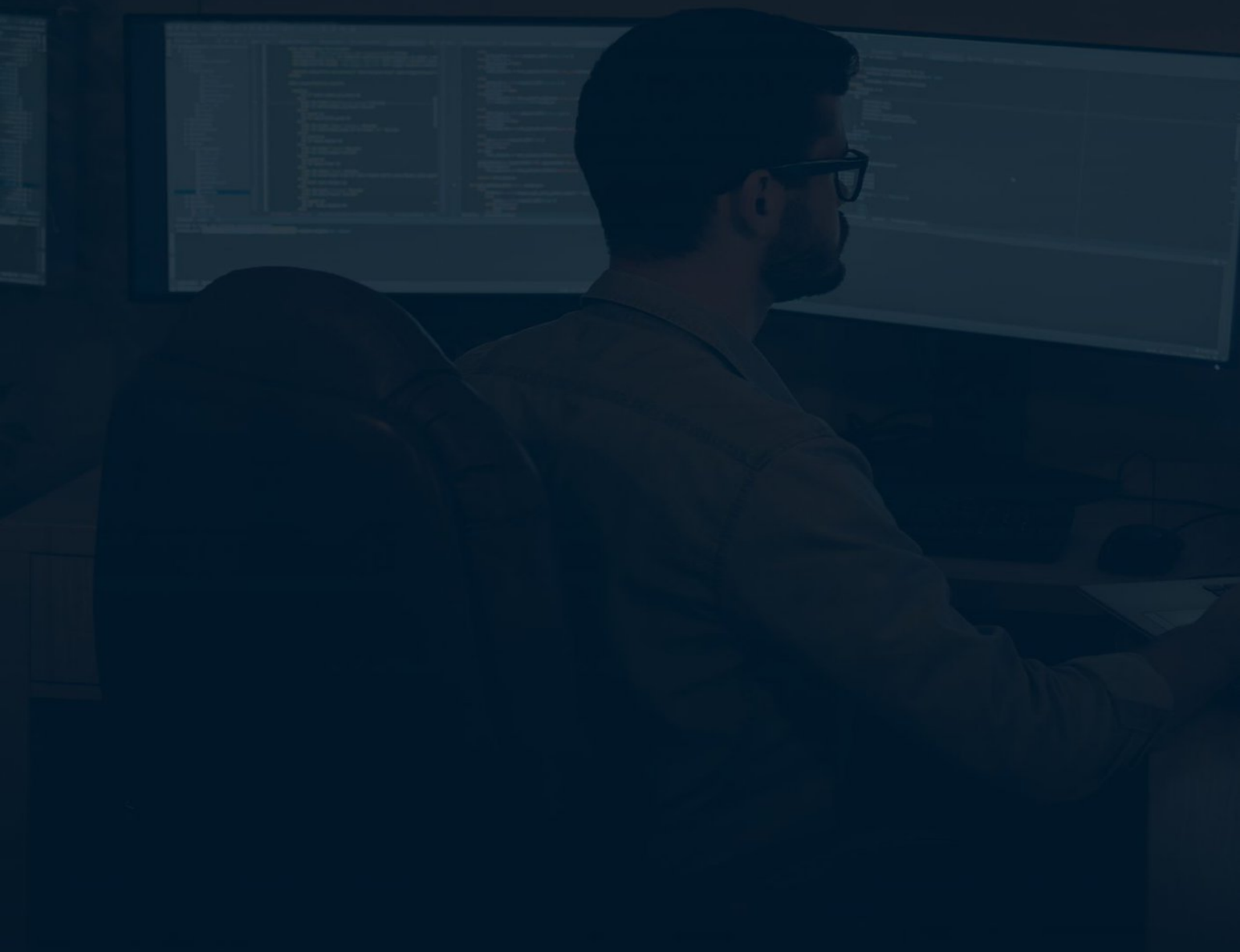
OPEN-SOURCE EBOOK

++101 Linux Commands



BOBBY ILIEV

Table of Contents

101 Linux commands Open-source eBook

This is an open-source eBook with 101 Linux commands that everyone should know. No matter if you are a DevOps/SysOps engineer, developer, or just a Linux enthusiast, you will most likely have to use the terminal at some point in your career.

Hacktoberfest

This eBook is made possible thanks to Hacktoberfest and the open source community!

About me

My name is Bobby Iliev, and I have been working as a Linux DevOps Engineer since 2014. I am an avid Linux lover and supporter of the open-source movement philosophy. I am always doing that which I cannot do in order that I may learn how to do it, and I believe in sharing knowledge.

I think it's essential always to keep professional and surround yourself with good people, work hard, and be nice to everyone. You have to perform at a consistently higher level than others. That's the mark of a true professional.

For more information, please visit my blog at https://bobbyiliev.com, follow me on Twitter @bobbyiliev_ and YouTube.

DigitalOcean

DigitalOcean is a cloud services platform delivering the simplicity developers love and businesses trust to run production applications at scale.

It provides highly available, secure, and scalable compute, storage, and networking solutions that help developers build great software faster.

Founded in 2012 with offices in New York and Cambridge, MA, DigitalOcean offers transparent and affordable pricing, an elegant user interface, and one of the largest libraries of open source resources available.

For more information, please visit https://www.digitalocean.com or follow @digitalocean on Twitter.

If you are new to DigitalOcean, you can get a free $100 credit and spin up your own servers via this referral link here:

Free $100 Credit For DigitalOcean

DevDojo

The DevDojo is a resource to learn all things web development and web design. Learn on your lunch break or wake up and enjoy a cup of coffee with us to learn something new.

Join this developer community, and we can all learn together, build together, and grow together.

Join DevDojo

For more information, please visit https://www.devdojo.com or follow @thedevdojo on Twitter.

Ebook PDF Generation Tool

This ebook was generated by Ibis developed by Mohamed Said.

Ibis is a PHP tool that helps you write eBooks in markdown.

Book Cover

The cover for this ebook was created by Suhail Kakar.

License

The `ls` command

The `ls` command lets you see the files and directories inside a specific directory *(current working directory by default)*. It normally lists the files and directories in ascending alphabetical order.

Examples:

1. To show the files inside your current working directory:

```
ls
```

2. To show the files and directory inside a specific Directory:

```
ls {Directory_Path}
```

Syntax:

```
ls [-OPTION] [DIRECTORY_PATH]
```

Interactive training

In this interactive tutorial, you will learn the different ways to use the `ls` command:

The ls command by Tony

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-l`	-	Show results in long format
`-S`	-	Sort results by file size
`-t`	-	Sort results by modification time
`-r`	`--reverse`	Show files and directories in reverse order *(descending alphabetical order)*
`-a`	`--all`	Show all files, including hidden files *(file names which begin with a period `.`)*
`-la`	-	Show long format files and directories including hidden files
`-lh`	-	list long format files and directories with readable size
`-A`	`--almost-all`	Shows all like `-a` but without showing `.`(current working directory) and `..` (parent directory)
`-d`	`--directory`	Instead of listing the files and directories inside the directory, it shows any information about the directory itself, it can be used with `-l` to show long formatted information
`-F`	`--classify`	Appends an indicator character to the end of each listed name, as an example: `/` character is appended after each directory name listed
`-h`	`--human-readable`	like `-l` but displays file size in human-readable unit not in bytes

Setting Persistent Options:

Using the alias command it's possible to set persistent options for various commands, including ls. This alias command sets color to auto, lists in long format, and show human-readable file sizes

```
alias ls="ls --color=auto -lh"
```

This alias will be active only on the current session until it ends. For this alias to be active for all new sessions, add the command to your user rc file for example for bash : ~/.bashrc

The `cd` command

The `cd` command is used to change the current working directory *(i.e., in which the current user is working)*. The "cd" stands for "**c**hange **d**irectory" and it is one of the most frequently used commands in the Linux terminal.

The `cd` command is often combined with the `ls` command (see chapter 1) when navigating through a system, however, you can also press the `TAB` key two times to list the contents of the new directory you just changed to.

Examples of uses:

1. Change the current working directory:

```
cd <specified_directory_path>
```

2. Change the current working directory to the home directory:

```
cd ~
```

OR

```
cd
```

3. Change to the previous directory:

```
cd -
```

This will also echo the absolute path of the previous directory.

4. Change the current working directory to the system's root directory:

```
cd /
```

Quick Tips

Adding a `..` as a directory will allow you to move "up" from a folder:

```
cd ..
```

This can also be done multiple times! For example, to move up three folders:

```
cd ../../../
```

Syntax:

```
cd [OPTIONS] directory
```

Additional Flags and Their Functionalities

Short flag	Long flag	Description
`-L`	-	Follow symbolic links. By default, `cd` behaves as if the `-L` option is specified.
`-P`	-	Don’t follow symbolic links.

The `cat` command

The `cat` command allows us to create single or multiple files, to view the content of a file or to concatenate files and redirect the output to the terminal or files.

The "cat" stands for 'concatenate.' and it's one of the most frequently used commands in the Linux terminal.

Examples of uses:

1. To display the content of a file in terminal:

```
cat <specified_file_name>
```

2. To display the content of multiple files in terminal:

```
cat file1 file2 ...
```

3. To create a file with the cat command:

```
cat > file_name
```

4. To display all files in current directory with the same filetype:

```
cat *.<filetype>
```

5. To display the content of all the files in current directory:

```
cat *
```

6. To put the output of a given file into another file:

```
cat old_file_name > new_file_name
```

7. Use cat command with `more` and `less` options:

```
cat filename | more
cat filename | less
```

8. Append the contents of file1 to file2:

```
cat file1 >> file2
```

9. To concatenate two files together in a new file:

```
cat file1_name file2_name merge_file_name
```

10. Some implementations of cat, with option -n, it's possible to show line numbers:

```
cat -n file1_name file2_name > new_numbered_file_name
```

Syntax:

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

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-A`	`--show-all`	equivalent to -vET
`-b`	`--number-nonblank`	number nonempty output lines, overrides -n
`-e`	-	equivalent to -vE
`-T`	-	Display tab separated lines in file opened with `cat` command.
`-E`	-	To show $ at the end of each file.
`-E`	-	Display file with line numbers.
`-n`	`--number`	number all output lines
`-s`	`--squeeze-blank`	suppress repeated empty output lines
`-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

The `tac` command

`tac` is a Linux command that allows you to view files line-by-line, beginning from the last line. (tac doesn't reverse the contents of each individual line, only the order in which the lines are presented.) It is named by analogy with `cat`.

Examples of uses:

1. To display the content of a file in terminal:

```
tac <specified_file_name>
```

2. This option attaches the separator before instead of after.

```
tac -b concat_file_name tac_example_file_name
```

3. This option will interpret the separator as a regular expression.

```
tac -r concat_file_name tac_example_file_name
```

4. This option uses STRING as the separator instead of newline.

```
tac -s concat_file_name tac_example_file_name
```

5. This option will display the help text and exit.

```
tac --help
```

6. This option will give the version information and exit.

```
tac --version
```

Syntax:

```
tac [OPTION]... [FILE]...
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-b`	`--before`	attach the separator before instead of after
`-r`	`--regex`	interpret the separator as a regular expression
`-s`	`--separator=STRING`	use STRING as the separator instead of newline
-	`--help`	display this help and exit
-	`--version`	output version information and exit

The `head` command

The `head` command prints the first ten lines of a file.

Example:

```
head filename.txt
```

Syntax:

```
head [OPTION] [FILENAME]
```

Get a specific number of lines:

Use the `-n` option with a number (should be an integer) of lines to display.

Example:

```
head -n 10 foo.txt
```

This command will display the first ten lines of the file `foo.txt`.

Syntax:

```
head -n <number> foo.txt
```

Additional Flags and their Functionalities

Short Flag	Long Flag	Description
`-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

The `tail` command

The `tail` command prints the last ten lines of a file.

Example:

```
tail filename.txt
```

Syntax:

```
tail [OPTION] [FILENAME]
```

Get a specific number of lines with `tail`:

Use the `-n` option with a number(should be an integer) of lines to display.

Example:

```
tail -n 10 foo.txt
```

This command will display the last ten lines of the file `foo.txt`.

Refresh the output on any new entry in a file

It is possible to let tail output any new line added to the file you are looking into. So, if a new line is written to the file, it will immediately be shown in your output. This can be done using the `--follow` or `-f` option. This is especially useful for monitoring log files.

Example:

```
tail -f foo.txt
```

Syntax:

```
tail -n <number> foo.txt
```

Additional Flags and their Functionalities

Short Flag	Long Flag	Description
-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 rename (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

Short Flag	Long Flag	Description
`-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

The `pwd` command

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

Example:

```
pwd
```

The output would be your current directory:

```
/home/your_user/some_directory
```

Syntax:

```
pwd [OPTION]
```

Tip: You can also check this by printing out the $PWD variable:

```
echo $PWD
```

The output would be the same as of the pwd command.

Options:

Short Flag	Long Flag	Description
-L	--logical	If the environment variable $PWD contains an absolute name of the current directory with no "." or ".." components, then output those contents, even if they contain symbolic links. Otherwise, fall back to default (-P) behavior.

Short Flag	Long Flag	Description
`-P`	`--physical`	Print a fully resolved name for the current directory, where all components of the name are actual directory names, and not symbolic links.
	`--help`	Display a help message, and exit.
	`--version`	Display version information, and exit.

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

The `touch` Command

The `touch` command modifies a file's timestamps. If the file specified doesn't exist, an empty file with that name is created.

Syntax

```
touch [OPTION]... FILE...
```

Options

Short Flag	Long Flag	Description
`-a`	-	Change only the access time.
`-c`	`--no-create`	Do not create any files.
`-d` STRING	`--date=STRING`	Parse *STRING* and use it instead of the current time.
`-f`	-	(Ignored) This option does nothing but is accepted to provide compatibility with BSD versions of the `touch` command.
`-h`	`--no-dereference`	Affect each symbolic link instead of any referenced file (useful only on systems that can change the timestamps of a symlink). This option implies `-c`, nothing is created if the file does not exist.
`-m`	-	Change only the modification time.
`-r=FILE`	`--reference=FILE`	Use this file's times instead of the current time.
`-t STAMP`	-	Use the numeric time *STAMP* instead of the current time. The format of *STAMP* is [**CC**]**YY**]**MMDDhhmm**[**.ss**].
-	`--time=WORD`	An alternate way to specify which type of time is set (e.g. *access*, *modification*, or *change*). This is equivalent to specifying `-a` or `-m`.

- WORD is `access`, `atime`, or `use`: equivalent to `-a`.
- WORD is `modify` or `mtime`: equivalent to `-m`.

An alternate way to specify what type of time to set (as with **-a** and **-m**).| |

-
|`--help`|Display a help message, and exit.| |
-
|`--version`|Display version information, and exit.|

Examples

1. If **file.txt** exists, set all of its timestamps to the current system time. If **file.txt** doesn't exist, create an empty file with that name.

```
touch file.txt
```

2. If **file.txt** exists, set its times to the current system time. If it does not exist, do nothing.

```
touch -c file.txt
```

3. Change the *access* time of **file.txt**. The *modification* time is not changed. The *change* time is set to the current system time. If **file.txt** does not exist, it is created.

```
touch -a file.txt
```

4. Change the times of file **symboliclink**. If it's a symbolic link, change the times of the symlink, ***NOT*** the times of the referenced file.

```
touch -h symboliclink
```

5. Change the *access* and *modification* times of **file-b.txt** to match the times of **file-a.txt**. The *change* time will be set to the current system time. If **file-b.txt** does not exist, it is not created. Note, **file-a.txt** must already exist in this context.

```
touch -cr file-a.txt file-b.txt
```

6. Set the *access* time and *modification* time of **file.txt** to ***February 1st*** of the current year. The *change* time is set to the current system time.

```
touch -d "1 Feb" file.txt
```

The `cal` Command

The `cal` command displays a formatted calendar in the terminal. If no options are specified, cal displays the current month, with the current day highlighted.

Syntax:

```
cal [general options] [-jy] [[month] year]
```

Options:

Option	Description
`-h`	Don't highlight today's date.
`-m month`	Specify a month to display. The month specifier is a full month name (e.g., February), a month abbreviation of at least three letters (e.g., Feb), or a number (e.g., 2). If you specify a number, followed by the letter "f" or "p", the month of the following or previous year, respectively, display. For instance, `-m 2f` displays February of next year.
`-y year`	Specify a year to display. For example, `-y 1970` displays the entire calendar of the year 1970.
`-3`	Display last month, this month, and next month.
`-1`	Display only this month. This is the default.
`-A num`	Display num months occurring after any months already specified. For example, `-3 -A 3` displays last month, this month, and four months after this one; and `-y 1970 -A 2` displays every month in 1970, and the first two months of 1971.
`-B num`	Display num months occurring before any months already specified. For example, `-3 -B 2` displays the previous three months, this month, and next month.
`-d YYYY-MM`	Operate as if the current month is number MM of year YYYY.

Examples:

1. Display the calendar for this month, with today highlighted.

```
cal
```

2. Same as the previous command, but do not highlight today.

```
cal -h
```

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

```
cal -3
```

4. Display this entire year's calendar.

```
cal -y
```

5. Display the entire year 2000 calendar.

```
cal -y 2000
```

6. Same as the previous command.

```
cal 2000
```

7. Display the calendar for December of this year.

```
cal -m [December, Dec, or 12]
```

10. Display the calendar for December 2000.

```
cal 12 2000
```

The `bc` command

The `bc` command provides the functionality of being able to perform mathematical calculations through the command line.

Examples:

1 . Arithmetic:

```
Input : $ echo "11+5" | bc
Output : 16
```

2 . Increment:

- var –++ : Post increment operator, the result of the variable is used first and then the variable is incremented.
- – ++var : Pre increment operator, the variable is increased first and then the result of the variable is stored.

```
Input: $ echo "var=3;++var" | bc
Output: 4
```

3 . Decrement:

- var – – : Post decrement operator, the result of the variable is used first and then the variable is decremented.
- – – var : Pre decrement operator, the variable is decreased first and then the result of the variable is stored.

```
Input: $ echo "var=3;--var" | bc
Output: 2
```

4 . Assignment:

- var = value : Assign the value to the variable
- var += value : similar to var = var + value
- var -= value : similar to var = var – value
- var *= value : similar to var = var * value
- var /= value : similar to var = var / value
- var ^= value : similar to var = var ^ value
- var %= value : similar to var = var % value

```
Input: $ echo "var=4;var" | bc
Output: 4
```

5 . Comparison or Relational:

- If the comparison is true, then the result is 1. Otherwise,(false), returns 0
- expr1<expr2 : Result is 1, if expr1 is strictly less than expr2.
- expr1<=expr2 : Result is 1, if expr1 is less than or equal to expr2.
- expr1>expr2 : Result is 1, if expr1 is strictly greater than expr2.
- expr1>=expr2 : Result is 1, if expr1 is greater than or equal to expr2.
- expr1==expr2 : Result is 1, if expr1 is equal to expr2.
- expr1!=expr2 : Result is 1, if expr1 is not equal to expr2.

```
Input: $ echo "6<4" | bc
Output: 0
```

```
Input: $ echo "2==2" | bc
Output: 1
```

6 . Logical or Boolean:

- expr1 && expr2 : Result is 1, if both expressions are non-zero.
- expr1 || expr2 : Result is 1, if either expression is non-zero.
- ! expr : Result is 1, if expr is 0.

```
Input: $ echo "! 1" | bc
Output: 0

Input: $ echo "10 && 5" | bc
Output: 1
```

Syntax:

```
bc [ -hlwsqv ] [long-options] [  file ... ]
```

Additional Flags and their Functionalities:

Note: This does not include an exhaustive list of options.

Short Flag	Long Flag	Description
`-i`	`--interactive`	Force interactive mode
`-l`	`--mathlib`	Use the predefined math routines
`-q`	`--quiet`	Opens the interactive mode for bc without printing the header
`-s`	`--standard`	Treat non-standard bc constructs as errors
`-w`	`--warn`	Provides a warning if non-standard bc constructs are used

Notes:

1. The capabilities of `bc` can be further appreciated if used within a script. Aside from basic arithmetic operations, `bc` supports increments/decrements, complex calculations, logical comparisons, etc.
2. Two of the flags in `bc` refer to non-standard constructs. If you evaluate `100>50 | bc` for example, you will get a strange warning. According to the POSIX page for bc, relational operators are only valid if used within an `if`, `while`, or `for` statement.

The `df` command

The `df` command in Linux/Unix is used to show the disk usage & information. `df` is an abbreviation for "disk free".

`df` displays the amount of disk space available on the file system containing each file name argument. If no file name is given, the space available on all currently mounted file systems is shown.

Syntax

```
df [OPTION]... [FILE]...
```

Options

Short Flag	Long Flag	Description
`-a`	`--all`	Include pseudo, duplicate, inaccessible file systems.
`-B`	`--block-size=SIZE`	Scale sizes by SIZE before printing them; e.g., `-BM` prints sizes in units of 1,048,576 bytes; see SIZE format below.
`-h`	`--human-readable`	Print sizes in powers of 1024 (e.g., 1023M).
`-H`	`--si`	Print sizes in powers of 1000 (e.g., 1.1G).
`-i`	`--inodes`	List inode information instead of block usage.
`-k`	-	Like `--block-size=1K`.
`-l`	`--local`	Limit listing to local file systems.
-	`--no-sync`	Do not invoke `sync` before getting usage info (default).
-	`--output[=FIELD_LIST]`	Use the output format defined by `FIELD_LIST`, or print all fields if FIELD_LIST is omitted.
`-P`	`--portability`	Use the `POSIX` output format
-	`--sync`	Invoke sync before getting usage info.

Short Flag	Long Flag	Description
-	`--total`	Elide all entries insignificant to available space, and produce a grand total.
`-t`	`--type=TYPE`	Limit listing to file systems of type `TYPE`.
`-T`	`--print-type`	Print file system type.
`-x`	`--exclude-type=TYPE`	Limit listing to file systems not of type `TYPE`.
`-v`	-	Ignored; included for compatibility reasons.
-	`--help`	Display help message and exit.
-	`--version`	Output version information and exit.

Examples:

1. Show available disk space **Action:** --- Output the available disk space and where the directory is mounted

Details: --- Outputted values are not human-readable (are in bytes)

Command:

```
df
```

2. Show available disk space in human-readable form **Action:** --- Output the available disk space and where the directory is mounted

Details: --- Outputted values ARE human-readable (are in GBs/MBs)

Command:

```
df -h
```

3. Show available disk space for the specific file system **Action:** --- Output the available disk space and where the directory is mounted

Details: --- Outputted values are only for the selected file system

Command:

```
df -hT file_system_name
```

4. Show available inodes **Action:** --- Output the available inodes for all file systems

Details: --- Outputted values are for inodes and not available space

Command:

```
df -i
```

5. Show file system type **Action:** --- Output the file system types

Details: --- Outputted values are for all file systems

Command:

```
df -T
```

6. Exclude file system type from the output **Action:** --- Output the information while excluding the chosen file system type

Details: --- Outputted values are for all file systems EXCEPT the chosen file system type

Command:

```
df -x file_system_type
```

The `help` command

The `help` command displays information about builtin commands. Display information about builtin commands.

If a `PATTERN` is specified, this command gives detailed help on all commands matching the `PATTERN`, otherwise the list of available help topics is printed.

Syntax

```
$ help [-dms] [PATTERN ...]
```

Options

Option	Description
`-d`	Output short description for each topic.
`-m`	Display usage in pseudo-manpage format.
`-s`	Output only a short usage synopsis for each topic matching the provided `PATTERN`.

Example

```
$ help ls
```

The `factor` command

The `factor` command prints the prime factors of each specified integer `NUMBER`. If none are specified on the command line, it will read them from the standard input.

Syntax

```
$ factor [NUMBER]...
```

OR:

```
$ factor OPTION
```

Options

Option	Description
`--help`	Display this a help message and exit.
`--version`	Output version information and exit.

Examples

1. Print prime factors of a prime number.

