:**

Because the **grep** command follows the normal STDIN/STDOUT model, you can use it to work with input streams as well as files.

grep command often used with shell pipes. In this example, show the name of the hard disk devices:

**Example:**

dmesg | egrep '(s|h)d[a-z]'

**3)** **locate file**

**Description:**

**locate**command in Linux is used to find the files by name. There is two most widely used file searching utilities accessible to users are called find and **locate**. The **locate** utility works better and faster than **find**command counterpart because instead of searching the file system when a file search is initiated, it would look through a database. This database contains bits and parts of files and their corresponding paths on your system. By default, locate command does not check whether the files found in the database still exist and it never reports files created after the most recent update of the relevant database.

**Exit Status:**

This command will exit with status 0 if any specified match found. If no match founds or a fatal error encountered, then it will exit with status 1.

options

-A, --all only print entries that match all patterns

-b, --basename match only the base name of path names

-c, --count only print number of found entries

-d, --database DBPATH use DBPATH instead of default database (which is

/var/lib/mlocate/mlocate.db)

-e, --existing only print entries for currently existing files

-L, --follow follow trailing symbolic links when checking file

existence (default)

-h, --help print this help

-i, --ignore-case ignore case distinctions when matching patterns

-p, --ignore-spaces ignore punctuation and spaces when matching patterns

-t, --transliterate ignore accents using iconv transliteration when

matching patterns

-l, --limit, -n LIMIT limit output (or counting) to LIMIT entries

-m, --mmap ignored, for backward compatibility

-P, --nofollow, -H don't follow trailing symbolic links when checking file

existence

-0, --null separate entries with NUL on output

-S, --statistics don't search for entries, print statistics about each

used database

-q, --quiet report no error messages about reading databases

-r, --regexp REGEXP search for basic regexp REGEXP instead of patterns

--regex patterns are extended regexps

-s, --stdio ignored, for backward compatibility

-V, --version print version information

-w, --wholename match whole path name (default)

**Example:**

locate sample.txt

**4) find**

**Description:**

Find command is used to search and locate the list of files and directories based on conditions you specify for files that match the arguments.

Find can be used in a variety of conditions like you can find files by permissions, users, groups, file type, date, size, and other possible criteria.

**Example:**

find -name document.pdf

find -iname document.pdf

**5)** **pgrep**

**Description:**

pgrep looks through the currently running processes and lists the process IDs which matches the selection criteria to stdout. All the criteria have to match

option

**Options:**

-d, --delimiter <string> specify output delimiter

-l, --list-name list PID and process name

-a, --list-full list PID and full command line

-v, --inverse negates the matching

-w, --lightweight list all TID

-c, --count count of matching processes

-f, --full use full process name to match

-g, --pgroup <PGID,...> match listed process group IDs

-G, --group <GID,...> match real group IDs

-i, --ignore-case match case insensitively

-n, --newest select most recently started

-o, --oldest select least recently started

-P, --parent <PPID,...> match only child processes of the given parent

-s, --session <SID,...> match session IDs

-t, --terminal <tty,...> match by controlling terminal

-u, --euid <ID,...> match by effective IDs

-U, --uid <ID,...> match by real IDs

-x, --exact match exactly with the command name

-F, --pidfile <file> read PIDs from file

-L, --logpidfile fail if PID file is not locked

--ns <PID> match the processes that belong to the same namespace as <pid>

--nslist <ns,...> list which namespaces will be considered for the --ns option.

Available namespaces: ipc, mnt, net, pid, user, uts

**Example:**

pgrep [process\_is/process\_name]

**Exit Status:**

* One or more processes matched the criteria.
* No processes matched.
* Syntax error in the command line.
* Fatal error: out of memory etc.