```
$ factor 50
```

2. Print prime factors of a non-prime number.

```
$ factor 75
```

The `uname` command

The `uname` command lets you print out system information and defaults to outputting the kernel name.

Syntax:

```
$ uname [OPTION]
```

Examples

1. Print out all system information.

```
$ uname -a
```

2. Print out the kernel version.

```
$ uname -v
```

Options

Short Flag	Long Flag	Description
`-a`	`--all`	Print all information, except omit processor and hardware platform if unknown.
`-s`	`--kernel-name`	Print the kernel name.
`-n`	`--nodename`	Print the network node hostname.
`-r`	`--kernel-release`	Print the kernel release.
`-v`	`--kernel-version`	Print the kernel version.
`-m`	`--machine`	Print the machine hardware name.
`-p`	`--processor`	Print the processor type (non-portable).
`-i`	`--hardware-platform`	Print the hardware platform (non-portable).
`-o`	`--operating-system`	Print the operating system.

The `mkdir` command

The `mkdir` command in Linux/Unix is used to create a directory.

Syntax

```
$ mkdir [-m=mode] [-p] [-v] [-Z=context] directory [directory
...]
```

Examples

1. Make a directory named **myfiles**.

```
$ mkdir myfiles
```

2. Create a directory named **myfiles** at the home directory:

```
$ mkdir ~/myfiles
```

3. Create the **mydir** directory, and set its file mode (`-m`) so that all users (`a`) may read (`r`), write (`w`), and execute (`x`) it.

```
$ mkdir -m a=rwx mydir
```

You can also create sub-directories of a directory. It will create the parent directory first, if it doesn't exist. If it already exists, then it move further to create the sub-directories without any error message.

For directories, this means that any user on the system may view ("read"), and create/modify/delete ("write") files in the directory. Any user may also change to ("execute") the directory, for example with the `cd` command.

4. Create the directory **/home/test/src/python**. If any of the parent directories **/home**, **/home/test**, or **/home/test/src** do not already exist, they are automatically created.

```
$ mkdir -p /home/test/src/python
```

Options

Short Flags	Long Flags	Descriptions
`-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`	`--context=CTX`	Set the SELinux security context of each created directory to CTX.
-	`--help`	Display a help message and exit.
-	`--version`	Output version information and exit.

The `gzip` command

The `gzip` command in Linux/Unix is used to compress/decompress data.

Usage

Compress a file

Action: --- Compressing a file

Details: --- Reduce the size of the file by applying compression

Command:

```
gzip file_name
```

Decompress a file

Action: --- Decompressing a file

Details: --- Restore the file's original form in terms of data and size

Command:

```
gzip -d archive_01.gz
```

Compress multiple files:

Action: --- Compress multiple files

Details: --- Compress multiple files into multiple archives

Command:

```
gzip file_name_01 file_name_02 file_name_03
```

Decompress multiple files:

Action: --- Decompress multiple files

Details: --- Decompress multiple files from multiple archives

Command:

```
gzip -d archive_01.gz archive_02.gz archive_03.gz
```

Compress a directory:

Action: --- Compress all the files in a directory

Details: --- Compress multiple files under a directory in one single archive

Command:

```
gzip -r directory_name
```

Decompress a directory:

Action: --- Decompress all the files in a directory

Details: --- Decompress multiple files under a directory from one single archive

Command:

```
gzip -dr directory_name
```

Verbose (detailed) output while compressing:

Action: --- Compress a file in a more verbose manner

Details: --- Output more information about the action of the command

Command:

```
gzip -v file_name
```

The `whatis` command

The `whatis` command is used to display one-line manual page descriptions for commands. It can be used to get a basic understanding of what a (unknown) command is used for.

Examples of uses:

1. To display what `ls` is used for:

```
whatis ls
```

2. To display the use of all commands which start with `make`, execute the following:

```
whatis -w make*
```

Syntax:

```
whatis [-OPTION] [KEYWORD]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-d`	`--debug`	Show debugging messages
`-r`	`--regex`	Interpret each keyword as a regex
`-w`	`--wildcard`	The keyword(s) contain wildcards

The `who` command

The `who` command lets you print out a list of logged-in users, the current run level of the system and the time of last system boot.

Examples

1. Print out all details of currently logged-in users

```
who -a
```

2. Print out the list of all dead processes

```
who -d -H
```

Syntax:

```
who [options] [filename]
```

Additional Flags and their Functionalities

Short Flag	Description
`-r`	prints all the current runlevel
`-d`	print all the dead processes
`-q`	print all the login names and total number of logged on users
`-h`	print the heading of the columns displayed
`-b`	print the time of last system boot

018-the-free-command.md

The `free` command

The `free` command in Linux/Unix is used to show memory (RAM/SWAP) information.

Usage

Show memory usage

Action: --- Output the memory usage - available and used, as well as swap

Details: --- Outputted values are not human-readable (are in bytes)

Command:

```
free
```

Show memory usage in human-readable form

Action: --- Output the memory usage - available and used, as well as swap

Details: --- Outputted values ARE human-readable (are in GB / MB)

Command:

```
free -h
```

The `top/htop` command

`top` is the default command-line utility that comes pre-installed on Linux distributions and Unix-like operating systems. It is used for displaying information about the system and its top CPU-consuming processes as well as RAM usage.

`htop` is interactive process-viewer and process-manager for Linux and Unix-like operating system based on ncurses. If you take top and put it on steroids, you get htop.

Comparison between top and htop:

Feature	top	htop
Type	Interactive system-monitor, process-viewer and process-manager	Interactive system-monitor, process-viewer and process-manager
Operating System	Linux distributions, macOS	Linux distributions, macOS
Installation	Built-in and is always there. Also has more adoption due to this fact.	Doesn't come preinstalled on most Linux distros. Manual installation is needed
User Interface	Basic text only	Colorful and nicer text-graphics interface
Scrolling Support	No	Yes, supports horizontal and vertical scrolling
Mouse Support	No	Yes
Process utilization	Displays processes but not in tree format	Yes, including user and kernel threads
Scrolling Support	No	Yes, supports horizontal and vertical scrolling
Mouse Support	No	Yes
Process utilization	Displays processes but not in tree format	Yes, including user and kernel threads
Network Utilization	No	No
Disk Utilization	No	No
Comments	Has a learning curve for some advanced options like searching, sending messages to processes, etc. It is good to have some knowledge of top because it is the default process viewer on many systems.	Easier to use and supports vi like searching with `/`. Sending messages to processes (kill, renice) is easier and doesn't require typing in the process number like top.

Examples:

`top`

1. To display dynamic real-time information about running processes:

```
top
```

2. Sorting processes by internal memory size (default order - process ID):

```
top -o mem
```

3. Sorting processes first by CPU, then by running time:

```
top -o cpu -O time
```

4. Display only processes owned by given user:

```
top -user {user_name}
```

`htop`

1. Display dynamic real-time information about running processes. An enhanced version of `top`.

```
htop
```

2. displaying processes owned by a specific user:

```
htop --user {user_name}
```

3. Sort processes by a specified `sort_item` (use `htop --sort help` for

available options):

```
htop --sort {sort_item}
```

Syntax:

```
top [OPTIONS]
```

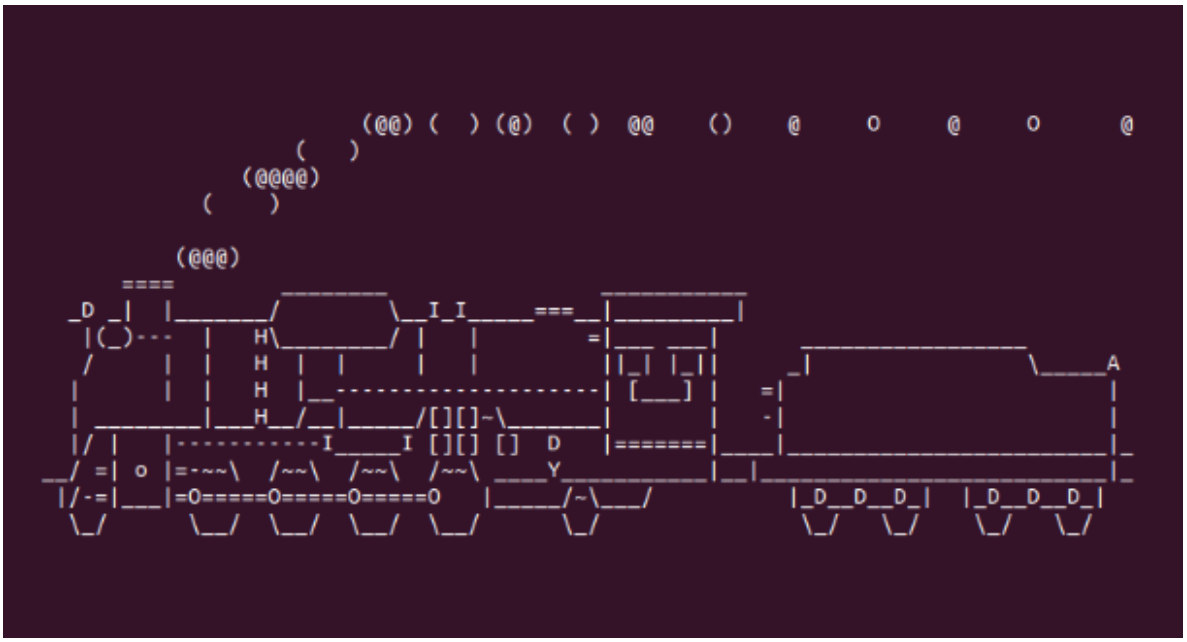
```
htop [OPTIONS]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-a`	-	Sort by memory usage.
`-b`	-	Batch mode operation. Starts top in 'Batch mode', which could be useful for sending output from top to other programs or to a file. In this mode, top will not accept input and runs until the iterations limit you've set with the '-n' command-line option or until killed.
`-h`	-	`top --user {user_name}` Only display processes owned by user.
`-U`	-user	Help.
`-u`	-	This is an alias equivalent to: -o cpu -O time.

The `sl` command

The `sl` command in Linux is a humorous program that runs a steam locomotive(sl) across your terminal.



Installation

Install the package before running.

```
sudo apt install sl
```

Syntax

```
sl
```

The `echo` command

The `echo` command lets you display the line of text/string that is passed as an argument

Examples:

1. To Show the line of text or string passed as an argument:

```
echo Hello There
```

2. To show all files/folders similar to the `ls` command:

```
echo *
```

3. To save text to a file named foo.bar:

```
echo "Hello There" > foo.bar
```

4. To append text to a file named foo.bar:

```
echo "Hello There" >> foo.bar
```

Syntax:

```
echo [option] [string]
```

It is usually used in shell scripts and batch files to output status text to the screen or a file.The `-e` used with it enables the interpretation of backslash escapes

Additional Options and their Functionalities:

Option	Description
`\b`	removes all the spaces in between the text
`\c`	suppress trailing new line with backspace interpretor ‘-e‘ to continue without emitting new line.
`\n`	creates new line from where it is used
`\t`	creates horizontal tab spaces
`\r`	carriage returns with backspace interpretor ‘-e‘ to have specified carriage return in output
`\v`	creates vertical tab spaces
`\a`	alert returns with a backspace interpretor ‘-e‘ to have sound alert
`-n`	omits echoing trailing newline .

The `finger` command

The `finger` displays information about the system users.

Examples:

1. View detail about a particular user.

```
finger abc
```

Output

```
Login: abc                              Name: (null)
Directory: /home/abc                    Shell: /bin/bash
On since Mon Nov  1 18:45 (IST) on :0 (messages off)
On since Mon Nov  1 18:46 (IST) on pts/0 from :0.0
New mail received Fri May  7 10:33 2013 (IST)
Unread since Sat Jun  7 12:59 2003 (IST)
No Plan.
```

2. View login details and Idle status about an user

```
finger -s root
```

Output

```
Login     Name                          Tty      Idle  Login Time
Office     Office Phone
root         root           *1    19d Wed 17:45
root         root           *2     3d Fri 16:53
root         root           *3        Mon 20:20
root         root           *ta    2  Tue 15:43
root         root           *tb    2  Tue 15:44
```

Syntax:

```
finger [-l] [-m] [-p] [-s] [username]
```

Additional Flags and their Functionalities:

Flag	Description
`-l`	Force long output format.
`-m`	Match arguments only on user name (not first or last name).
`-p`	Suppress printing of the .plan file in a long format printout.
`-s`	Force short output format.

Additional Information

Default Format

The default format includes the following items:

Login name
Full username
Terminal name
Write status (an * (asterisk) before the terminal name indicates that write permission is denied)
For each user on the host, the default information list also includes, if known, the following items:

Idle time (Idle time is minutes if it is a single integer, hours and minutes if a : (colon) is present, or days and hours if a “d” is present.)
Login time
Site-specific information

Longer Format

A longer format is used by the finger command whenever a list of user's names is given. (Account names as well as first and last names of users are accepted.) This format is multiline, and includes all the information described above along with the following:

User's $HOME directory
User's login shell
Contents of the .plan file in the user's $HOME directory
Contents of the .project file in the user's $HOME directory

The `groups` command

In Linux, there can be multiple users (those who use/operate the system), and groups (a collection of users). Groups make it easy to manage users with the same security and access privileges. A user can be part of different groups.

Important Points:

The `groups` command prints the names of the primary and any supplementary groups for each given username, or the current process if no names are given. If more than one name is given, the name of each user is printed before the list of that user's groups and the username is separated from the group list by a colon.

Syntax:

```
groups [username]
```

Example 1

Provided with a username

```
groups demon
```

In this example, username demon is passed with groups command and the output shows the groups in which the user demon is present, separated by a colon.

Example 2

When no username is passed then this will display the group membership for the current user:

```
groups
```

Here the current user is demon . So when we run the `groups` command without

arguments we get the groups in which demon is a user.

Example 3

Passing root with groups command:

```
$demon# groups
```

Note: Primary and supplementary groups for a process are normally inherited from its parent and are usually unchanged since login. This means that if you change the group database after logging in, groups will not reflect your changes within your existing login session. The only options are –help and –version.

The `man` command

The `man` command is used to display the manual of any command that we can run on the terminal. It provides information like: DESCRIPTION, OPTIONS, AUTHORS and more.

Examples:

1. Man page for printf:

```
man printf
```

2. Man page section 2 for intro:

```
man 2 intro
```

Syntax:

```
man [SECTION-NUM] [COMMAND NAME]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-f`	-	Return the sections of an command
`-a`	-	Display all the manual pages of an command
`-k`	-	Searches the given command with RegEx in all man pages
`-w`	-	Returns the location of a given command man page
`-I`	-	Searches the command manual case sensitive

The `passwd` command

In Linux, `passwd` command changes the password of user accounts. A normal user may only change the password for their own account, but a superuser may change the password for any account. `passwd` also changes the account or associated password validity period.

Example

```
$ passwd
```

The syntax of the `passwd` command is :

```
$ passwd [options] [LOGIN]
```

options

```
-a, --all
        This option can be used only with -S and causes show
status for all users.

-d, --delete
        Delete a user's password.

-e, --expire
        Immediately expire an account's password.

-h, --help
        Display help message and exit.

-i, --inactive
        This option is used to disable an account after the
password has been expired for a number of days.

-k, --keep-tokens
        Indicate password change should be performed only for
expired authentication tokens (passwords).

-l, --lock
        Lock the password of the named account.

-q, --quiet
        Quiet mode.

-r, --repository
        change password in repository.

-S, --status
        Display account status information.
```

The `w` command

The `w` command displays information about the users that are currently active on the machine and their processes.

Examples:

1. Running the `w` command without arguments shows a list of logged on users and their processes.

```
w
```

2. Show information for the user named *hope*.

```
w hope
```

Syntax:

```
finger [-l] [-m] [-p] [-s] [username]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-h`	`--no-header`	Don't print the header.
`-u`	`--no-current`	Ignores the username while figuring out the current process and cpu times. *(To see an example of this, switch to the root user with `su` and then run both `w` and `w -u`.)*
`-s`	`--short`	Display abbreviated output *(don't print the login time, JCPU or PCPU times).*

Short Flag	Long Flag	Description
`-f`	`--from`	Toggle printing the from *(remote hostname)* field. The default as released is for the from field to not be printed, although your system administrator or distribution maintainer may have compiled a version where the from field is shown by default.
`--help`	-	Display a help message, and exit.
`-V`	`--version`	Display version information, and exit.
`-o`	`--old-style`	Old style output *(prints blank space for idle times less than one minute)*.
`user`	-	Show information about the specified the user only.

Additional Information

The header of the output shows (in this order): the current time, how long the system has been running, how many users are currently logged on, and the system load averages for the past 1, 5, and 15 minutes.

The following entries are displayed for each user:

- login name the tty
- name the remote
- host they are
- logged in from the amount of time they are logged in their
- idle time JCPU
- PCPU
- command line of their current process

The JCPU time is the time used by all processes attached to the tty. It does not include past background jobs, but does include currently running background jobs.

The PCPU time is the time used by the current process, named in the "what" field.

The `whoami` command

The `whoami` command displays the username of the current effective user. In other words it just prints the username of the currently logged-in user when executed.

To display your effective user id just type `whoami` in your terminal:

```
manish@godsmack:~$ whoami
# Output:
manish
```

Syntax:

```
whoami [-OPTION]
```

There are only two options which can be passed to it :

`--help`: Used to display the help and exit

Example:

```
whoami --help
```

Output:

```
Usage: whoami [OPTION]...
Print the user name associated with the current effective user
ID.
Same as id -un.

      --help     display this help and exit
      --version  output version information and exit
```

`--version`: Output version information and exit

Example:

```
whoami --version
```

Output:

```
whoami (GNU coreutils) 8.32
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<https://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 Richard Mlynarik.
```

The `history` command

If you type `history` you will get a list of the last 500 commands used. This gives you the possibility to copy and paste commands that you executed in the past.

This is powerful in combination with grep. So you can search for a command in your command history.

Examples:

1. If you want to search in your history for artisan commands you ran in the past.

```
history | grep artisan
```

2. If you only want to show the last 10 commands you can.

```
history 10
```

The `login` Command

The `login` command initiates a user session.

Syntax

```
$ login [-p] [-h host] [-H] [-f username|username]
```

Flags and their functionalities

Short Flag	Description
`-f`	Used to skip a login authentication. This option is usually used by the getty(8) autologin feature.
`-h`	Used by other servers (such as telnetd(8) to pass the name of the remote host to login so that it can be placed in utmp and wtmp. Only the superuser is allowed use this option.
`-p`	Used by getty(8) to tell login to preserve the environment.
`-H`	Used by other servers (for example, telnetd(8)) to tell login that printing the hostname should be suppressed in the login: prompt.
`--help`	Display help text and exit.
`-v`	Display version information and exit.

Examples

To log in to the system as user abhishek, enter the following at the login prompt:

```
$ login: abhishek
```

If a password is defined, the password prompt appears. Enter your password at this prompt.

`lscpu` command

`lscpu` in Linux/Unix is used to display CPU Architecture info. `lscpu` gathers CPU architecture information from `sysfs` and `/proc/cpuinfo` files.

For example :

```
 manish@godsmack:~$ lscpu
   Architecture:        x86_64
   CPU op-mode(s):      32-bit, 64-bit
   Byte Order:          Little Endian
   CPU(s):              4
   On-line CPU(s) list: 0-3
   Thread(s) per core:  2
   Core(s) per socket:  2
   Socket(s):           1
   NUMA node(s):        1
   Vendor ID:           GenuineIntel
   CPU family:          6
   Model:               142
   Model name:          Intel(R) Core(TM) i5-7200U CPU @
2.50GHz
   Stepping:            9
   CPU MHz:             700.024
   CPU max MHz:         3100.0000
   CPU min MHz:         400.0000
   BogoMIPS:            5399.81
   Virtualization:      VT-x
   L1d cache:           32K
   L1i cache:           32K
   L2 cache:            256K
   L3 cache:            3072K
   NUMA node0 CPU(s):   0-3
```

Options

`-a, --all` Include lines for online and offline CPUs in the output (default for -e). This option may only specified together with option -e or -p. For example: `lsof -a`

`-b, --online` Limit the output to online CPUs (default for -p). This option may only be specified together with option -e or -p. For example: `lscpu -b`

`-c, --offline` Limit the output to offline CPUs. This option may only be specified together with option -e or -p.

`-e, --extended [=list]` Display the CPU information in human readable format. For example: `lsof -e`

For more info: use `man lscpu` or `lscpu --help`

The `cp` command

The `cp` is a command-line utility for copying files and directory. `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. The cp command requires at least two filenames in its arguments.

Examples:

1. To copy the contents of the source file to the destination file.

```
cp sourceFile destFile
```

If the destination file doesn't exist then the file is created and the content is copied to it. If it exists then the file is overwritten.

2. To copy a file to another directory specify the absolute or the relative path to the destination directory.

```
cp sourceFile /folderName/destFile
```

3. To copy a directory, including all its files and subdirectories

```
cp -R folderName1 folderName2
```

The command above creates the destination directory and recursively copies all files and subdirectories from the source to the destination directory.

If the destination directory already exists, the source directory itself and its content are copied inside the destination directory.

4. To copy only the files and subdirectories but not the source directory

```
cp -RT folderName1 folderName2
```

Syntax:

The general syntax for the cp command is as follows:

```
cp [OPTION] SOURCE DESTINATION
cp [OPTION] SOURCE DIRECTORY
cp [OPTION] SOURCE-1 SOURCE-2 SOURCE-3 SOURCE-n DIRECTORY
```

The first and second syntax is used to copy Source file to Destination file or Directory. The third syntax is used to copy multiple Sources(files) to Directory.

Some useful options

1. `-i` (interactive) `i` stands for Interactive copying. With this option system first warns the user before overwriting the destination file. cp prompts for a response, if you press y then it overwrites the file and with any other option leave it uncopied.

```
$ cp -i file1.txt fileName2.txt
cp: overwrite 'file2.txt'? y
```

2. `-b`(backup) -b(backup): With this option cp command creates the backup of the destination file in the same folder with the different name and in different format.

```
$ ls
a.txt b.txt

$ cp -b a.txt b.txt

$ ls
a.txt b.txt b.txt~
```

3. `-f`(force) If the system is unable to open destination file for writing operation because the user doesn't have writing permission for this file then by using -f

option with cp command, destination file is deleted first and then copying of content is done from source to destination file.

```
$ ls -l b.txt
-r-xr-xr-x+ 1 User User 3 Nov 24 08:45 b.txt
```

User, group and others doesn't have writing permission.

Without `-f` option, command not executed

```
$ cp a.txt b.txt
cp: cannot create regular file 'b.txt': Permission denied
```

With -f option, command executed successfully

```
$ cp -f a.txt b.txt
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-i`	--interactive	prompt before overwrite
`-f`	--force	If an existing destination file cannot be opened, remove it and try again
`-b`	-	Creates the backup of the destination file in the same folder with the different name and in different format.
`-r or -R`	`--recursive`	**cp** command shows its recursive behavior by copying the entire directory structure recursively.
`-n`	`--no-clobber`	do not overwrite an existing file (overrides a previous -i option)
`-p`	-	preserve the specified attributes (default: mode,ownership,timestamps), if possible additional attributes: context, links, xattr, all

The `mv` command

The `mv` command lets you **move one or more files or directories** from one place to another in a file system like UNIX. It can be used for two distinct functions:

- To rename a file or folder.
- To move a group of files to a different directory.

Note: *No additional space is consumed on a disk during renaming, and the mv command doesn't provide a prompt for confirmation*

Syntax:

```
mv [options] source (file or directory)  destination
```

Examples:

1. To rename a file called old_name.txt:

```
mv old_name.txt new_name.txt
```

2. To move a file called *essay.txt* from the current directory to a directory called *assignments* and rename it *essay1.txt*:

```
mv essay.txt assignments/essay1.txt
```

3. To move a file called *essay.txt* from the current directory to a directory called *assignments* without renaming it

```
mv essay.txt assignments
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-f`	`--force`	Force move by overwriting destination file without prompt
`-i`	`--interactive`	Interactive prompt before overwrite
`-u`	`--update`	Move only when the source file is newer than the destination file or when the destination file is missing
`-n`	`--no-clobber`	Do not overwrite an existing file
`-v`	`--verbose`	Print source and destination files
`-b`	`--backup`	Create a Backup of Existing Destination File

The `ps` command

The `ps` command is used to identify programs and processes that are running on the system and the resources they are using. Its frequently pipelined with other commands like `grep` to search for a program/process or `less` so that the user can analyze the output one page at a time.

Let's say you have a program like openshot which is notorious for hogging system resources when exporting a video, and you want to close it, but the GUI has become unresponsive.

Example

1. You want to find the PID of openshot and kill it.

```
ps aux | grep openshot
kill - <openshot PID>
```

2. To Show all the running processes:

```
ps -A
```

Syntax

`ps [options]`

When run without any options, it's useless and will print: `CMD` - the executable processes/(program) running, their `PID` - process ID, `TTY` - terminal type and `Time` - How long the process has utilized the CPU or thread.

Common Option

If you are going to remember only one thing from this page let it be these three letter

`aux`: `a` - which displays all processes running, including those being run by other users. `u` - which shows the effective user of a process, i.e. the person whose file access permissions are used by the process. `x` - which shows processes that do not have a `TTY` associated with them.

Additional Options:

Option	Description
`a`	Shows list all processes with a terminal (tty)
`-A`	Lists all processes. Identical to `-e`
`-a`	Shows all processes except both session leaders and processes not associated with a terminal
`-d`	Select all processes except session leaders
`--deselect`	Shows all processes except those that fulfill the specified conditions. Identical to `-N`
`-e`	Lists all processes. Identical to `-A`
`-N`	Shows all processes except those that fulfill the specified conditions. Identical to `-deselect`
`T`	Select all processes associated with this terminal. Identical to the `-t` option without any argument
`r`	Restrict the selection to only running processes
`--help simple`	Shows all the basic options
`--help all`	Shows every available options

Another useful command which give a realtime snapshot of the processes and the resources they are using about every ten seconds is `top`.

The `kill` command

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

Signals can be specified in three ways:

- **By number (e.g. -5)**
- **With SIG prefix (e.g. -SIGkill)**
- **Without SIG prefix (e.g. -kill)**

Syntax

```
kill [OPTIONS] [PID]...
```

Examples:

1. To display all the available signals you can use below command option:

```
kill -l
```

2. To show how to use a *PID* with the *kill* command.

```
$kill pid
```

3. To show how to send signal to processes.

```
kill {-signal | -s signal} pid
```

4. Specify Signal:

- using numbers as signals

```
kill -9 pid
```

- using SIG prefix in signals

```
kill -SIGHUP pid
```

- without SIG prefix in signals

```
kill -HUP pid
```

Arguments:

The list of processes to be signaled can be a mixture of names and PIDs.

```
   pid    Each pid can be expressed in one of the following
ways:

          n      where n is larger than 0.  The process with
PID n is signaled.

          0      All processes in the current process group
are signaled.

          -1     All processes with a PID larger than 1 are
signaled.

          -n     where n is larger than 1.  All processes in
process group  n  are  signaled.
                 When  an  argument  of  the  form '-n' is
given, and it is meant to denote a
                 process group, either a signal must be
specified first, or the argument must
                 be  preceded  by  a '--' option, otherwise it
will be taken as the signal to
                 send.

   name   All processes invoked using this name will be
signaled.
```

Options:

```
   -s, --signal signal
          The signal to send.  It may be given as a name or a
number.

   -l, --list [number]
          Print a list of signal names, or convert the given
signal number to  a  name.   The
          signals can be found in /usr/include/linux/signal.h.

   -L, --table
          Similar to -l, but it will print signal names and
their corresponding numbers.

   -a, --all
          Do  not  restrict the command-name-to-PID conversion
to processes with the same UID
          as the present process.

   -p, --pid
          Only print the process ID (PID) of the named
processes, do not send any signals.

   --verbose
          Print PID(s) that will be signaled with kill along
with the signal.
```

The `killall` command

`killall` sends a signal to **all** processes running any of the specified commands. If no signal name is specified, `SIGTERM` is sent. In general, `killall` command kills all processes by knowing the name of the process.

Signals can be specified either by name (e.g. `-HUP` or `-SIGHUP`) or by number (e.g. `-1`) or by option `-s`.

If the command name is not a regular expression (option `-r`) and contains a slash (`/`), processes executing that particular file will be selected for killing, independent of their name.

`killall` returns a zero return code if at least one process has been killed for each listed command, or no commands were listed and at least one process matched the `-u` and `-Z` search criteria. `killall` returns non-zero otherwise.

A `killall` process never kills itself (but may kill other `killall` processes).

Examples:

1. Kill all processes matching the name `conky` with `SIGTERM`:

```
killall conky
# OR
killall -SIGTERM conky
# OR
kilall -15 conky
```

I was able to kill Wine (which are Windows exe files running on Linux) applications this way too.

```
killall TQ.exe
```

2. List all the supported signals:

```
$ killall -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
POLL PWR SYS
```

As for the numbers.

```
$ for s in $(killall -l); do echo -n "$s " && kill -l $s; done
HUP 1
INT 2
QUIT 3
ILL 4
TRAP 5
ABRT 6
BUS 7
FPE 8
KILL 9
USR1 10
SEGV 11
USR2 12
PIPE 13
ALRM 14
TERM 15
STKFLT 16
CHLD 17
CONT 18
STOP 19
TSTP 20
TTIN 21
TTOU 22
URG 23
XCPU 24
XFSZ 25
VTALRM 26
PROF 27
WINCH 28
POLL 29
PWR 30
SYS 31
```

3. Ask before killing, to prevent unwanted kills:

```
$ killall -i conky
Kill conky(1685) ? (y/N)
```

4. Kill all processes and wait until the processes die.

```
killall -w conky
```

5. Kill based on time:

```
# Kill all firefox younger than 2 minutes
killall -y 2m  firefox

# Kill all firefox older than 2 hours
killall -o 2h  firefox
```

Syntax:

```
killall [OPTION]... [--] NAME...
killall -l, --list
killall -V, --version
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-e`	`--exact`	require an exact match for very long names
`-I`	`--ignore-case`	case insensitive process name match
`-g`	`--process-group`	kill process group instead of process
`-y`	`--younger-than`	kill processes younger than TIME
`-o`	`--older-than`	kill processes older than TIME
`-i`	`--interactive`	ask for confirmation before killing
`-l`	`--list`	list all known signal names
`-q`	`--quiet`	don't print complaints
`-r`	`--regexp`	interpret NAME as an extended regular expression

Short Flag	Long Flag	Description
`-s`	`--signal SIGNAL`	send this signal instead of SIGTERM
`-u`	`--user USER`	kill only process(es) running as USER
`-v`	`--verbose`	report if the signal was successfully sent
`-w`	`--wait`	wait for processes to die
`-n`	`--ns PID`	match processes that belong to the same namespaces as PID
`-Z`	`--context`	REGEXP kill only process(es) having context (must precede other arguments)

Related commands

kill, `pidof`

The `env` command

The `env` command in Linux/Unix is used to either print a list of the current environment variables or to run a program in a custom environment without changing the current one.

Syntax

```
env [OPTION]... [-] [NAME=VALUE]... [COMMAND [ARG]...]
```

Usage

1. Print out the set of current environment variables
   ```
   env
   ```

2. Run a command with an empty environment
   ```
    env -i command_name
   ```

3. Remove variable from the environment
   ```
   env -u variable_name
   ```

4. End each output with NULL
   ```
   env -0
   ```

Full List of Options

Short Flag	Long Flag	Description
`-i`	`--ignore-environment`	Start with an empty environment
`-0`	`--null`	End each output line with NUL, not newline
`-u`	`--unset=NAME`	Remove variable from the environment
`-C`	`--chdir=DIR`	Change working directory to DIR
`-S`	`--split-string=S`	Process and split S into separate arguments. It's used to pass multiple arguments on shebang lines
`-v`	`--debug`	Print verbose information for each processing step
-	`--help`	Print a help message
-	`--version`	Print the version information

The `printenv` command

The `printenv` prints the values of the specified environment *VARIABLE(s)*. If no *VARIABLE* is specified, print name and value pairs for them all.

Examples:

1. Display the values of all environment variables.

```
printenv
```

2. Display the location of the current user's home directory.

```
printenv HOME
```

3. To use the `--null` command line option as the terminating character between output entries.

```
printenv --null SHELL HOME
```

NOTE: By default, the `printenv` command uses newline as the terminating character between output entries.

Syntax:

```
printenv [OPTION]... PATTERN...
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-0`	`--null`	End each output line with **0** byte rather than newline.
`--help`	-	Display a help message, and exit.

The `hostname` command

`hostname` is used to display the system's DNS name, and to display or set its hostname or NIS domain name.

Syntax:

```
hostname [-a|--alias] [-d|--domain] [-f|--fqdn|--long] [-A|--
all-fqdns] [-i|--ip-address] [-I|--all-ip-addresses] [-s|--
short] [-y|--yp|--nis]
```

Examples:

1. `hostname -a, hostname --alias` Display the alias name of the host (if used). This option is deprecated and should not be used anymore.

2. `hostname -s, hostname --short` Display the short host name. This is the host name cut at the first dot.

3. `hostname -V, hostname --version` Print version information on standard output and exit successfully.

Help Command

Run below command to view the complete guide to `hostname` command.

```
man hostname
```

The `nano` command

The `nano` command lets you create/edit text files.

Installation:

Nano text editor is pre-installed on macOS and most Linux distros. It's an alternative to `vi` and `vim`. To check if it is installed on your system type:

```
nano --version
```

If you don't have `nano` installed you can do it by using the package manager:

Ubuntu or Debian:

```
sudo apt install nano
```

Examples:

1. Open an existing file, type `nano` followed by the path to the file:

```
nano /path/to/filename
```

2. Create a new file, type `nano` followed by the filename:

```
nano filename
```

3. Open a file with the cursor on a specific line and character use the following syntax:

```
nano +line_number,character_number filename
```

Overview of some Shortcuts and their Functionalities:

Shortcut	Description
`Ctrl + S`	Save current file
`Ctrl + O`	Offer to write file ("Save as")
`Ctrl + X`	Close buffer, exit from nano
`Ctrl + K`	Cut current line into cutbuffer
`Ctrl + U`	Paste contents of cutbuffer
`Alt + 6`	Copy current line into cutbuffer
`Alt + U`	Undo last action
`Alt + E`	Redo last undone action

The `rm` command

`rm` which stands for "remove" is a command used to remove *(delete)* specific files. It can also be used to remove directories by using the appropriate flag.

Example:

```
rm filename.txt
```

Syntax

```
rm [OPTION] [FILE|DIRECTORY]
```

Flags and their Functionalities:

Short Flag	Long Flag	Description
`-f`	`--force`	Ignore nonexistance of files or directories, never prompt
`-i`	-	Prompt before every removal
`-I`	-	Prompt once before removal of more than 3 files, or when removing recursively
`-d`	`--dir`	remove empty directories
`-v`	`--verbose`	explain what is being done
`-r` or `-R`	`--recursive`	remove directories and their contents recursively
-	`--help`	Display help then exit
-	`--version`	First, Print version Information, Then exit
-	`--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

Short Flag	Long Flag	Description
-	`--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 argument0

IMPORTANT NOTICE:

1. `rm` doesn't remove directories by default, so use `-r`, `-R`, `--recursive` options to remove each listed directory, along with all of its contents.
2. To remove a file whose name starts with `-` such as `-foo`, use one of the following commands:
    - `rm -- -foo`
    - `rm ./-foo`
3. To ensure that files/directories being deleted are truly unrecoverable, consider using the `shred` command.

The `ifconfig` command

`ifconfig` is used to configure the kernel-resident network interfaces. It is used at boot time to set up interfaces as necessary. After that, it is usually only needed when debugging or when system tuning is needed.

If no arguments are given, `ifconfig` displays the status of the currently active interfaces. If a single interface argument is given, it displays the status of the given interface only; if a single -a argument is given, it displays the status of all interfaces, even those that are down. Otherwise, it configures an interface.

Syntax:

```
ifconfig [-v] [-a] [-s] [interface]
ifconfig [-v] interface [aftype] options
```

Examples:

1. To display the currently active interfaces:

```
ifconfig
```

2. To show all interfaces which are currently active, even if down:

```
ifconfig -a
```

3. To show all the error conditions:

```
ifconfig -v
```

4. To show a short list:

```
ifconfig -s
```

5. To display details of the specific network interface (say `eth0`):

```
ifconfig eth0
```

6. To activate the driver for a interface (say `eth0`):

```
ifconfig eth0 up
```

7. To deactivate the driver for a interface (say `eth0`):

```
ifconfig eth0 down
```

8. To assign a specific IP address to a network interface (say `eth0`):

```
ifconfig eth0 10.10.1.23
```

9. To change MAC(Media Access Control) address of a network interface (say `eth0`):

```
ifconfig eth0 hw ether AA:BB:CC:DD:EE:FF
```

10. To define a netmask for a network interface (say `eth0`):

```
ifconfig eth0 netmask 255.255.255.224
```

11. To enable promiscous mode on a network interface (say `eth0`):

```
ifconfig eth0 promisc
```

In normal mode, when a packet is received by a network card, it verifies that it belongs to itself. If not, it drops the packet normally. However, in the promiscuous mode, it accepts all the packets that flow through the network card.

12. To disable promiscous mode on a network interface (say `eth0`):

```
ifconfig eth0 -promisc
```

13. To set the maximum transmission unit to a network interface (say `eth0`):

```
ifconfig eth0 mtu 1000
```

The MTU allows you to set the limit size of packets that are transmitted on an interface. The MTU is able to handle a maximum number of octets to an interface in one single transaction.

14. To add additional IP addresses to a network interface, you can configure a network alias to the network interface:

```
ifconfig eth0:0 10.10.1.24
```

Please note that the alias network address is in the same subnet mask of the network interface. For example, if your eth0 network ip address is `10.10.1.23`, then the alias ip address can be `10.10.1.24`. Example of an invalid IP address is `10.10.2.24` since the interface subnet mask is `255.255.255.224`

15. To remove a network alias:

```
ifconfig eth0:0 down
```

Remember that for every scope (i.e. same net with address/netmask combination) all aiases are deleted, if you delete the first alias.

Help Command

Run below command to view the complete guide to `ifconfig` command.

```
man ifconfig
```

The `ip` command

The `ip` command is present in the net-tools which is used for performing several network administration tasks. IP stands for Internet Protocol. This command is used to show or manipulate routing, devices, and tunnels. It can perform tasks like configuring and modifying the default and static routing, setting up tunnel over IP, listing IP addresses and property information, modifying the status of the interface, assigning, deleting and setting up IP addresses and routes.

Examples:

1. To assign an IP Address to a specific interface (eth1) :

```
ip addr add 192.168.50.5 dev eth1
```

2. To show detailed information about network interfaces like IP Address, MAC Address information etc. :

```
ip addr show
```

Syntax:

```
ip [ OPTIONS ] OBJECT { COMMAND | help }
```

Additional Flags and their Functionalities:

Flag	Description
`-a`	Display and modify IP Addresses
`-l`	Display and modify network interfaces
`-r`	Display and alter the routing table

Flag	Description
`-n`	Display and manipulate neighbor objects (ARP table)
`-ru`	Rule in routing policy database.
`-s`	Output more information. If the option appears twice or more, the amount of information increases
`-f`	Specifies the protocol family to use
`-r`	Use the system's name resolver to print DNS names instead of host addresses
`-c`	To configure color output

The `clear` command

In linux, the `clear` command is used to clear terminal screen.

Example

```
$ clear
```

Before:

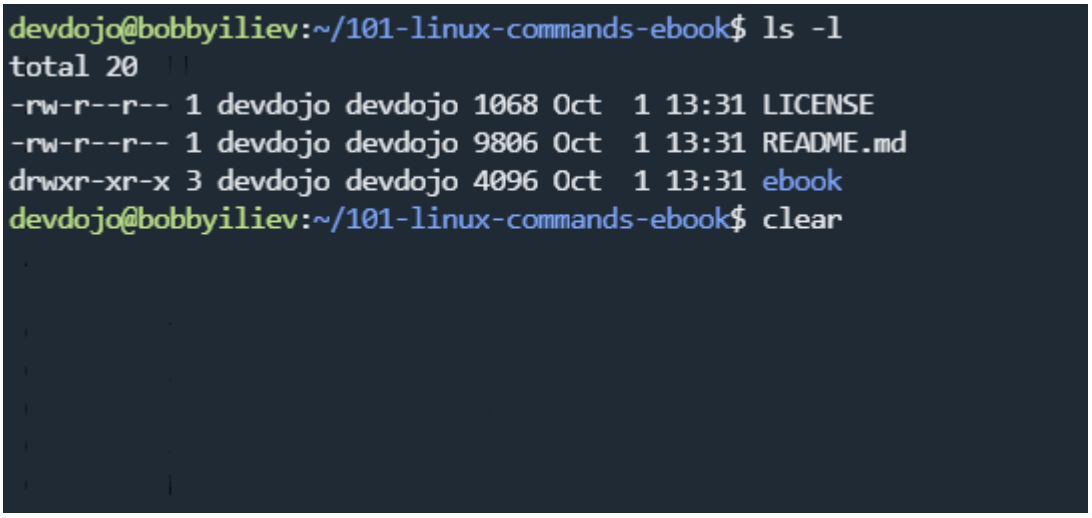
```
$ echo Hello World
Hello World

$ clear
```

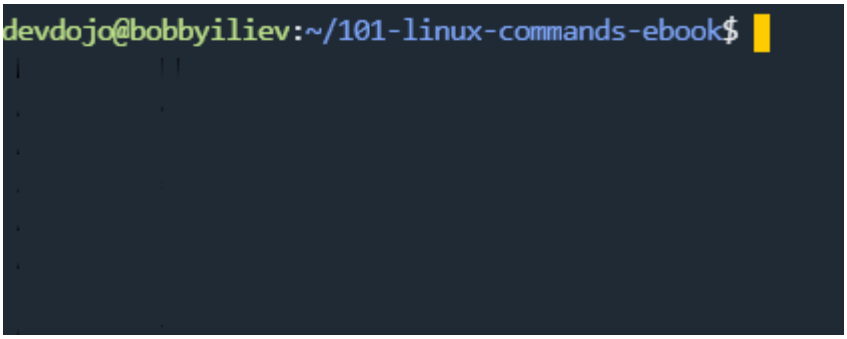
After executing clear command:

```
$
```

Screenshot:



After running the command your terminal screen will be clear:



The `su` command

In linux, `su` allows you to run commands with a substitute user and group ID.

When called without arguments, `su` defaults to running an interactive shell as root.

Example :

```
$ su
```

In case that you wanted to switch to a user called `devdojo`, you could do that by running the following command:

```
$ su devdojo
```

The syntax of the `su` command is :

```
$ su [options] [-] [<user>[<argument>...]]
```

Options :

```
-m, -p          --> do not reset environment variables
-w              --> do not reset specified variables
-g              --> specify the primary group
-G              --> specify a supplemental group
-l              --> make the shell a login shell
-f              --> pass -f to the shell (for csh or tcsh)
-s              --> run <shell> if /etc/shell allows it
-p              --> create a new pseudo terminal
-h              --> display this help
-v              --> display version
```

The `wget` command

The `wget` command is used for downloading files from the Internet. It supports downloading files using HTTP, HTTPS and FTP protocols. It allows you to download several files at once, download in the background, resume downloads, limit the bandwidth, mirror a website, and much more.

Syntax

The `wget` syntax requires you to define the downloading options and the URL the to be downloaded file is coming from.

```
$ wget [options] [URL]
```

Examples

In this example we will download the Ubuntu 20.04 desktop iso file from different sources. Go over to your terminal or open a new one and type in the below `wget`. This will stat the download. The download may take a few minutes to complete.

1. Starting a regular download

```
wget
https://releases.ubuntu.com/20.04/ubuntu-20.04.3-desktop-amd64
.iso
```

2. You can resume a download using the `-c` option

```
wget -c
https://mirrors.piconets.webwerks.in/ubuntu-mirror/ubuntu-rele
ases/20.04.3/ubuntu-20.04.3-desktop-amd64.iso
```

3. To download in the background, use the `-b` option

```
wget -b
https://mirrors.piconets.webwerks.in/ubuntu-mirror/ubuntu-rele
ases/20.04.3/ubuntu-20.04.3-desktop-amd64.iso
```

More options

On top of downloading, `wget` provides many more features, such as downloading multiple files, dowloading in the background, limiting download bandwith and resuming stopped downloads. View all `wget` options in its man page.

```
man wget
```

Additional Flags and their Functionalities

Short Flag	Description
`-v`	prints version of the wget available on your system
`-h`	print help message displaying all the possible options
`-b`	This option is used to send a process to the background as soon as it starts.
`-t`	This option is used to set number of retries to a specified number of times
`-c`	This option is used to resume a partially downloaded file

The `curl` command

In linux, `curl` is a tool to transfer data from or to a server, using one of the supported protocols(DICT, FILE ,FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP).

Example :

```
$ curl example.com
```

The command will print the source code of the example.com homepage in the terminal window.

The syntax of the `curl` command is :

```
$ curl [options...] <url>
```

Options :

Options start with one or two dashes. Many of the options require an additional value next to them.

The short "single-dash" form of the options, `-d` for example, may be used with or without a space between it and its value, although a space is a recommended separator. The long "double-dash" form, `-d`, `--data` for example, requires a space between it and its value.

Short version options that don't need any additional values can be used immediately next to each other, like for example you can specify all the options `-O`, `-L` and `-v` at once as `-OLv`.

In general, all boolean options are enabled with `--option` and yet again disabled with `--no-option`. That is, you use the exact same option name but prefix it with `no-`. However, in this list we mostly only list and show the `--option` version of them. (This concept with `--no` options was added in 7.19.0. Previously most options were toggled on/off through repeated use of the same command line option.)

Installation:

The curl command comes with most of the Linux distributions. But, if the system does not carry the curl by default. You need to install it manually. To install the curl, execute the following commands:

Update the system by executing the following commands:

```
$ sudo apt update
$ sudo apt upgrade
```

Now, install the curl utility by executing the below command:

```
$ sudo apt install curl
```

Verify the installation by executing the below command:

```
$ curl -version
```

The above command will display the installed version of the curl command.

The `yes` command

The `yes` command in linux is used to print a continuous output stream of given *STRING*. If *STRING* is not mentioned then it prints 'y'. It outputs a string repeatedly unit killed (using something like ctrl + c).

Examples :

1. Prints hello world infinitely in the terminal until killed :

```
yes hello world
```

2. A more generalized command:

```
yes [STRING]
```

Options

It accepts the following options:

1. --help
   display this help and exit

2. --version
   output version information and exit

The `last` command

This command shows you a list of all the users that have logged in and out since the creation of the `var/log/wtmp` file. There are also some parameters you can add which will show you for example when a certain user has logged in and how long he was logged in for.

If you want to see the last 5 logs, just add `-5` to the command like this:

```
last -5
```

And if you want to see the last 10, add `-10`.

Another cool thing you can do is if you add `-F` you can see the login and logout time including the dates.

```
last -F
```

There are quite a lot of stuff you can view with this command. If you need to find out more about this command you can run:

```
last --help
```

The `locate` command

The `locate` command searches the file system for files and directories whose name matches a given pattern through a database file that is generated by the `updatedb` command.

Examples:

1. Running the `locate` command to search for a file named `.bashrc`.

```
locate .bashrc
```

Output

```
/etc/bash.bashrc
/etc/skel/.bashrc
/home/linuxize/.bashrc
/usr/share/base-files/dot.bashrc
/usr/share/doc/adduser/examples/adduser.local.conf.examples/ba
sh.bashrc
/usr/share/doc/adduser/examples/adduser.local.conf.examples/sk
el/dot.bashrc
```

The `/root/.bashrc` file will not be shown because we ran the command as a normal user that doesn't have access permissions to the `/root` directory.

If the result list is long, for better readability, you can pipe the output to the `less` command:

```
locate .bashrc | less
```

2. To search for all `.md` files on the system

```
locate *.md
```

3. To search all `.py` files and display only 10 results

```
locate -n 10 *.py
```

4. To performs case-insensitive search.

```
locate -i readme.md
```

Output

```
/home/linuxize/p1/readme.md
/home/linuxize/p2/README.md
/home/linuxize/p3/ReadMe.md
```

5. To return the number of all files containing `.bashrc` in their name.

```
locate -c .bashrc
```

Output

```
6
```

6. The following would return only the existing `.json` files on the file system.

```
locate -e *.json
```

7. To run a more complex search the `-r` (`--regexp`) option is used. To search for all `.mp4` and `.avi` files on your system and ignore case.

```
locate --regex -i "(\.mp4|\.avi)"
```

Syntax:

```
1.  locate [OPTION]... PATTERN...
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
-A	--all	It is used to display only entries that match all PATTERNs instead of requiring only one of them to match.
-b	--basename	It is used to match only the base name against the specified patterns.
-c	--count	It is used for writing the number matching entries instead of writing file names on standard output.
-d	--database DBPATH	It is used to replace the default database with DBPATH.
-e	--existing	It is used to display only entries that refer to existing files during the command is executed.
-L	--follow	If the --existing option is specified, It is used for checking whether files exist and follow trailing symbolic links. It will omit the broken symbolic links to the output. This is the default behavior. The opposite behavior can be specified using the --nofollow option.
-h	--help	It is used to display the help documentation that contains a summary of the available options.
-i	--ignore-case	It is used to ignore case sensitivity of the specified patterns.
-p	--ignore-spaces	It is used to ignore punctuation and spaces when matching patterns.
-t	--transliterate	It is used to ignore accents using iconv transliteration when matching patterns.
-l	--limit, -n LIMIT	If this option is specified, the command exit successfully after finding LIMIT entries.
-m	--mmap	It is used to ignore the compatibility with BSD, and GNU locate.

Short Flag	Long Flag	Description
`-0`	`--null`	It is used to separate the entries on output using the ASCII NUL character instead of writing each entry on a separate line.
`-S`	`--statistics`	It is used to write statistics about each read database to standard output instead of searching for files.
`-r`	`--regexp REGEXP`	It is used for searching a basic regexp REGEXP.
`--regex`	-	It is used to describe all PATTERNs as extended regular expressions.
`-V`	`--version`	It is used to display the version and license information.
`-w`	`--wholename`	It is used for matching only the whole path name in specified patterns.

The `iostat` command

The `iostat` command in Linux is used for monitoring system input/output statistics for devices and partitions. It monitors system input/output by observing the time the devices are active in relation to their average transfer rates. The iostat produce reports may be used to change the system configuration to raised balance the input/output between the physical disks. iostat is being included in sysstat package. If you don't have it, you need to install first.

Syntax:

```
iostat [ -c ] [ -d ] [ -h ] [ -N ] [ -k | -m ] [ -t ] [ -V ] [
-x ]
       [ -z ] [ [ [ -T ] -g group_name ] { device [...] | ALL
} ]
       [ -p [ device [,...] | ALL ] ] [ interval [ count ] ]
```

Examples:

1. Display a single history-since-boot report for all CPU and Devices:

```
iostat -d 2
```

2. Display a continuous device report at two-second intervals:

```
iostat -d 2 6
```

3.Display, for all devices, six reports at two-second intervals:

```
iostat -x sda sdb 2 6
```

4.Display, for devices sda and sdb, six extended reports at two-second intervals:

```
iostat -p sda 2 6
```

Additional Flags and their Functionalities:

Short Flag	Description
`-x`	Show more details statistics information.
`-c`	Show only the cpu statistic.
`-d`	Display only the device report
`-xd	Show extended I/O statistic for device only.
`-k`	Capture the statistics in kilobytes or megabytes.
`-k23`	Display cpu and device statistics with delay.
`-j ID mmcbkl0 sda6 -x -m 2 2`	Display persistent device name statistics.
`-p`	Display statistics for block devices.
`-N`	Display lvm2 statistic information.

The `sudo` command

The `sudo` ("substitute user do" or "super user do") command allows a user with proper permissions to execute a command as another user, such as the superuser.

This is the equivalent of "run as administrator" option in Windows. The `sudo` command allows you to elevate your current user account to have root privileges. Also, the root privilege in `sudo` is only valid for a temporary amount of time. Once that time expires, you have to enter your password again to regain root privilege.

WARNING: Be very careful when using the `sudo` command. You can cause irreversible and catastrophic changes while acting as root!

Syntax:

```
sudo [-OPTION] command
```

Additional Flags and their Functionalities:

Flag	Description
`-V`	The -V (version) option causes sudo to print the version number and exit. If the invoking user is already root, the -V option prints out a list of the defaults sudo was compiled with and the machine's local network addresses
`-l`	The -l (list) option prints out the commands allowed (and forbidden) the user on the current host.
`-L`	The -L (list defaults) option lists out the parameters set in a Defaults line with a short description for each. This option is useful in conjunction with grep.
`-h`	The -h (help) option causes sudo to print a usage message and exit.
`-v`	If given the `-v` (validate) option, `sudo` updates the user's timestamp, prompting for the user's password if necessary. This extends the sudo timeout for another 5 minutes (or whatever the timeout is set to in sudoers) but does not run a command.
`-K`	The -K (sure kill) option to sudo removes the user's timestamp entirely. Likewise, this option does not require a password.

Flag	Description
`-u`	The -u (user) option causes sudo to run the specified command as a user other than root. To specify a uid instead of a username, use #uid.
`-s`	The -s (shell) option runs the shell specified by the SHELL environment variable if it's set or the shell as specified in the file passwd.
`--`	The -- flag indicates that sudo should stop processing command line arguments. It is most useful in conjunction with the -s flag.

Examples

This command switches your command prompt to the BASH shell as a root user:

```
sudo bash
```

Your command line should change to:

```
root@hostname:/home/[username]
```

Adding a string of text to a file is often used to add the name of a software repository to the sources file, without opening the file for editing. Use the following syntax with echo, sudo and tee command:

```
echo ‘string-of-text’ | sudo tee -a [path_to_file]
```

Example:

```
echo "deb http://nginx.org/packages/debian `lsb_release -cs`
nginx" \ | sudo tee /etc/apt/sources.list.d/nginx.list
```

The `apt` command

`apt` (Advantage package system) command is used for interacting with `dpkg` (packaging system used by debian). There is already the `dpkg` command to manage `.deb` packages. But `apt` is a more user-friendly and efficient way.

In simple terms `apt` is a command used for installing, deleting and performing other operations on debian based Linux.

You will be using the `apt` command mostly with `sudo` privileges.

Installing packages:

`install` followed by `package_name` is used with `apt` to install a new package.

Syntax:

```
sudo apt install package_name
```

Example:

```
sudo apt install g++
```

This command will install g++ on your system.

Removing packages:

`remove` followed by `package_name` is used with `apt` to remove a specific package.

Syntax:

```
sudo apt remove package_name
```

Example:

```
sudo apt remove g++
```

This command will remove g++ from your system.

Searching for a package:

`search` followed by the `package_name` used with apt to search a package across all repositories.

Syntax:

```
apt search package_name
```

note: sudo not required

Example:

```
apt search g++
```

Removing unused packages:

Whenever a new package that depends on other packages is installed on the system, the package dependencies will be installed too. When the package is removed, the dependencies will stay on the system. This leftover packages are no longer used by anything else and can be removed.

Syntax:

```
sudo apt autoremove
```

This command will remove all unused from your system.

Updating package index:

`apt` package index is nothing but a database that stores records of available packages that are enabled on your system.

Syntax:

```
sudo apt update
```

This command will update the package index on your system.

Upgrading packages:

If you want to install the latest updates for your installed packages you may want to run this command.

Syntax:

```
sudo apt upgrade
```

The command doesn't upgrade any packages that require removal of installed packages.

If you want to upgrade a single package, pass the package name:

Syntax:

```
sudo apt upgrade package_name
```

This command will upgrade your packages to the latest version.

The `yum` command

The `yum`command is the primary package management tool for installing, updating, removing, and managing software packages in Red Hat Enterprise Linux. It is an acronym for *`Yellow Dog Updater, Modified`*.

`yum` performs dependency resolution when installing, updating, and removing software packages. It can manage packages from installed repositories in the system or from .rpm packages.

Syntax:

```
yum -option command
```

Examples:

1. To see an overview of what happened in past transactions:

```
yum history
```

2. To undo a previous transaction:

```
yum history undo <id>
```

3. To install firefox package with 'yes' as a response to all confirmations

```
yum -y install firefox
```

4. To update the mysql package it to the latest stable version

```
yum update mysql
```

Commonly used commands along with yum:

Command	Description
`install`	Installs the specified packages
`remove`	Removes the specified packages
`search`	Searches package metadata for keywords
`info`	Lists the description
`update`	Updates each package to the latest version
`repolist`	Lists repositories
`history`	Displays what has happened in past transactions
`groupinstall`	To install a particular package group
`clean`	To clean all cached files from enabled repository

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-C`	`--cacheonly`	Runs entirely from system cache, doesn't update the cache and use it even in case it is expired.
-	`--security`	Includes packages that provide a fix for a security issue. Applicable for the upgrade command.
`-y`	`--assumeyes`	Automatically answer yes for all questions.
-	`--skip-broken`	Resolves depsolve problems by removing packages that are causing problems from the transaction. It is an alias for the strict configuration option with value False.
`-v`	`--verbose`	Verbose operation, show debug messages.

The `zip` command

The `zip` command is used to compress files and reduce their size. It outputs an archive containing one or more compressed files or directories.

Examples:

In order to compress a single file with the `zip` command the syntax would be the following:

```
zip myZipFile.zip filename.txt
```

This also works with multiple files as well:

```
zip multipleFiles.zip file1.txt file2.txt
```

If you are compressing a whole directory, don't forget to add the `-r` flag:

```
zip -r zipFolder.zip myFolder/
```

Syntax:

```
zip [OPTION] zipFileName filesList
```

Possible options:

Flag	Description
`-d`	Removes the file from the zip archive. After creating a zip file, you can remove a file from the archive using the `-d` option

Flag	Description
`-u`	Updates the file in the zip archive. This option can be used to update the specified list of files or add new files to the existing zip file. Update an existing entry in the zip archive only if it has been modified more recently than the version already in the zip archive.
`-m`	Deletes the original files after zipping.
`-r`	To zip a directory recursively, it will recursively zip the files in a directory. This option helps to zip all the files present in the specified directory.
`-x`	Exclude the files in creating the zip
`-v`	Verbose mode or print diagnostic version info. Normally, when applied to real operations, this option enables the display of a progress indicator during compression and requests verbose diagnostic info about zip file structure oddities

The `unzip` command

The `unzip` command extracts all files from the specified ZIP archive to the current directory.

Examples:

In order to extract the files the syntax would be the following:

```
unzip myZipFile.zip
```

To unzip a ZIP file to a different directory than the current one, don't forget to add the `-d` flag:

```
unzip myZipFile.zip -d /path/to/directory
```

To unzip a ZIP file and exclude specific file or files or directories from being extracted, don't forget to add the `-x` flag:

```
unzip myZipFile.zip -x file1.txt file2.txt
```

Syntax:

```
unzip zipFileName [OPTION] [PARAMS]
```

Possible options:

Flag	Description	Params
`-d`	Unzip an archive to a different directory.	/path/to/directory

Flag	Description	Params
`-x`	Extract the archive but do not extract the specified files.	filename(s)
`-j`	Unzip without creating new folders, if the zipped archive contains a folder structure.	-
`-l`	Lists the contents of an archive file without extracting it.	-
`-n`	Do not overwrite existing files; supply an alternative filename instead.	-
`-o`	Overwrite files.	-
`-P`	Supplies a password to unzip a protected archive file.	password
`-q`	Unzips without writing status messages to the standard output.	-
`-t`	Tests whether an archive file is valid.	-
`-v`	Displays detailed (verbose) information about the archive without extracting it.	-

The `shutdown` command

The `shutdown` command lets you bring your system down in a secure way. When `shutdown` is executed the system will notify all logged-in users and disallow further logins. You have the option to shut down your system immediately or after a specific time.

Only users with root (or sudo) privileges can use the `shutdown` command.

Examples:

1. Shut down your system immediately:

```
sudo shutdown now
```

2. Shut down your system after 10 minutes:

```
sudo shutdown +10
```

3. Shut down your system with a message after 5 minutes:

```
sudo shutdown +5 "System will shutdown in 5 minutes"
```

Syntax:

```
shutdown [OPTIONS] [TIME] [MESSAGE]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-r`	-	Reboot the system
`-c`	-	Cancel an scheduled shut down

The `dir` command

The `dir` command lists the contents of a directory(*the current directory by default*). **It differs from ls command in the format of listing the content**. By default, the dir command lists the files and folders in columns, sorted vertically and special characters are represented by backslash escape sequences.

Syntax:

```
dir [OPTIONS] [FILE]
```

Examples:

1. To list files in the current directory:

```
dir
```

2. To list even the hidden files in the current directory:

```
dir -a
```

3. To list the content with detailed information for each entry

```
dir -l
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-a`	`--all`	It displays all the hidden files(starting with `.`) along with two files denoted by `.` and `..`
`-A`	`--almost-all`	It is **similar to -a** option except that it *does not display files that signals the current directory and previous directory.*
`-l`	-	Display detailed information for each entry
`-s`	`--size`	Print the allocated size of each file, in blocks File
`-h`	`--human-readable`	Used with with -l and -s, to print sizes like in human readable format like 1K, 2M and so on
`-F`	-	Classifies entries into their type based on appended symbol (`/`, `*`, `@`, `%`, `=`)
`-v`	`--verbose`	Print source and destination files
-	`--group-directories-first`	To group directories before files
`-R`	`--recursive`	To List subdirectories recursively.
`-S`	-	sort by file size, display largest first

The `reboot` Command

The `reboot` command is used to restart a linux system. However, it requires elevated permission using the sudo command. Necessity to use this command usually arises after significant system or network updates have been made to the system.

Syntax

```
reboot [OPTIONS...]
```

Options

- **-help** : This option prints a short help text and exit.
- **-halt** : This command will stop the machine.
- **-w**, **–wtmp-only** : This option only writes wtmp shutdown entry, it do not actually halt, power-off, reboot.

Examples

1. Basic Usage. Mainly used to restart without any further details

```
$ sudo reboot
```

However, alternatively the shutdown command with the `-r` option

```
$ sudo shutdown -r now
```

Note that the usage of the reboot, halt and power off is almost similar in syntax and effect. Run each of these commands with –help to see the details.

2. The `reboot` command has limited usage, and the `shutdown` command is being used instead of reboot command to fulfill much more advance reboot and shutdown requirements. One of those situations is a scheduled restart. Syntax is as follows

```
$ sudo shutdown –r [TIME] [MESSAGE]
```

Here the TIME has various formats. The simplest one is `now`, already been listed in the previous section, and tells the system to restart immediately. Other valid formats we have are +m, where m is the number of minutes we need to wait until restart and

HH:MM which specifies the TIME in a 24hr clock.

Example to reboot the system in 2 minutes

```
$ sudo shutdown –r +2
```

Example of a scheduled restart at 03:00 A.M

```
$ sudo shutdown –r 03:00
```

3. Cancelling a Reboot. Usually happens in case one wants to cancel a scheduled restart

Syntax

```
$ sudo shutdown –c [MESSAGE]
```

Usage

```
$sudo shutdown -c "Scheduled reboot cancelled because the
chicken crossed the road"
```

4. Checking your reboot logs

```
$ last reboot
```

The `sort` command

the `sort` command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts a file assuming the contents are ASCII. Using options in the sort command can also be used to sort numerically.

Examples:

Suppose you create a data file with name file.txt:

```
Command :
$ cat > file.txt
abhishek
chitransh
satish
rajan
naveen
divyam
harsh
```

Sorting a file: Now use the sort command

Syntax :

```
sort filename.txt
```

```
Command:
$ sort file.txt

Output :
abhishek
chitransh
divyam
harsh
naveen
rajan
satish
```

Note: This command does not actually change the input file, i.e. file.txt.

The sort function on a file with mixed case content

i.e. uppercase and lower case: When we have a mix file with both uppercase and lowercase letters then first the upper case letters would be sorted following with the lower case letters.

Example:

Create a file mix.txt

```
Command :
$ cat > mix.txt
abc
apple
BALL
Abc
bat
```

Now use the sort command

```
Command :
$ sort mix.txt
Output :
Abc
BALL
abc
apple
bat
```

The `paste` command

The `paste` command writes lines of two or more files, sequentially and separated by TABs, to the standard output

Syntax:

```
paste [OPTIONS]... [FILE]...
```

Examples:

1. To paste two files

```
paste file1 file2
```

2. To paste two files using new line as delimiter

```
paste -d '\n' file1 file2
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-d`	`--delimiter`	use charater of TAB
`-s`	`--serial`	paste one file at a time instead of in parallel
`-z`	`--zero-terminated`	set line delimiter to NUL, not newline
	`--help`	print command help
	`--version`	print version information

The `exit` command

The `exit` command is used to terminate (close) an active shell session

Syntax:

```
exit
```

Shortcut: *Instead of typing `exit`, press `ctrl + D`, it will do the same Functionality.*

The `diff/sdiff` command

This command is used to display the differences in the files by comparing the files line by line.

Syntax:

```
diff [options] File1 File2
```

Example

1. Lets say we have two files with names a.txt and b.txt containing 5 Indian states as follows-:

```
$ cat a.txt
Gujarat
Uttar Pradesh
Kolkata
Bihar
Jammu and Kashmir

$ cat b.txt
Tamil Nadu
Gujarat
Andhra Pradesh
Bihar
Uttar pradesh
```

On typing the diff command we will get below output.

```
$ diff a.txt b.txt
0a1
> Tamil Nadu
2,3c3
< Uttar Pradesh
 Andhra Pradesh
5c5
 Uttar pradesh
```

Flags and their Functionalities

Short Flag	Description
`-c`	To view differences in context mode, use the -c option.
`-u`	To view differences in unified mode, use the -u option. It is similar to context mode
`-i`	By default this command is case sensitive. To make this command case insensitive use -i option with diff.
`-version`	This option is used to display the version of diff which is currently running on your system.

The `tar` command

The `tar` command stands for tape archive, is used to create Archive and extract the Archive files. This command provides archiving functionality in Linux. We can use tar command to create compressed or uncompressed Archive files and also maintain and modify them.

Examples:

1. To create a tar file in abel directory:

```
tar -cvf file-14-09-12.tar /home/abel/
```

2. To un-tar a file in the current directory:

```
 tar -xvf file-14-09-12.tar
```

Syntax:

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

Additional Flags and their Functionalities:

Use Flag	Description
`-c`	Creates Archive
`-x`	Extract the archive
`-f`	Creates archive with given filename
`-t`	Displays or lists files in archived file
`-u`	Archives and adds to an existing archive file
`-v`	Displays Verbose Information

Use Flag	Description
-A	Concatenates the archive files
-z	zip, tells tar command that creates tar file using gzip
-j	Filter archive tar file using tbzip
w	Verify a archive file
r	update or add file or directory in already existed .tar file
-?	Displays a short summary of the project
-d	Find the difference between an archive and file system
--usage	shows available tar options
--version	Displays the installed tar version
--show-defaults	Shows default enabled options

Option Flag	Description
--check-device	Check device numbers during incremental archive
-g	Used to allow compatibility with GNU-format incremental ackups
--hole-detection	Used to detect holes in the sparse files
-G	Used to allow compatibility with old GNU-format incremental backups
--ignore-failed-read	Don't exit the program on file read errors
--level	Set the dump level for created archives
-n	Assume the archive is seekable
--no-check-device	Do not check device numbers when creating archives
--no-seek	Assume the archive is not seekable
--occurrence=N	`Process only the Nth occurrence of each file
--restrict	`Disable use of potentially harmful options
--sparse-version=MAJOR,MINOR	Set version of the sparce format to use
-S	Handle sparse files efficiently.

Overwright control Flag	Description
-k	Don't replace existing files
--keep-newer-files	Don't replace existing files that are newer than the archives version
--keep-directory-symlink	Don't replace existing symlinks
--no-overwrite-dir	Preserve metadata of existing directories
--one-top-level=DIR	Extract all files into a DIR
--overwrite	Overwrite existing files
--overwrite-dir	Overwrite metadata of directories
--recursive-unlink	Recursivly remove all files in the directory before extracting

Overwright control Flag	Description
`--remove-files`	Remove files after adding them to a directory
`--skip-old-files`	Don't replace existing files when extracting
`-u`	Remove each file before extracting over it
`-w`	Verify the archive after writing it

The `gunzip` command

The `gunzip` command is an antonym command of `gzip` command. In other words, it decompresses files deflated by the `gzip` command.

`gunzip` takes a list of files on its command line and replaces each file whose name ends with *.gz*, *-gz*, *.z*, *-z*, or *_z* (ignoring case) and which begins with the correct magic number with an uncompressed file without the original extension. `gunzip` also recognizes the special extensions *.tgz* and *.taz* as shorthands for *.tar.gz* and *.tar.Z* respectively.

Examples:

1. Uncompress a file

```
gunzip filename.gz
```

2. Recursively uncompress content inside a directory, that match extension (suffix) compressed formats accepted by `gunzip`:

```
gunzip -r directory_name/
```

3. Uncompress all files in the current/working directory whose suffix match *.tgz*:

```
gunzip -S .tgz *
```

4. List compressed and uncompressed sizes, compression ratio and uncompressed name of input compressed file/s:

```
gunzip -l file_1 file_2
```

Syntax:

```
gunzip [ -acfhklLnNrtvV ] [-S suffix] [ name ...  ]
```

Video tutorial about using gzip, gunzip and tar commands:

This video shows how to compress and decompress in a Unix shell. It uses `gunzip` as decompression command.

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
-c	--stdout	write on standard output, keep original files unchanged
-h	--help	give help information
-k	--keep	keep (don't delete) input files
-l	--list	list compressed file contents
-q	--quiet	suppress all warnings
-r	--recursive	operate recursively on directories
-S	--suffix=SUF	use suffix SUF on compressed files
	--synchronous	synchronous output (safer if system crashes, but slower)
-t	--test	test compressed file integrity
-v	--verbose	verbose mode
-V	--version	display version number

The `hostnamectl` command

The `hostnamectl` command provides a proper API used to control Linux system hostname and change its related settings. The command also helps to change the hostname without actually locating and editing the `/etc/hostname` file on a given system.

Syntax

```
$ hostnamectl [OPTIONS...] COMMAND ...
```

where **COMMAND** can be any of the following

status: Used to check the current hostname settings

set-hostname NAME: Used to set system hostname

set-icon-name NAME: Used to set icon name for host

Example

1. Basic usage to view the current hostnames

```
$ hostnamectl
```

or

```
$ hostnamectl status
```

2. To change the static host name to *myhostname*. It may or may not require root access

```
$ hostnamectl set-hostname myhostname --static
```

3. To set or change a transient hostname

```
$ hostnamectl set-hostname myotherhostname --transient
```

4. To set the pretty hostname. The name that is to be set needs to be in the double quote(” “).

```
$ hostname set-hostname "prettyname" --pretty
```

The `iptables` Command

The `iptables` command is used to set up and maintain tables for the Netfilter firewall for IPv4, included in the Linux kernel. The firewall matches packets with rules defined in these tables and then takes the specified action on a possible match.

Syntax:

```
iptables --table TABLE -A/-C/-D... CHAIN rule --jump Target
```

Example and Explanation:

This command will append to the chain provided in parameters:

```
iptables [-t table] --append [chain] [parameters]
```

This command drops all the traffic coming on any port:

```
iptables -t filter --append INPUT -j DROP
```

Flags and their Functionalities:

Flag	Description
`-C`	Check if a rule is present in the chain or not. It returns 0 if the rule exists and returns 1 if it does not.
`-A`	Append to the chain provided in parameters.

The `netstat` command

The term `netstat` stands for Network Statistics. In layman's terms, netstat command displays the current network connections, networking protocol statistics, and a variety of other interfaces.

Check if you have `netstat` on your PC:

```
netstat –v
```

If you don't have `netstat` installed on your PC, you can install it with the following command:

```
sudo apt install net-tools
```

You can use `netstat` command for some use cases given below:

- `Netstat` command with `-nr` flag shows the routing table detail on the terminal.

Example:

```
netstat  -nr
```

- `Netstat` command with `-i` flag shows statistics for the currently configured network interfaces. This command will display the first 10 lines of file `foo.txt` .

Example:

```
netstat  -i
```

- `Netstat` command with `-tunlp` will gives a list of networks, their current states, and their associated ports.

Example:

```
netstat -tunlp
```

- You can get the list of all TCP port connection by using `-at` with `netstat`.

```
netstat  -at
```

- You can get the list of all UDP port connection by using `-au` with `netstat`.

```
netstat  -au
```

- You can get the list of all active connection by using `-l` with `netstat`.

```
netstat  -l
```

The `lsof` command

The `lsof` command shows **file infomation** of all the files opened by a running process. It' name is also derived from the fact that, list open files > `lsof`

An open file may be a regular file, a directory, a block special file, a character special file, an executing text reference, a library , a stream or a network file (Internet socket, NFS file or UNIX domain socket). A specific file or all the files in a file system may be selected by path.

Syntax:

```
lsof [-OPTION] [USER_NAME]
```

Examples:

1. To show all the files opened by all active processes:

```
lsof
```

2. To show the files opened by a particular user:

```
lsof -u [USER_NAME]
```

3. To list the processes with opened files under a specified directory:

```
lsof +d [PATH_TO_DIR]
```

Options and their Functionalities:

Option	Additional Options	Description
`-i`	`tcp`/ `udp`/ `:port`	List all network connections running, Additionally, on udp/tcp or on specified port.
`-i4`	-	List all processes with ipv4 connections.
`-i6`	-	List all processes with ipv6 connections.
`-c`	`[PROCESS_NAME]`	List all the files of a particular process with given name.
`-p`	`[PROCESS_ID]`	List all the files opened by a specified process id.
`-p`	`^[PROCESS_ID]`	List all the files that are not opened by a specified process id.
`+d`	`[PATH]`	List the processes with opened files under a specified directory
`+R`	-	List the files opened by parent process Id.

Help Command

Run below command to view the complete guide to `lsof` command.

```
man lsof
```

The `bzip2` command

The `bzip2` command lets you compress and decompress the files i.e. it helps in binding the files into a single file which takes less storage space as the original file use to take.

Syntax:

```
bzip2 [OPTIONS] filenames ...
```

Note : Each file is replaced by a compressed version of itself, with the name original name of the file followed by extension bz2.

Options and their Functionalities:

Option	Alias	Description
`-d`	`--decompress`	to decompress compressed file
`-f`	`--force`	to force overwrite an existing output file
`-h`	`--help`	to display the help message and exit
`-k`	`--keep`	to enable file compression, doesn't deletes the original input file
`-L`	`--license`	to display the license terms and conditions
`-q`	`--quiet`	to suppress non-essential warning messages
`-t`	`--test`	to check integrity of the specified .bz2 file, but don't want to decompress them
`-v`	`--erbose`	to display details for each compression operation
`-V`	`--version`	to display the software version
`-z`	`--compress`	to enable file compression, but deletes the original input file

By default, when bzip2 compresses a file, it deletes the original (or input) file. However, if you don't want that to happen, use the -k command line option.

Examples:

1. To force compression:

```
bzip2 -z input.txt
```

Note: This option deletes the original file also

2. To force compression and also retain original input file:

```
bzip2 -k input.txt
```

3. To force decompression:

```
bzip2 -d input.txt.bz2
```

4. To test integrity of compressed file:

```
bzip2 -t input.txt.bz2
```

5. To show the compression ratio for each file processed:

```
bzip2 -v input.txt
```

The `service` command

Service runs a System V init script in as predictable environment as possible, removing most environment variables and with current working directory set to /.

The SCRIPT parameter specifies a System V init script, located in /etc/init.d/SCRIPT. The supported values of COMMAND depend on the invoked script, service passes COMMAND and OPTIONS it to the init script unmodified. All scripts should support at least the start and stop commands. As a special case, if COMMAND is --full-restart, the script is run twice, first with the stop command, then with the start command.

The COMMAND can be at least start, stop, status, and restart.

service --status-all runs all init scripts, in alphabetical order, with the `status` command

Examples :

1. To check the status of all the running services:

```
service --status-all
```

2. To run a script

```
service SCRIPT-Name start
```

3. A more generalized command:

```
service [SCRIPT] [COMMAND] [OPTIONS]
```

The `vmstat` command

The `vmstat` command lets you monitor the performance of your system. It shows you information about your memory, disk, processes, CPU scheduling, paging, and block IO. This command is also referred to as **virtual memory statistic report**.

The very first report that is produced shows you the average details since the last reboot and after that, other reports are made which report over time.

`vmstat`

As you can see it is a pretty useful little command. The most important things that we see above are the `free`, which shows us the free space that is not being used, `si` shows us how much memory is swapped in every second in kB, and `so` shows how much memory is swapped out each second in kB as well.

`vmstat -a`

If we run `vmstat -a`, it will show us the active and inactive memory of the system running.

`vmstat -d`

The `vmstat -d` command shows us all the disk statistics.

As you can see this is a pretty useful little command that shows you different statistics about your virtual memory

The `mpstat` command

The `mpstat` command is used to report processor related statistics. It accurately displays the statistics of the CPU usage of the system and information about CPU utilization and performance.

Syntax:

```
mpstat [options] [<interval> [<count>]]
```

Note : It initializes the first processor with CPU 0, the second one with CPU 1, and so on.

Options and their Functionalities:

Option	Description
`-A`	to display all the detailed statistics
`-h`	to display mpstat help
`-I`	to display detailed interrupts statistics
`-n`	to report summary CPU statistics based on NUMA node placement
`-N`	to indicate the NUMA nodes for which statistics are to be reported
`-P`	to indicate the processors for which statistics are to be reported
`-o`	to display the statistics in JSON (Javascript Object Notation) format
`-T`	to display topology elements in the CPU report
`-u`	to report CPU utilization
`-v`	to display utilization statistics at the virtual processor level
`-V`	to display mpstat version
`-ALL`	to display detailed statistics about all CPUs

Examples:

1. To display processor and CPU statistics:

```
mpstat
```

2. To display processor number of all CPUs:

```
mpstat -P ALL
```

3. To get all the information which the tool may collect:

```
mpstat -A
```

4. To display CPU utilization by a specific processor:

```
mpstat -P 0
```

5. To display CPU usage with a time interval:

```
mpstat 1 5
```

Note: This command will print 5 reports with 1 second time interval

The `ncdu` Command

`ncdu` (NCurses Disk Usage) is a curses-based version of the well-known `du` command. It provides a fast way to see what directories are using your disk space.

Example

1. Quiet Mode

```
ncdu -q
```

2. Omit mounted directories

```
ncdu -q -x
```

Syntax

```
ncdu [-hqvx] [--exclude PATTERN] [-X FILE] dir
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-h`	-	Print a small help message
`-q`	-	Quiet mode. While calculating disk space, ncdu will update the screen 10 times a second by default, this will be decreased to once every 2 seconds in quiet mode. Use this feature to save bandwidth over remote connections.
`-v`	-	Print version.
`-x`	-	Only count files and directories on the same filesystem as the specified dir.
-	`--exclude PATTERN`	Exclude files that match PATTERN. This argument can be added multiple times to add more patterns.
`-X FILE`	`--exclude-from FILE`	Exclude files that match any pattern in FILE. Patterns should be separated by a newline.

The `uniq` command

The `uniq` command in Linux is a command line utility that reports or filters out the repeated lines in a file. In simple words, `uniq` is the tool that helps you to detect the adjacent duplicate lines and also deletes the duplicate lines. It filters out the adjacent matching lines from the input file(that is required as an argument) and writes the filtered data to the output file .

Examples:

In order to omit the repeated lines from a file, the syntax would be the following:

```
uniq kt.txt
```

In order to tell the number of times a line was repeated, the syntax would be the following:

```
uniq -c kt.txt
```

In order to print repeated lines, the syntax would be the following:

```
uniq -d kt.txt
```

In order to print unique lines, the syntax would be the following:

```
uniq -u kt.txt
```

In order to allows the N fields to be skipped while comparing uniqueness of the lines, the syntax would be the following:

```
uniq -f 2 kt.txt
```

In order to allows the N characters to be skipped while comparing uniqueness of the lines, the syntax would be the following:

```
uniq -s 5 kt.txt
```

In order to to make the comparison case-insensitive, the syntax would be the following:

```
uniq -i kt.txt
```

Syntax:

```
uniq [OPTION] [INPUT[OUTPUT]]
```

Possible options:

Flag	Description	Params
`-c`	It tells how many times a line was repeated by displaying a number as a prefix with the line.	-
`-d`	It only prints the repeated lines and not the lines which aren't repeated.	-
`-i`	By default, comparisons done are case sensitive but with this option case insensitive comparisons can be made.	-
`-f`	It allows you to skip N fields(a field is a group of characters, delimited by whitespace) of a line before determining uniqueness of a line.	N
`-s`	It doesn't compares the first N characters of each line while determining uniqueness. This is like the -f option, but it skips individual characters rather than fields.	N
`-u`	It allows you to print only unique lines.	-
`-z`	It will make a line end with 0 byte(NULL), instead of a newline.	-
`-w`	It only compares N characters in a line.	N
`--help`	It displays a help message and exit.	-
`--version`	It displays version information and exit.	-

The `RPM` command

`rpm` - RPM Package Manager

`rpm` is a powerful **Package Manager**, which can be used to build, install, query, verify, update, and erase individual software packages. A **package** consists of an archive of files and meta-data used to install and erase the archive files. The meta-data includes helper scripts, file attributes, and descriptive information about the package. Packages come in two varieties: binary packages, used to encapsulate software to be installed, and source packages, containing the source code and recipe necessary to produce binary packages.

One of the following basic modes must be selected: **Query, Verify, Signature Check, Install/Upgrade/Freshen, Uninstall, Initialize Database, Rebuild Database, Resign, Add Signature, Set Owners/Groups, Show Querytags, and Show Configuration.**

General Options

These options can be used in all the different modes.

Short Flag	Long Flag	Description
-?	--help	Print a longer usage message then normal.
-	--version	Print a single line containing the version number of rpm being used.
-	--quiet	Print as little as possible - normally only error messages will be displayed.
-v	-	Print verbose information - normally routine progress messages will be displayed.
-vv	-	Print lots of ugly debugging information.
-	--rcfile FILELIST	Each of the files in the colon separated FILELIST is read sequentially by rpm for configuration information. Only the first file in the list must exist, and tildes will be expanded to the value of $HOME. The default FILELIST is /usr/lib/rpm/rpmrc:/usr/lib/rpm/redhat/rpmrc:/etc/rpmrc:~/.rpmrc.
-	--pipe CMD	Pipes the output of rpm to the command CMD.
-	--dbpath DIRECTORY	Use the database in DIRECTORY rather than the default path /var/lib/rpm

Short Flag	Long Flag	Description
-	--root DIRECTORY	Use the file system tree rooted at DIRECTORY for all operations. Note that this means the database within DIRECTORY will be used for dependency checks and any scriptlet(s) (e.g. %post if installing, or %prep if building, a package) will be run after a chroot(2) to DIRECTORY.
-D	--define='MACRO EXPR'	Defines MACRO with value EXPR.
-E	--eval='EXPR'	Prints macro expansion of EXPR.

Synopsis

Querying and Verifying Packages:

```
rpm {-q|--query} [select-options] [query-options]

rpm {-V|--verify} [select-options] [verify-options]

rpm --import PUBKEY ...

rpm {-K|--checksig} [--nosignature] [--nodigest] PACKAGE_FILE
...
```

Installing, Upgrading, and Removing Packages:

```
rpm {-i|--install} [install-options] PACKAGE_FILE ...

rpm {-U|--upgrade} [install-options] PACKAGE_FILE ...

rpm {-F|--freshen} [install-options] PACKAGE_FILE ...

rpm {-e|--erase} [--allmatches] [--nodeps] [--noscripts] [--
notriggers] [--test] PACKAGE_NAME ...
```

Miscellaneous:

```
rpm {--initdb|--rebuilddb}

rpm {--addsign|--resign} PACKAGE_FILE...

rpm {--querytags|--showrc}

rpm {--setperms|--setugids} PACKAGE_NAME .
```

query-options

```
[--changelog] [-c,--configfiles] [-d,--docfiles] [--dump]
[--filesbypkg] [-i,--info] [--last] [-l,--list]
[--provides] [--qf,--queryformat QUERYFMT]
[-R,--requires] [--scripts] [-s,--state]
[--triggers,--triggerscripts]
```

verify-options

```
[--nodeps] [--nofiles] [--noscripts]
[--nodigest] [--nosignature]
[--nolinkto] [--nofiledigest] [--nosize] [--nouser]
[--nogroup] [--nomtime] [--nomode] [--nordev]
[--nocaps]
```

install-options

```
[--aid] [--allfiles] [--badreloc] [--excludepath OLDPATH]
[--excludedocs] [--force] [-h,--hash]
[--ignoresize] [--ignorearch] [--ignoreos]
[--includedocs] [--justdb] [--nodeps]
[--nodigest] [--nosignature] [--nosuggest]
[--noorder] [--noscripts] [--notriggers]
[--oldpackage] [--percent] [--prefix NEWPATH]
[--relocate OLDPATH=NEWPATH]
[--replacefiles] [--replacepkgs]
[--test]
```

The `scp` command

SCP (secure copy) is a command-line utility that allows you to securely copy files and directories between two locations.

Both the files and passwords are encrypted so that anyone snooping on the traffic doesn't get anything sensitive.

Different ways to copy a file or directory:

- From local system to a remote system.
- From a remote system to a local system.
- Between two remote systems from the local system.

Examples:

1. To copy the files from a local system to a remote system:

```
scp /home/documents/local-file root@{remote-ip-address}:/home/
```

2. To copy the files from a remote system to the local system:

```
scp root@{remote-ip-address}:/home/remote-file
/home/documents/
```

3. To copy the files between two remote systems from the local system.

```
scp root@{remote1-ip-address}:/home/remote-file root@{remote2-
ip-address}/home/
```

4. To copy file though a jump host server.

```
scp /home/documents/local-file -oProxyJump=<jump-host-ip>
root@{remote-ip-address}/home/
```

On newer version of scp on some machines you can use the above command with a `-J` flag.

```
scp /home/documents/local-file -J <jump-host-ip> root@{remote-
ip-address}/home/
```

Syntax:

```
scp [OPTION] [user@]SRC_HOST:]file1 [user@]DEST_HOST:]file2
```

- `OPTION` - scp options such as cipher, ssh configuration, ssh port, limit, recursive copy ...etc.
- `[user@]SRC_HOST:]file1` - Source file
- `[user@]DEST_HOST:]file2` - Destination file

Local files should be specified using an absolute or relative path, while remote file names should include a user and host specification.

scp provides several that control every aspect of its behaviour. The most widely used options are:

Short Flag	Long Flag	Description
`-P`	-	Specifies the remote host ssh port.
`-p`	-	Preserves files modification and access times.
`-q`	-	Use this option if you want to suppress the progress meter and non-error messages.
`-C`	-	This option forces scp to compresses the data as it is sent to the destination machine.
`-r`	-	This option tells scp to copy directories recursively.

Before you begin

The `scp` command relies on `ssh` for data transfer, so it requires an `ssh key` or

`password` to authenticate on the remote systems.

The `colon (:)` is how scp distinguish between local and remote locations.

To be able to copy files, you must have at least read permissions on the source file and write permission on the target system.

Be careful when copying files that share the same name and location on both systems, `scp` will overwrite files without warning.

When transferring large files, it is recommended to run the scp command inside a `screen` or `tmux` session.

The `sleep` command

The `sleep` command is used to create a dummy job. A dummy job helps in delaying the execution. It takes time in seconds by default but a small suffix(s, m, h, d) can be added at the end to convert it into any other format. This command pauses the execution for an amount of time which is defined by NUMBER.

Note: If you will define more than one NUMBER with sleep command then this command will delay for the sum of the values.

Examples :

1. To sleep for 10s

```
sleep 10s
```

2. A more generalized command:

```
sleep NUMBER[SUFFIX]...
```

Options

It accepts the following options:

1. --help
   display this help and exit

2. --version
   output version information and exit

The `split` command

The `split` command in Linux is used to split a file into smaller files.

Examples

1. Split a file into a smaller file using file name

```
split filename.txt
```

2. Split a file named filename into segments of 200 lines beginning with prefix file

```
split -l 200 filename file
```

This will create files of the name fileaa, fileab, fileac, filead, etc. of 200 lines.

3. Split a file named filename into segments of 40 bytes with prefix file

```
split -b 40 filename file
```

This will create files of the name fileaa, fileab, fileac, filead, etc. of 40 bytes.

4. Split a file using --verbose to see the files being created.

```
split filename.txt --verbose
```

Syntax:

```
split [options] filename [prefix]
```

Additional Flags and their Functionalities

Short Flag	Long Flag	Description
-a	--suffix-length=N	Generate suffixes of length N (default 2)
	--additional-suffix=SUFFIX	Append an additional SUFFIX to file names
-b	--bytes=SIZE	Put SIZE bytes per output file
-C	--line-bytes=SIZE	Put at most SIZE bytes of records per output file
-d		Use numeric suffixes starting at 0, not alphabetic
	--numeric-suffixes[=FROM]	Same as -d, but allow setting the start value
-x		Use hex suffixes starting at 0, not alphabetic
	--hex-suffixes[=FROM]	Same as -x, but allow setting the start value
-e	--elide-empty-files	Do not generate empty output files with '-n'
	--filter=COMMAND	Write to shell COMMAND; file name is $FILE
-l	--lines=NUMBER	Put NUMBER lines/records per output file
-n	--number=CHUNKS	Generate CHUNKS output files; see explanation below
-t	--separator=SEP	Use SEP instead of newline as the record separator; '\0' (zero) specifies the NUL character
-u	--unbuffered	Immediately copy input to output with '-n r/...'
	--verbose	Print a diagnostic just before each output file is opened
	--help	Display this help and exit
	--version	Output version information and exit

The SIZE argument is an integer and optional unit (example: 10K is 10*1024). Units are K,M,G,T,P,E,Z,Y (powers of 1024) or KB,MB,... (powers of 1000).

CHUNKS may be:

CHUNKS	Description
`N`	Split into N files based on size of input
`K/N`	Output Kth of N to stdout
`l/N`	Split into N files without splitting lines/records
`l/K/N`	Output Kth of N to stdout without splitting lines/records
`r/N`	Like 'l' but use round robin distribution
`r/K/N`	Likewise but only output Kth of N to stdout

The `stat` command

The `stat` command lets you display file or file system status. It gives you useful information about the file (or directory) on which you use it.

Examples:

1. Basic command usage

```
stat file.txt
```

2. Use the `-c` (or `--format`) argument to only display information you want to see (here, the total size, in bytes)

```
stat file.txt -c %s
```

Syntax:

```
stat [OPTION] [FILE]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-L`	`--dereference`	Follow links
`-f`	`--file-system`	Display file system status instead of file status
`-c`	`--format=FORMAT`	Specify the format (see below)
`-t`	`--terse`	Print the information in terse form
-	`--cached=MODE`	Specify how to use cached attributes. Can be: `always`, `never`, or `default`

Short Flag	Long Flag	Description
-	`--printf=FORMAT`	Like `--format`, but interpret backslash escapes (`\n`, `\t`, ...)
-	`--help`	Display the help and exit
-	`--version`	Output version information and exit

Example of Valid Format Sequences for Files:

Format	Description
`%a`	Permission bits in octal
`%A`	Permission bits and file type in human readable form
`%d`	Device number in decimal
`%D`	Device number in hex
`%F`	File type
`%g`	Group ID of owner
`%G`	Group name of owner
`%h`	Number of hard links
`%i`	Inode number
`%m`	Mount point
`%n`	File name
`%N`	Quoted file name with dereference if symbolic link
`%s`	Total size, in bytes
`%u`	User ID of owner
`%U`	User name of owner
`%w`	Time of file birth, human-readable; - if unknown
`%x`	Time of last access, human-readable
`%y`	Time of last data modification, human-readable
`%z`	Time of last status change, human-readable

The `useradd` command

The `useradd` command is used to add or update user accounts to the system.

Examples:

To add a new user with the `useradd` command the syntax would be the following:

```
useradd NewUser
```

To add a new user with the `useradd` command and give a home directory path for this new user the syntax would be the following:

```
useradd -d /home/NewUser NewUser
```

To add a new user with the `useradd` command and give it a specific id the syntax would be the following:

```
useradd -u 1234 NewUser
```

Syntax:

```
useradd [OPTIONS] NameOfUser
```

Possible options:

Flag	Description	Params
-d	The new user will be created using /path/to/directory as the value for the user's login directory	/path/to/directory
-u	The numerical value of the user's ID	ID
-g	Create a user with specific group id	GroupID
-M	Create a user without home directory	-
-e	Create a user with expiry date	DATE (format: YYYY-MM-DD)
-c	Create a user with a comment	COMMENT
-s	Create a user with changed login shell	/path/to/shell
-p	Set an unencrypted password for the user	PASSWORD

The `userdel` command

The `userdel` command is used to delete a user account and related files

Examples:

To delete a user with the `userdel` command the syntax would be the following:

```
userdel userName
```

To force the removal of a user account even if the user is still logged in, using the `userdel` command the syntax would be the following:

```
userdel -f userName
```

To delete a user along with the files in the user's home directory using the `userdel` command the syntax would be the following:

```
userdel -r userName
```

Syntax:

```
userdel [OPTIONS] userName
```

Possible options:

Flag	Description
`-f`	Force the removal of the specified user account even if the user is logged in

Flag	Description
`-r`	Remove the files in the user’s home directory along with the home directory itself and the user’s mail spool
`-Z`	Remove any SELinux(Security-Enhanced Linux) user mapping for the user’s login.

The `usermod` command

The `usermod` command lets you change the properties of a user in Linux through the command line. After creating a user we sometimes have to change their attributes, like their password or login directory etc. So in order to do that we use the `usermod` command.

Syntax:

```
usermod [options] USER
```

Note : Only superuser (root) is allowed to execute `usermod` command

Options and their Functionalities:

Option	Description
`-a`	to add anyone of the group to a secondary group
`-c`	to add comment field for the useraccount
`-d`	to modify the directory for any existing user account
`-g`	change the primary group for a User
`-G`	to add supplementary groups
`-l`	to change existing user login name
`-L`	to lock system user account
`-m`	to move the contents of the home directory from existing home dir to new dir
`-p`	to create an un-encrypted password
`-s`	to create a specified shell for new accounts
`-u`	to assigned UID for the user account
`-U`	to unlock any locked user

Examples:

1. To add a comment/description for a user:

```
sudo usermod -c "This is test user" test_user
```

2. To change the home directory of a user:

```
sudo usermod -d /home/sam test_user
```

3. To change the expiry date of a user:

```
sudo usermod -e 2021-10-05 test_user
```

4. To change the group of a user:

```
sudo usermod -g sam test_user
```

5. To change user login name:

```
sudo usermod -l test_account test_user
```

6. To lock a user:

```
sudo usermod -L test_user
```

7. To unlock a user:

```
sudo usermod -U test_user
```

8. To set an unencrypted password for the user:

```
sudo usermod -p test_password test_user
```

9. To create a shell for the user:

```
sudo usermod -s /bin/sh test_user
```

10. To change the user id of a user:

```
sudo usermod -u 1234 test_user
```

The `ionice` command

The `ionice` command is used to set or get process I/O scheduling class and priority.

If no arguments are given , `ionice` will query the current I/O scheduling class and priority for that process.

Usage

```
ionice [options] -p <pid>
```

```
ionice [options] -P <pgid>
```

```
ionice [options] -u <uid>
```

```
ionice [options] <command>
```

A process can be of three scheduling classes:

- ### Idle

  A program with idle I/O priority will only get disk time when `no other program has asked for disk I/O for a defined grace period`.

  The impact of idle processes on normal system actively should be `zero`.

  This scheduling class `doesn't take priority` argument.

  Presently this scheduling class is permitted for an `ordinary user (since kernel 2.6.25)`.

- ### Best Effort

  This is `effective` scheduling class for any process that has `not asked for a specific I/O priority`.

  This class `takes priority argument from 0-7`, with `lower` number being `higher priority`.

  Programs running at the same best effort priority are served in `round-robbin fashion`.

  Note that before kernel 2.6.26 a process that has not asked for an I/O priority formally uses "None" as scheduling class , but the io schedular will treat such processes as if it were in the best effort class.

  The priority within best effort class will be dynamically derived form the CPU nice level of the process : io_priority = (cpu_nice + 20) / 5/ for kernels after 2.6.26 with CFQ I/O schedular a process that has not asked for sn io priority inherits CPU scheduling class.

  `The I/O priority is derived from the CPU nice level of the process` (smr sd before kernel 2.6.26).

- ## Real Time

  The real time schedular class is `given first access to disk, regardless of what else is going on in the system`.

  Thus the real time class needs to be used with some care, as it cans tarve other processes .

  As with the best effort class, `8 priority levels are defined denoting how big a time slice a given process will receive on each scheduling window`.

  This scheduling class is `not permitted for an ordinary user(non-root)`.

Options

Options	Description
-c, --class	name or number of scheduling class, 0: none, 1: realtime, 2: best-effort, 3: idle
-n, --classdata	priority (0..7) in the specified scheduling class,only for the realtime and best-effort classes
-p, --pid ...	act on these already running processes
-P, --pgid ...	act on already running processes in these groups
-t, --ignore	ignore failures
-u, --uid ...	act on already running processes owned by these users
-h, --help	display this help
-V, --version	display version

For more details see ionice(1).

Examples

Command	O/P	Explanation
`$ ionice`	*none: prio 4*	Running alone `ionice` will give the class and priority of current process
`$ ionice -p 101`	*none : prio 4*	Give the details(*class : priority*) of the process specified by given process id
`$ ionice -p 2`	*none: prio 4*	Check the class and priority of process with pid 2 it is none and 4 resp.
`$ ionice -c2 -n0 -p2`	2 (best-effort) priority 0 process 2	Now lets set process(pid) 2 as a best-effort program with highest priority
$ `ionice` -p 2	best-effort : prio 0	Now if I check details of Process 2 you can see the updated one
$ `ionice` /bin/ls		get priority and class info of bin/ls
$ `ionice` -n4 -p2		set priority 4 of process with pid 2
$ `ionice` -p 2	best-effort: prio 4	Now observe the difference between the command ran above and this one we have changed priority from 0 to 4
$ `ionice` -c0 -n4 -p2	ionice: ignoring given class data for none class	(Note that before kernel 2.6.26 a process that has not asked for an I/O priority formally uses “None” as scheduling class ,
		but the io schedular will treat such processes as if it were in the best effort class.)
		-t option : ignore failure
$ `ionice` -c0 -n4 -p2 -t		For ignoring the warning shown above we can use -t option so it will ignore failure

Conclusion

Thus we have successfully learnt about `ionice` command.

The `du` command

The `du` command, which is short for `disk usage` lets you retrieve information about disk space usage information in a specified directory. In order to customize the output according to the information you need, this command can be paired with the appropriate options or flags.

Examples:

1. To show the estimated size of sub-directories in the current directory:

```
du
```

2. To show the estimated size of sub-directories inside a specified directory:

```
du {PATH_TO_DIRECTORY}
```

Syntax:

```
du [OPTION]... [FILE]...
du [OPTION]... --files0-from=F
```

Additional Flags and their Functionalities:

Note: This does not include an exhaustive list of options.

Short Flag	Long Flag	Description
`-a`	`--all`	Includes information for both files and directories
`-c`	`--total`	Provides a grand total at the end of the list of files/directories

Short Flag	Long Flag	Description
`-d`	`--max-depth=N`	Provides information up to `N` levels from the directory where the command was executed
`-h`	`--human-readable`	Displays file size in human-readable units, not in bytes
`-s`	`--summarize`	Display only the total filesize instead of a list of files/directories

The `ping` command

The `ping` (Packet Internet Groper) command is used to check the network connectivity between host and server/host. This command takes as input the IP address or the URL and sends a data packet to the specified address with the message “PING” and get a response from the server/host this time is recorded which is called latency. Ping uses ICMP(Internet Control Message Protocol) to send an ICMP echo message to the specified host if that host is available then it sends ICMP reply message. Ping is generally measured in millisecond every modern operating system has this ping pre-installed.

The basic ping syntax includes ping followed by a hostname, a name of a website, or the exact IP address.

```
ping [option] [hostname] or [IP address]
```

Examples:

1. To get ping version installed on your system.

```
sudo ping -v
```

2. To check whether a remote host is up, in this case, google.com, type in your terminal:

```
ping google.com
```

3. Controlling the number of pings: Earlier we did not define the number of packets to send to the server/host by using -c option we can do so.

```
ping -c 5 google.com
```

4. Controlling the number of pings: Earlier a default sized packets were sent to a host but we can send light and heavy packet by using -s option.

```
ping -s 40 -c 5 google.com
```

5. Changing the time interval: By default ping wait for 1 sec to send next packet we can change this time by using -i option.

```
ping -i 2 google.com
```

The `rsync` command

The `rsync` command is probably one of the most used commands out there. It is used to securely copy files from one server to another over SSH.

Compared to the `scp` command, which does a similar thing, `rsync` makes the transfer a lot faster, and in case of an interruption, you could restore/resume the transfer process.

In this tutorial, I will show you how to use the `rsync` command and copy files from one server to another and also share a few useful tips!

Before you get started, you would need to have 2 Linux servers. I will be using DigitalOcean for the demo and deploy 2 Ubuntu servers.

You can use my referral link to get a free $100 credit that you could use to deploy your virtual machines and test the guide yourself on a few DigitalOcean servers:

DigitalOcean $100 Free Credit

Transfer Files from local server to remote

This is one of the most common causes. Essentially this is how you would copy the files from the server that you are currently on (the source server) to remote/destination server.

What you need to do is SSH to the server that is holding your files, cd to the directory that you would like to transfer over:

```
cd /var/www/html
```

And then run:

```
rsync -avz user@your-remote-server.com:/home/user/dir/
```

The above command would copy all the files and directories from the current folder on your server to your remote server.

Rundown of the command:

- `-a`: is used to specify that you want recursion and want to preserve the file permissions and etc.
- `-v`: is verbose mode, it increases the amount of information you are given during the transfer.
- `-z`: this option, rsync compresses the file data as it is sent to the destination machine, which reduces the amount of data being transmitted -- something that is useful over a slow connection.

I recommend having a look at the following website which explains the commands and the arguments very nicely:

https://explainshell.com/explain?cmd=rsync+-avz

In case that the SSH service on the remote server is not running on the standard `22` port, you could use `rsync` with a special SSH port:

```
rsync -avz -e 'ssh -p 1234' user@your-remote-
server.com:/home/user/dir/
```

Transfer Files remote server to local

In some cases you might want to transfer files from your remote server to your local server, in this case, you would need to use the following syntax:

```
rsync -avz your-user@your-remote-server.com:/home/user/dir/
/home/user/local-dir/
```

Again, in case that you have a non-standard SSH port, you can use the following command:

```
rsync -avz -e 'ssh -p 2510' your-user@your-remote-
server.com:/home/user/dir/ /home/user/local-dir/
```

Transfer only missing files

If you would like to transfer only the missing files you could use the `--ignore-existing` flag.

This is very useful for final sync in order to ensure that there are no missing files after a website or a server migration.

Basically the commands would be the same apart from the appended --ignore-existing flag:

```
 rsync -avz --ignore-existing  user@your-remote-
server.com:/home/user/dir/
```

Conclusion

Using `rsync` is a great way to quickly transfer some files from one machine over to another in a secure way over SSH.

For more cool Linux networking tools, I would recommend checking out this tutorial here:

Top 15 Linux Networking tools that you should know!

Hope that this helps!

Initially posted here: How to Transfer Files from One Linux Server to Another Using rsync

The `dig` command

dig - DNS lookup utility

The `dig` is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried.

Examples:

1. Dig is a network administration command-line tool for querying the Domain Name System.

```
dig google.com
```

2. The system will list all google.com DNS records that it finds, along with the IP addresses.

```
dig google.com ANY
```

Syntax:

```
dig [server] [name] [type] [q-type] [q-class] {q-opt}
            {global-d-opt} host [@local-server] {local-d-opt}
            [ host [@local-server] {local-d-opt} [...]]
```

Additional Flags and their Functionalities:

```
domain    is in the Domain Name System
        q-class  is one of (in,hs,ch,...) [default: in]
```

```
        q-type   is one of
(a,any,mx,ns,soa,hinfo,axfr,txt,...) [default:a]
                 (Use ixfr=version for type ixfr)
        q-opt    is one of:
                 -4                  (use IPv4 query transport
only)
                 -6                  (use IPv6 query transport
only)
                 -b address[#port]   (bind to source
address/port)
                 -c class            (specify query class)
                 -f filename         (batch mode)
                 -k keyfile          (specify tsig key file)
                 -m                  (enable memory usage
debugging)
                 -p port             (specify port number)
                 -q name             (specify query name)
                 -r                  (do not read ~/.digrc)
                 -t type             (specify query type)
                 -u                  (display times in usec
instead of msec)
                 -x dot-notation     (shortcut for reverse
lookups)
                 -y [hmac:]name:key  (specify named base64
tsig key)
        d-opt    is of the form +keyword[=value], where
keyword is:
                 +[no]aaflag         (Set AA flag in query
(+[no]aaflag))
                 +[no]aaonly         (Set AA flag in query
(+[no]aaflag))
                 +[no]additional     (Control display of
additional section)
                 +[no]adflag         (Set AD flag in query
(default on))
                 +[no]all            (Set or clear all display
flags)
                 +[no]answer         (Control display of
answer section)
                 +[no]authority      (Control display of
authority section)
                 +[no]badcookie      (Retry BADCOOKIE
responses)
                 +[no]besteffort     (Try to parse even
illegal messages)
                 +bufsize[=###]      (Set EDNS0 Max UDP packet
```

```
size)
                +[no]cdflag         (Set checking disabled
flag in query)
                +[no]class          (Control display of class
in records)
                +[no]cmd            (Control display of
command line -
                                     global option)
                +[no]comments       (Control display of
packet header
                                     and section name
comments)
                +[no]cookie         (Add a COOKIE option to
the request)
                +[no]crypto         (Control display of
cryptographic
                                     fields in records)
                +[no]defname        (Use search list
(+[no]search))
                +[no]dnssec         (Request DNSSEC records)
                +domain=###         (Set default domainname)
                +[no]dscp[=###]     (Set the DSCP value to
### [0..63])
                +[no]edns[=###]     (Set EDNS version) [0]
                +ednsflags=###      (Set EDNS flag bits)
                +[no]ednsnegotiation (Set EDNS version
negotiation)
                +ednsopt=###[:value] (Send specified EDNS
option)
                +noednsopt          (Clear list of +ednsopt
options)
                +[no]expandaaaa     (Expand AAAA records)
                +[no]expire         (Request time to expire)
                +[no]fail           (Don't try next server on
SERVFAIL)
                +[no]header-only    (Send query without a
question section)
                +[no]identify       (ID responders in short
answers)
                +[no]idnin          (Parse IDN names
[default=on on tty])
                +[no]idnout         (Convert IDN response
[default=on on tty])
                +[no]ignore         (Don't revert to TCP for
TC responses.)
                +[no]keepalive      (Request EDNS TCP
```

```
keepalive)
                  +[no]keepopen       (Keep the TCP socket open
between queries)
                  +[no]mapped         (Allow mapped IPv4 over
IPv6)
                  +[no]multiline      (Print records in an
expanded format)
                  +ndots=###          (Set search NDOTS value)
                  +[no]nsid           (Request Name Server ID)
                  +[no]nssearch       (Search all authoritative
nameservers)
                  +[no]onesoa         (AXFR prints only one soa
record)
                  +[no]opcode=###     (Set the opcode of the
request)
                  +padding=###        (Set padding block size
[0])
                  +[no]qr             (Print question before
sending)
                  +[no]question       (Control display of
question section)
                  +[no]raflag         (Set RA flag in query
(+[no]raflag))
                  +[no]rdflag         (Recursive mode
(+[no]recurse))
                  +[no]recurse        (Recursive mode
(+[no]rdflag))
                  +retry=###          (Set number of UDP
retries) [2]
                  +[no]rrcomments     (Control display of per-
record comments)
                  +[no]search         (Set whether to use
searchlist)
                  +[no]short          (Display nothing except
short
                                       form of answers - global
option)
                  +[no]showsearch     (Search with intermediate
results)
                  +[no]split=##       (Split hex/base64 fields
into chunks)
                  +[no]stats          (Control display of
statistics)
                  +subnet=addr        (Set edns-client-subnet
option)
                  +[no]tcflag         (Set TC flag in query
```

```
(+[no]tcflag))
                +[no]tcp            (TCP mode (+[no]vc))
                +timeout=###        (Set query timeout) [5]
                +[no]trace          (Trace delegation down
from root [+dnssec])
                +tries=###          (Set number of UDP
attempts) [3]
                +[no]ttlid          (Control display of ttls
in records)
                +[no]ttlunits       (Display TTLs in human-
readable units)
                +[no]unexpected     (Print replies from
unexpected sources
                                     default=off)
                +[no]unknownformat  (Print RDATA in RFC 3597
"unknown" format)
                +[no]vc             (TCP mode (+[no]tcp))
                +[no]yaml           (Present the results as
YAML)
                +[no]zflag          (Set Z flag in query)
        global d-opts and servers (before host name) affect
all queries.
        local d-opts and servers (after host name) affect only
that lookup.
        -h                          (print help and exit)
        -v                          (print version and exit)
```

The `whois` command

The `whois` command in Linux to find out information about a domain, such as the owner of the domain, the owner's contact information, and the nameservers that the domain is using.

Examples:

1. Performs a whois query for the domain name:

```
whois {Domain_name}
```

2. -H option omits the lengthy legal disclaimers that many domain registries deliver along with the domain information.

```
whois -H {Domain_name}
```

Syntax:

```
whois [ -h HOST ] [ -p PORT ] [ -aCFHlLMmrRSVx ] [ -g
SOURCE:FIRST-LAST ]
      [ -i ATTR ] [ -S SOURCE ] [ -T TYPE ] object
```

```
whois -t TYPE
```

```
whois -v TYPE
```

```
whois -q keyword
```

Additional Flags and their Functionalities:

Flag	Description
`-h HOST`, `--host HOST`	Connect to HOST.
`-H`	Do not display the legal disclaimers some registries like to show you.
`-p`, `--port PORT`	Connect to PORT.
`--verbose`	Be verbose.
`--help`	Display online help.
`--version`	Display client version information. Other options are flags understood by whois.ripe.net and some other RIPE-like servers.
`-a`	Also search all the mirrored databases.
`-b`	Return brief IP address ranges with abuse contact.
`-B`	Disable object filtering *(show the e-mail addresses)*
`-c`	Return the smallest IP address range with a reference to an irt object.
`-d`	Return the reverse DNS delegation object too.
`-g SOURCE:FIRST-LAST`	Search updates from SOURCE database between FIRST and LAST update serial number. It's useful to obtain Near Real Time Mirroring stream.
`-G`	Disable grouping of associated objects.
`-i ATTR[,ATTR]...`	Search objects having associated attributes. ATTR is attribute name. Attribute value is positional OBJECT argument.
`-K`	Return primary key attributes only. Exception is members attribute of set object which is always returned. Another exceptions are all attributes of objects organisation, person, and role that are never returned.
`-l`	Return the one level less specific object.
`-L`	Return all levels of less specific objects.
`-m`	Return all one level more specific objects.
`-M`	Return all levels of more specific objects.
`-q KEYWORD`	Return list of keywords supported by server. KEYWORD can be version for server version, sources for list of source databases, or types for object types.
`-r`	Disable recursive look-up for contact information.

Flag	Description
`-R`	Disable following referrals and force showing the object from the local copy in the server.
`-s SOURCE[,SOURCE]...`	Request the server to search for objects mirrored from SOURCES. Sources are delimited by comma and the order is significant. Use `-q` sources option to obtain list of valid sources.
`-t TYPE`	Return the template for a object of TYPE.
`-T TYPE[,TYPE]...`	Restrict the search to objects of TYPE. Multiple types are separated by a comma.
`-v TYPE`	Return the verbose template for a object of TYPE.
`-x`	Search for only exact match on network address prefix.

The `ssh` command

The `ssh` command in Linux stands for "Secure Shell". It is a protocol used to securely connect to a remote server/system. ssh is more secure in the sense that it transfers the data in encrypted form between the host and the client. ssh runs at TCP/IP port 22.

Examples:

1. Use a Different Port Number for SSH Connection:

```
ssh test.server.com -p 3322
```

2. -i ssh to remote server using a private key?

```
ssh -i private.key user_name@host
```

3. -l ssh specifying a different username

```
ssh -l alternative-username sample.ssh.com
```

Syntax:

```
ssh user_name@host(IP/Domain_Name)
```

```
ssh -i private.key user_name@host
```

```
ssh sample.ssh.com  ls /tmp/doc
```

Additional Flags and their Functionalities:

Flag	Description
`-1`	Forces ssh to use protocol SSH-1 only.
`-2`	Forces ssh to use protocol SSH-2 only.
`-4`	Allows IPv4 addresses only.
`-A`	Authentication agent connection forwarding is enabled..
`-a`	Authentication agent connection forwarding is disabled.
`-B bind_interface`	Bind to the address of bind_interface before attempting to connect to the destination host. This is only useful on systems with more than one address.
`-b bind_address`	Use bind_address on the local machine as the source address of the connection. Only useful on systems with more than one address.
`-C`	Compresses all data (including stdin, stdout, stderr, and data for forwarded X11 and TCP connections) for a faster transfer of data.
`-c cipher_spec`	Selects the cipher specification for encrypting the session.
`-D [bind_address:]port`	Dynamic application-level port forwarding. This allocates a socket to listen to port on the local side. When a connection is made to this port, the connection is forwarded over the secure channel, and the application protocol is then used to determine where to connect to from the remote machine.
`-E log_file`	Append debug logs instead of standard error.

Flag	Description
`-e escape_char`	Sets the escape character for sessions with a pty (default: ‘~’). The escape character is only recognized at the beginning of a line. The escape character followed by a dot (‘.’) closes the connection; followed by control-Z suspends the connection; and followed by itself sends the escape character once. Setting the character to “none” disables any escapes and makes the session fully transparent.
`-F configfile`	Specifies a per-user configuration file. The default for the per-user configuration file is ~/.ssh/config.
`-f`	Requests ssh to go to background just before command execution.
`-G`	Causes ssh to print its configuration after evaluating Host and Match blocks and exit.
`-g`	Allows remote hosts to connect to local forwarded ports.
`-I pkcs11`	Specify the PKCS#11 shared library ssh should use to communicate with a PKCS#11 token providing keys.
`-i identity_file`	A file from which the identity key (private key) for public key authentication is read.
`-J [user@]host[:port]`	Connect to the target host by first making a ssh connection to the pjump host[(/iam/jump-host) and then establishing a TCP forwarding to the ultimate destination from there.
`-K`	Enables GSSAPI-based authentication and forwarding (delegation) of GSSAPI credentials to the server.
`-k`	Disables forwarding (delegation) of GSSAPI credentials to the server.

Flag	Description
`-L [bind_address:]port:host:hostport`, `-L [bind_address:]port:remote_socket`, `-L local_socket:host:hostport`, `-L local_socket:remote_socket`	Specifies that connections to the given TCP port or Unix socket on the local (client) host are to be forwarded to the given host and port, or Unix socket, on the remote side. This works by allocating a socket to listen to either a TCP port on the local side, optionally bound to the specified bind_address, or to a Unix socket. Whenever a connection is made to the local port or socket, the connection is forwarded over the secure channel, and a connection is made to either host port hostport, or the Unix socket remote_socket, from the remote machine.
`-l login_name`	Specifies the user to log in as on the remote machine.
`-M`	Places the ssh client into “master” mode for connection sharing. Multiple -M options places ssh into “master” mode but with confirmation required using ssh-askpass before each operation that changes the multiplexing state (e.g. opening a new session).
`-m mac_spec`	A comma-separated list of MAC (message authentication code) algorithms, specified in order of preference.
`-N`	Do not execute a remote command. This is useful for just forwarding ports.
`-n`	Prevents reading from stdin.

Flag	Description
`-O ctl_cmd`	Control an active connection multiplexing master process. When the -O option is specified, the ctl_cmd argument is interpreted and passed to the master process. Valid commands are: “check” (check that the master process is running), “forward” (request forwardings without command execution), “cancel” (cancel forwardings), “exit” (request the master to exit), and “stop” (request the master to stop accepting further multiplexing requests).
`-o`	Can be used to give options in the format used in the configuration file. This is useful for specifying options for which there is no separate command-line flag.
`-p`, `--port PORT`	Port to connect to on the remote host.
`-Q query_option`	Queries ssh for the algorithms supported for the specified version 2. The available features are: cipher (supported symmetric ciphers), cipher-auth (supported symmetric ciphers that support authenticated encryption), help (supported query terms for use with the -Q flag), mac (supported message integrity codes), kex (key exchange algorithms), kex-gss (GSSAPI key exchange algorithms), key (keytypes), key-cert (certificate key types), key-plain (non-certificate key types), key-sig (all keytypes and signature algorithms), protocol-version (supported SSH protocol versions), and sig (supported signature algorithms). Alternatively, any keyword from ssh_config(5) or sshd_config(5) thattakes an algorithm list may be used as an alias for the corresponding query_option.
`-q`	Qiet mode. Causes most warning and diagnostic messages to be suppressed.

Flag	Description
`-R [bind_address:]port:host:hostport, -R [bind_address:]port:local_socket, -R remote_socket:host:hostport, -R remote_socket:local_socket, -R [bind_address:]port`	Specifies that connections to the given TCP port or Unix socket on the remote (server) host are to be forwarded to the local side.
`-S ctl_path`	Specifies the location of a control socket for connection sharing, or the string “none” to disable connection sharing.
`-s`	May be used to request invocation of a subsystem on the remote system. Subsystems facilitate the use of SSH as a secure transport for other applications (e.g. sftp(1)). The subsystem is specified as the remote command.
`-T`	Disable pseudo-terminal allocation.
`-t`	Force pseudo-terminal allocation. This can be used to execute arbitrary screen-based programs on a remote machine, which can be very useful, e.g. when implementing menu services. Multiple -t options force tty allocation, even if ssh has no local tty.
`-V`	Display the version number.
`-v`	Verbose mode. It echoes everything it is doing while establishing a connection. It is very useful in the debugging of connection failures.
`-W host:port`	Requests that standard input and output on the client be forwarded to host on port over the secure channel. Implies -N, -T, ExitOnForwardFailure and ClearAllForwardings, though these can be overridden in the configuration file or using -o command line options.

Flag	Description
`-w local_tun[remote_tun]`	Requests tunnel device forwarding with the specified tun devices between the client (local_tun) and the server (remote_tun). The devices may be specified by numerical ID or the keyword “any”, which uses the next available tunnel device. If remote_tun is not specified, it defaults to “any”. If the Tunnel directive is unset, it will be set to the default tunnel mode, which is “point-to-point”. If a different Tunnel forwarding mode it desired, then it should be specified before -w.
`-X`	Enables X11 forwarding (GUI Forwarding).
`-x`	Disables X11 forwarding (GUI Forwarding).
`-Y`	Enables trusted X11 Forwarding.
`-y`	Send log information using the syslog system module. By default this information is sent to stderr.

The `awk` command

Awk is a general-purpose scripting language designed for advanced text processing. It is mostly used as a reporting and analysis tool.

WHAT CAN WE DO WITH AWK?

1. AWK Operations: (a) Scans a file line by line (b) Splits each input line into fields (c) Compares input line/fields to pattern (d) Performs action(s) on matched lines

2. Useful For: (a) Transform data files (b) Produce formatted reports

3. Programming Constructs: (a) Format output lines (b) Arithmetic and string operations (c) Conditionals and loops

Syntax

```
awk options 'selection _criteria {action }' input-file > output-file
```

Example

Consider the following text file as the input file for below example:

```
```
$cat > employee.txt
```
```
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
```

1. Default behavior of Awk: By default Awk prints every line of data from the specified file.

```
$ awk '{print}' employee.txt
```

```
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
```

In the above example, no pattern is given. So the actions are applicable to all the lines. Action print without any argument prints the whole line by default, so it prints all the lines of the file without failure.

2. Print the lines which match the given pattern.

```
awk '/manager/ {print}' employee.txt
```

```
ajay manager account 45000
varun manager sales 50000
amit manager account 47000
```

In the above example, the awk command prints all the line which matches with the ‘manager’.
```

3. Splitting a Line Into Fields : For each record i.e line, the awk command splits the record delimited by whitespace character by default and stores it in the $n variables. If the line has 4 words, it will be stored in $1, $2, $3 and $4 respectively. Also, $0 represents the whole line.

```
$ awk '{print $1,$4}' employee.txt
```

```
ajay 45000
sunil 25000
varun 50000
amit 47000
tarun 15000
```

Built-In Variables In Awk

Awk's built-in variables include the field variables—$1, $2, $3, and so on ($0 is the entire line) — that break a line of text into individual words or pieces called fields.

NR: NR command keeps a current count of the number of input records. Remember that records are usually lines. Awk command performs the pattern/action statements once for each record in a file. NF: NF command keeps a count of the number of fields within the current input record. FS: FS command contains the field separator character which is used to divide fields on the input line. The default is "white space", meaning space and tab characters. FS can be reassigned to another character (typically in BEGIN) to change the field separator. RS: RS command stores the current record separator character. Since, by default, an input line is the input record, the default record separator character is a newline. OFS: OFS command stores the output field separator, which separates the fields when Awk prints them. The default is a blank space. Whenever print has several parameters separated with commas, it will print the value of OFS in between each parameter. ORS: ORS command stores the output record separator, which separates the output lines when Awk prints them. The default is a newline character. print automatically outputs the contents of ORS at the end of whatever it is given to print.

The `crontab` command

`crontab` is used to maintain crontab files for individual users (Vixie Cron)

crontab is the program used to install, uninstall or list the tables used to drive the cron(8) daemon in Vixie Cron. Each user can have their own crontab, and though these are files in `/var/spool/cron/crontabs`, they are not intended to be edited directly.

Syntax:

```
crontab [ -u user ] file
crontab [ -u user ] [ -i ] { -e | -l | -r }
```

Examples:

1. The -l option causes the current crontab to be displayed on standard output.

```
crontab -l
```

2. The -r option causes the current crontab to be removed.

```
crontab -r
```

3. The -e option is used to edit the current crontab using the editor specified by the VISUAL or EDITOR environment variables. After you exit from the editor, the modified crontab will be installed automatically. If neither of the environment variables is defined, then the default editor /usr/bin/editor is used.

```
crontab -e
```

4. You can specify the user you want to edit the crontab for. Every user has its own

crontab. Assume you have a `www-data` user, which is in fact the user Apache is default running as. If you want to edit the crontab for this user you can run the following command

```
crontab -u www-data -e
```

Help Command

Run below command to view the complete guide to `crontab` command.

```
man crontab
```

The `xargs` command

`xargs` is used to build and execute command lines from standard input

Some commands like grep can accept input as parameters, but some commands accepts arguments, this is place where xargs came into picture.

Syntax:

```
xargs [options] [command [initial-arguments]]
```

Options:

```
-0, --null
```

Input items are terminated by a null character instead of by whitespace, and the quotes and backslash are not special (every character is taken literal-ly). Disables the end of file string, which is treated like any other argument. Useful when input items might contain white space, quote marks, or back-slashes.

```
-a file, --arg-file=file
```

Read items from file instead of standard input. If you use this option, stdin remains unchanged when commands are run. Otherwise, stdin is redirected from /dev/null.

```
-o, --open-tty
```

Reopen stdin as /dev/tty in the child process before executing the command. This is useful if you want xargs to run an interactive application.

```
--delimiter=delim, -d delim
```

Input items are terminated by the specified character. The specified delimiter may be a single character, a C-style character escape such as \n, or an octal or hexadecimal escape code. Octal and hexadecimal escape codes are understood as for the printf command. Multibyte characters are not supported. When processing the input, quotes and backslash are not special; every character in the input is taken literally. The -d option disables any end-of-file string, which is treated like any other argument. You can use this option when the input consists of simply newline-separated items, although it is al- most always better to design your program to use --null where this is possible.

```
-p, --interactive
```

Prompt the user about whether to run each command line and read a line from the terminal. Only run the command line if the response starts with `y' or` Y'. Implies -t.

Examples:

```
find /tmp -name core -type f -print | xargs /bin/rm -f
```

Find files named core in or below the directory /tmp and delete them. Note that this will work incorrectly if there are any filenames containing newlines or spaces.

```
find /tmp -name core -type f -print0 | xargs -0 /bin/rm -f
```

Find files named core in or below the directory /tmp and delete them, processing filenames in such a way that file or directory names containing spaces or new- lines are correctly handled.

```
find /tmp -depth -name core -type f -delete
```

Find files named core in or below the directory /tmp and delete them, but more efficiently than in the previous example (because we avoid the need to use fork(2) and exec(2) to launch rm and we don't need the extra xargs process).

```
cut -d: -f1 < /etc/passwd | sort | xargs echo
```

Generates a compact listing of all the users on the system.

Help Command

Run below command to view the complete guide to `xargs` command.

```
man xargs
```

The `nohup` command

When a shell exits (maybe while logging out of an SSH session), the HUP ('hang up') signal is send to all of its child processes, causing them to terminate. If you require a long-running process to continue after exiting shell, you'll need the `nohup` command. Prefixing any command with `nohup` causes the command to become *immune* to HUP signals. Additionally, STDIN is being ignored and all output gets redirected to local file `./nohup.out`.

Examples:

1. Applying nohup to a long-running debian upgrade:

```
nohup apt-get -y upgrade
```

Syntax:

```
nohup COMMAND [ARG]...
nohup OPTION
```

The `pstree` command

The `pstree` command is similar to `ps`, but instead of listing the running processes, it shows them as a tree. The tree-like format is sometimes more suitable way to display the processes hierarchy which is a much simpler way to visualize running processes. The root of the tree is either init or the process with the given pid.

Examples

1. To display a hierarchical tree structure of all running processes:

```
pstree
```

2. To display a tree with the given process as the root of the tree:

```
pstree [pid]
```

3. To show only those processes that have been started by a user:

```
pstree [USER]
```

4. To show the parent processes of the given process:

```
pstree -s [PID]
```

5. To view the output one page at a time, pipe it to the `less` command:

```
pstree | less
```

Syntax

```
ps [OPTIONS] [USER or PID]
```

Additional Flags and their Functionalities

Short Flag	Long Flag	Description
`-a`	`--arguments`	Show command line arguments
`-A`	`--ascii`	use ASCII line drawing characters
`-c`	`--compact`	Don't compact identical subtrees
`-h`	`--highlight-all`	Highlight current process and its ancestors
`-H PID`	`--highlight-pid=PID`	highlight this process and its ancestors
`-g`	`--show-pgids`	show process group ids; implies `-c`
`-G`	`--vt100`	use VT100 line drawing characters
`-l`	`--long`	Don't truncate long lines
`-n`	`--numeric-sort`	Sort output by PID
`-N type`	`--ns-sort=type`	Sort by namespace type (cgroup, ipc, mnt, net, pid, user, uts)
`-p`	`--show-pids`	show PIDs; implies -c
`-s`	`--show-parents`	Show parents of the selected process
`-S`	`--ns-changes`	show namespace transitions
`-t`	`--thread-names`	Show full thread names
`-T`	`--hide-threads`	Hide threads, show only processes
`-u`	`--uid-changes`	Show uid transitions
`-U`	`--unicode`	Use UTF-8 (Unicode) line drawing characters
`-V`	`--version`	Display version information
`-Z`	`--security-context`	Show SELinux security contexts

The `tree` command

The `tree` command in Linux recursively lists directories as tree structures. Each listing is indented according to its depth relative to root of the tree.

Examples:

1. Show a tree representation of the current directory.

```
tree
```

2. -L NUMBER limits the depth of recursion to avoid display very deep trees.

```
tree -L 2 /
```

Syntax:

```
tree  [-acdfghilnpqrstuvxACDFQNSUX]  [-L  level [-R]] [-H
baseHREF] [-T title]
      [-o filename] [--nolinks] [-P pattern] [-I  pattern]  [-
-inodes]
      [--device] [--noreport] [--dirsfirst] [--version] [--
help] [--filelimit #]
      [--si] [--prune] [--du] [--timefmt  format]  [--
matchdirs]  [--from-file]
      [--] [directory ...]
```

Additional Flags and their Functionalities:

Flag	Description
`-a`	Print all files, including hidden ones.

Flag	Description
`-d`	Only list directories.
`-l`	Follow symbolic links into directories.
`-f`	Print the full path to each listing, not just its basename.
`-x`	Do not move across file-systems.
`-L #`	Limit recursion depth to #.
`-P REGEX`	Recurse, but only list files that match the REGEX.
`-I REGEX`	Recurse, but do not list files that match the REGEX.
`--ignore-case`	Ignore case while pattern-matching.
`--prune`	Prune empty directories from output.
`--filelimit #`	Omit directories that contain more than # files.
`-o FILE`	Redirect STDOUT output to FILE.
`-i`	Do not output indentation.

The `whereis` command

The `whereis` command is used to find the location of source/binary file of a command and manuals sections for a specified file in Linux system. If we compare `whereis` command with find command they will appear similar to each other as both can be used for the same purposes but `whereis` command produces the result more accurately by consuming less time comparatively.

Points to be kept on mind while using the whereis command:

Since the `whereis` command uses chdir(change directory 2V) to give you the result in the fastest possible way, the pathnames given with the -M, -S, or -B must be full and well-defined i.e. they must begin with a `/` and should be a valid path that exist in the system's directories, else it exits without any valid result. `whereis` command has a hard-coded(code which is not dynamic and changes with specification) path, so you may not always find what you're looking for.

Syntax

```
whereis [options] [filename]
```

Options

-b : This option is used when we only want to search for binaries. -m : This option is used when we only want to search for manual sections. -s : This option is used when we only want to search for source files. -u: This option search for unusual entries. A source file or a binary file is said to be unusual if it does not have any existence in system as per [-bmsu] described along with "–u". Thus `whereis -m -u *' asks for those files in the current directory which have unsual entries.

-B : This option is used to change or otherwise limit the places where whereis searches for binaries. -M : This option is used to change or otherwise limit the places where whereis searches for manual sections. -S : This option is used to change or otherwise limit the places where whereis searches for source files.

-f : This option simply terminate the last directory list and signals the start of file names. This must be used when any of the -B, -M, or -S options are used. -V: Displays

version information and exit. -h: Displays the help and exit.

The `printf` command

This command lets you print the value of a variable by formatting it using rules. It is pretty similar to the printf in C language.

Syntax:

```
$printf [-v variable_name] format [arguments]
```

Options:

OPTION	Description
`FORMAT`	FORMAT controls the output, and defines the way that the ARGUMENTs will be expressed in the output
`ARGUMENT`	An ARGUMENT will be inserted into the formatted output according to the definition of FORMAT
`--help`	Display help and exit
`--version`	Output version information adn exit

Formats:

The anatomy of the FORMAT string can be extracted into three different parts,

- *ordinary characters*, which are copied exactly the same characters as were used originally to the output.
- *interpreted character* sequences, which are escaped with a backslash ("\").
- *conversion specifications*, this one will define the way the ARGUMENTs will be expressed as part of the output.

You can see those parts in this example,

```
printf " %s is where over %d million developers shape \"the
future of sofware.\" " Github 65
```

The output:

```
Github is where over 65 million developers shape "the future
of sofware."
```

There are two conversion specifications `%s` and `%d`, and there are two escaped characters which are the opening and closing double-quotes wrapping the words of *the future of software*. Other than that are the ordinary characters.

Conversion Specifications:

Each conversion specification begins with a `%` and ends with a `conversion character`. Between the `%` and the `conversion character` there may be, in order:

`-`	A minus sign. This tells printf to left-adjust the conversion of the argument
number	An integer that specifies field width; printf prints a conversion of ARGUMENT in a field at least number characters wide. If necessary it will be padded on the left (or right, if left-adjustment is called for) to make up the field width
`.`	A period, which separates the field width from the precision
number	An integer, the precision, which specifies the maximum number of characters to be printed from a string, or the number of digits after the decimal point of a floating-point value, or the minimum number of digits for an integer
`h` or `l`	These differentiate between a short and a long integer, respectively, and are generally only needed for computer programming

The conversion characters tell `printf` what kind of argument to print out, are as follows:

Conversion char	Argument type
`s`	A string
`c`	An integer, expressed as a character corresponds ASCII code
`d, i`	An integer as a decimal number
`o`	An integer as an unsigned octal number
`x, X`	An integer as an unsigned hexadecimal number
`u`	An integer as an unsigned decimal number

Conversion char	Argument type
`f`	A floating-point number with a default precision of 6
`e, E`	A floating-point number in scientific notation
`p`	A memory address pointer
`%`	No conversion

Here is the list of some examples of the `printf` output the ARGUMENT. we can put any word but in this one we put a 'linuxcommand` word and enclosed it with quotes so we can see easier the position related to the whitespaces.

FORMAT string	ARGUMENT string	Output string
`"%s"`	`"linuxcommand"`	"linuxcommand"
`"%5s"`	`"linuxcommand"`	"linuxcommand"
`"%.5s"`	`"linuxcommand"`	"linux"
`"%-8s"`	`"linuxcommand"`	"linuxcommand"
`"%-15s"`	`"linuxcommand"`	"linuxcommand "
`"%12.5s"`	`"linuxcommand"`	" linux"
`"%-12.5"`	`"linuxcommand"`	"linux "
`"%-12.4"`	`"linuxcommand"`	"linu "

Notes:

- `printf` requires the number of conversion strings to match the number of ARGUMENTs
- `printf` maps the conversion strings one-to-one, and expects to find exactly one ARGUMENT for each conversion string
- Conversion strings are always interpreted from left to right.

Here's the example:

The input

```
printf "We know %f is %s %d" 12.07 "larger than" 12
```

The output:

```
We know 12.070000 is larger than 12
```

The example above shows 3 arguments, *12.07*, *larger than*, and *12*. Each of them

interpreted from left to right one-to-one with the given 3 conversion strings (`%f`, `%d`, `%s`).

Character sequences which are interpreted as special characters by `printf`:

Escaped char	Description
`\a`	issues an alert (plays a bell). Usually ASCII BEL characters
`\b`	prints a backspace
`\c`	instructs `printf` to produce no further output
`\e`	prints an escape character (ASCII code 27)
`\f`	prints a form feed
`\n`	prints a newline
`\r`	prints a carriage return
`\t`	prints a horizontal tab
`\v`	prints a vertical tab
`\"`	prints a double-quote (")
`\\`	prints a backslash ()
`\NNN`	prints a byte with octal value `NNN` (1 to 3 digits)
`\xHH`	prints a byte with hexadecimal value `HH` (1 to 2 digits)
`\uHHHH`	prints the unicode character with hexadecimal value `HHHH` (4 digits)
`\UHHHHHHHH`	prints the unicode character with hexadecimal value `HHHHHHHH` (8 digits)
`%b`	prints ARGUMENT as a string with "\" escapes interpreted as listed above, with the exception that octal escapes take the form `\0` or `\0NN`

Examples:

The format specifiers usually used with printf are stated in the examples below:

- %s

```
$printf "%s\n" "Printf command documentation!"
```

This will print `Printf command documentation!` in the shell.

Other important attributes of printf command:

- `%b` - Prints arguments by expanding backslash escape sequences.

- `%q` - Prints arguments in a shell-quoted format which is reusable as input.
- `%d` , `%i` - Prints arguments in the format of signed decimal integers.
- `%u` - Prints arguments in the format of unsigned decimal integers.
- `%o` - Prints arguments in the format of unsigned octal(base 8) integers.
- `%x`, `%X` - Prints arguments in the format of unsigned hexadecimal(base 16) integers. %x prints lower-case letters and %X prints upper-case letters.
- `%e`, `%E` - Prints arguments in the format of floating-point numbers in exponential notation. %e prints lower-case letters and %E prints upper-case.
- `%a`, `%A` - Prints arguments in the format of floating-point numbers in hexadecimal(base 16) fractional notation. %a prints lower-case letters and %A prints upper-case.
- `%g`, `%G` - Prints arguments in the format of floating-point numbers in normal or exponential notation, whichever is more appropriate for the given value and precision. %g prints lower-case letters and %G prints upper-case.
- `%c` - Prints arguments as single characters.
- `%f` - Prints arguments as floating-point numbers.
- `%s` - Prints arguments as strings.
- `%%` - Prints a "%" symbol.

More Examples:

The input:

```
printf 'Hello\nyoung\nman!'
```

The output:

```
hello
young
man!
```

The two `\n` break the sentence into 3 parts of words.

The input:

```
printf "%f\n" 2.5 5.75
```

The output

```
2.500000
5.750000
```

The `%f` specifier combined with the `\n` interpreted the two arguments in the form of floating point in the seperated new lines.

The `cut` command

The `cut` command lets you remove sections from each line of files. Print selected parts of lines from each FILE to standard output. With no FILE, or when FILE is -, read standard input.

Usage and Examples:

1. Selecting specific fields in a file

```
cut -d "delimiter" -f (field number) file.txt
```

2. Selecting specific characters:

```
cut -c [(k)-(n)/(k),(n)/(n)] filename
```

Here, **k** denotes the starting position of the character and **n** denotes the ending position of the character in each line, if *k* and *n* are separated by "-" otherwise they are only the position of character in each line from the file taken as an input.

3. Selecting specific bytes:

```
cut -b 1,2,3 filename                    //select bytes 1,2
and 3
cut -b 1-4 filename                           //select bytes 1
through 4
cut -b 1- filename                             //select bytes
1 through the end of file
cut -b -4 filename                             //select bytes
from the beginning till the 4th byte
```

Tabs and backspaces are treated like as a character of 1 byte.

Syntax:

```
cut OPTION... [FILE]...
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
-b	--bytes=LIST	select only these bytes
-c	--characters=LIST	select only these characters
-d	--delimiter=DELIM	use DELIM instead of TAB for field delimiter
-f	--fields	select only these fields; also print any line that contains no delimiter character, unless the -s option is specified
-s	--only-delimited	do not print lines not containing delimiters
-z	--zero-terminated	line delimiter is NUL, not newline

The `sed` command

`sed` command stands for stream editor. A stream editor is used to perform basic text transformations on an input stream (a file or input from a pipeline). For instance, it can perform lot's of functions on files like searching, find and replace, insertion or deletion. While in some ways it is similar to an editor which permits scripted edits (such as `ed`), `sed` works by making only one pass over the input(s), and is consequently more efficient. But it is sed's ability to filter text in a pipeline that particularly distinguishes it from other types of editors.

The most common use of `sed` command is for a substitution or for find and replace. By using sed you can edit files even without opening it, which is a much quicker way to find and replace something in the file. It supports basic and extended regular expressions that allow you to match complex patterns. Most Linux distributions come with GNU and `sed` is pre-installed by default.

Examples:

1. To Find and Replace String with `sed`

```
sed -i 's/{search_regex}/{replace_value}/g' input-file
```

2. For Recursive Find and Replace *(along with `find`)*

Sometimes you may want to recursively search directories for files containing a string and replace the string in all files. This can be done using commands such as find to recursively find files in the directory and piping the file names to `sed`. The following command will recursively search for files in the current working directory and pass the file names to `sed`. It will recursively search for files in the current working directory and pass the file names to `sed`.

```
find . -type f -exec sed -i
's/{search_regex}/{replace_value}/g' {} +
```

Syntax:

```
sed [OPTION]... {script-only-if-no-other-script} [INPUT-
FILE]...
```

- `OPTION` - sed options in-place, silent, follow-symlinks, line-length, null-data ...etc.
- `{script-only-if-no-other-script}` - Add the script to command if available.
- `INPUT-FILE` - Input Stream, A file or input from a pipeline.

If no option is given, then the first non-option argument is taken as the sed script to interpret. All remaining arguments are names of input files; if no input files are specified, then the standard input is read.

GNU sed home page: http://www.gnu.org/software/sed/

Short Flag	Long Flag	Description
`-i[SUFFIX]`	--in-place[=SUFFIX]	Edit files in place (makes backup if SUFFIX supplied).
`-n`	--quiet, --silent	Suppress automatic printing of pattern space.
`-e script`	--expression=script	Add the script to the commands to be executed.
`-f script-file`	--file=script-file	Add the contents of script-file to the commands to be executed.
`-l N`	--line-length=N	Specify the desired line-wrap length for the `l` command.
`-r`	--regexp-extended	Use extended regular expressions in the script.
`-s`	--separate	Consider files as separate rather than as a single continuous long stream.
`-u`	--unbuffered	Load minimal amounts of data from the input files and flush the output buffers more often.
`-z`	--null-data	Separate lines by NULL characters.

Before you begin

It may seem complicated and complex at first, but searching and replacing text in files with sed is very simple.

To find out more: https://www.gnu.org/software/sed/manual/sed.html

The `vim` command

The vim is a text editor for Unix that comes with Linux, BSD, and macOS. It is known to be fast and powerful, partly because it is a small program that can run in a terminal (although it has a graphical interface). Vim text editor is developed by Bram Moolenaar. It supports most file types and the vim editor is also known as a programmer's editor. It is mainly because it can be managed entirely without menus or a mouse with a keyboard.

Note: Do not confuse `vim` with `vi`. `vi`, which stands for "Visual", is a text editor that was developed by Bill Joy in 1976. `vim` stands for "Vi Improved", and is an improved clone of the `vi` editor.

The most searched question about `vim` :

`How to exit vim editor?`

The most searched question about vim editor looks very funny but it's true that the new user gets stuck at the very beginning when using vim editor.

The command to save the file and exit vim editor: `:wq`

The command to exit vim editor without saving the file: `:q!`

Fun reading:

Here's a survey for the same question, look at this and do not think to quit the vim editor.

Installation:

First check if vim is already installed or not, enter the following command:

```
vim --version
```

If it is already installed it will show its version, else we can run the below commands for the installations:

On Ubuntu/Debian:

```
sudo apt-get install vim
```

On CentOS/Fedora:

```
sudo yum install vim
```

If you want to use advanced features on CentOS/Fedora, you'll need to install enhanced vim editor, to do this run the following command:

```
sudo yum install -y vim-enhanced
```

Syntax:

```
vim [FILE_PATH/FILE_NAME]
```

Examples:

1. To open the file named "demo.txt" from your current directory:

```
vim demo.txt
```

2. To open the file in a specific directory:

```
vim {File_Path/filename}
```

3. To open the file starting on a specific line in the file:

```
vim {File_Path/filename} +LINE_NUMBER
```

Modes in vim editor:

There are some arguments as to how many modes that vim has, but the modes you're most likely to use are `command mode` and `insert mode`. These modes will allow you to do just about anything you need, including creating your document, saving your document, and doing advanced editing, including taking advantage of search and replace functions.

Workflow of `vim` editor:

1. Open a new or existing file with `vim filename`.
2. Type `i` to switch into insert mode so that you can start editing the file.
3. Enter or modify the text of your file.
4. When you're done, press the `Esc` key to exit insert mode and back to command mode.
5. Type :w or :wq to save the file or save and exit from the file respectively.

Interactive training

In this interactive tutorial, you will learn the different ways to use the `vim` command:

The Open vim Tutorial

Additional Flags and their Functionalities:

Flags/Options	Description
`-e`	Start in Ex mode (see Ex-mode)
`-R`	Start in read-only mode
`-R`	Start in read-only mode
`-g`	Start the GUI
`-eg`	Start the GUI in Ex mode
`-Z`	Like "vim", but in restricted mode
`-d`	Start in diff mode diff-mode
`-h`	Give usage (help) message and exit

Flags/Options	Description
`+NUMBER`	Open a file and place the cursor on the line number specified by NUMBER

Read more about vim:

vim can not be learned in a single day, use in day-to-day tasks to get hands-on in vim editor.

To learn more about `vim` follow the given article:

Article By Daniel Miessler

The `chown` command

The `chown` command makes it possible to change the ownership of a file or directory. Users and groups are fundamental in Linux, with `chown` you can change the owner of a file or directory. It's also possible to change ownership on folders recursively

Examples:

1. Change the owner of a file

```
chown user file.txt
```

2. Change the group of a file

```
chown :group file.txt
```

3. Change the user and group in one line

```
chown user:group file.txt
```

4. Change to ownership on a folder recursively

```
chown -R user:group folder
```

Syntax:

```
chown [-OPTION] [DIRECTORY_PATH]
```

The `find` command

The `find` command lets you **search for files in a directory hierarchy**

- Search a file with specific name.
- Search a file with pattern
- Search for empty files and directories.

Examples:

1. Search a file with specific name:

```
find ./directory1 -name sample.txt
```

2. Search a file with pattern:

```
find ./directory1 -name '*.txt'
```

3. To find all directories whose name is test in / directory.

```
find / -type d -name test
```

4. Searching empty files in current directory

```
find . -size 0k
```

Syntax:

```
find [options] [paths] [expression]
```

In Simple words

```
find [where to start searching from]
 [expression determines what to find] [-options] [what to
find]
```

Additional Flags and their Functionalities:

Commonly-used primaries include:

- `name` pattern - tests whether the file name matches the shell-glob pattern given.
- `type` type - tests whether the file is a given type. Unix file types accepted include:

options	Description
`b`	block device (buffered)
`d`	directory
`f`	regular file
`l`	Symbolic link
`-print`	always returns true; prints the name of the current file plus a newline to the stdout.
`-mtime n`	find's all the files which are modified n days back.
`-atime n`	find's all the files which are accessed 50 days back.
`-cmin n`	find's all the files which are modified in the last 1 hour.
`-newer file`	find's file was modified more recently than file.
`-size n`	File uses n units of space, rounding up.

Help Command

Run below command to view the complete guide to `find` command or click here.

```
man find
```

The `rmdir` command

The **rmdir** command is used to remove empty directories from the filesystem in Linux. The rmdir command removes each and every directory specified in the command line only if these directories are empty.

Usage and Examples:

1. remove directory and its ancestors

```
rmdir -p a/b/c                          // is similar to 'rmdir
a/b/c a/b a'
```

2. remove multiple directories

```
rmdir a b c                             // removes empty
directories a,b and c
```

Syntax:

```
rmdir [OPTION]... DIRECTORY...
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
-	`--ignore-fail-on-non-empty`	ignore each failure that is solely because a directory is non-empty
`-p`	`--parents`	remove DIRECTORY and its ancestors
`-d`	`--delimiter=DELIM`	use DELIM instead of TAB for field delimiter

Short Flag	Long Flag	Description
`-v`	`--verbose`	output a diagnostic for every directory processed

The `lsblk` command

Summary

The `lsblk` command displays the block and loop devices on the system. It is especially useful when you want to format disks, write filesystems, check the filesystem and know the mount point of a device.

Examples

1. Basic usage is fairly simple - just execute 'lsblk' sans any option.

```
lsblk
```

2. Make lsblk display empty devices as well

```
lsblk -a
```

3. Make lsblk print size info in bytes

```
lsblk -b
```

4. Make lsblk print zone model for each device

```
lsblk -z
```

5. Make lsblk skip entries for slaves

```
lsblk -d
```

6. Make lsblk use ascii characters for tree formatting

```
lsblk -i
```

7. Make lsblk display info about device owner, group, and mode

```
lsblk -m
```

8. Make lsblk output select columns

```
lsblk -o NAME,SIZE
```

Syntax

```
lsblk [options] [<device> ...]
```

Reading information given by `lsblk`

On running `lsblk` with no flags or command-line arguments, it writes general disk information to the STDOUT. Here is a table that interpretes that information:

Column Name	Meaning	Interpretation
NAME	Name of the device.	Shows name of the device.
RM	Removable.	Shows 1 if the device is removable, 0 if not.
SIZE	Size of the device.	Shows size of the device.
RO	Read-Only.	Shows 1 if read-only, 0 if not.
TYPE	The type of block or loop device.	Shows `disk` for entire disk and `part` for partitions.
MOUNTPOINTS	Where the device is mounted.	Shows where the device is mounted. Empty if not mounted.

Reading information of a specific device

`lsblk` can display information of a specific device when the device's absolute path is passed to it. For example, `lsblk` command for displaying the information of the `sda` disk is:

```
lsblk /dev/sda
```

Useful flags for `lsblk`

Here is a table that show some of the useful flags that can be used with lsblk

Short Flag	Long Flag	Description
`-a`	`--all`	`lsblk` does not list empty devices by default. This option disables this restriction.
`-b`	`--bytes`	Print the SIZE column in bytes rather than in human-readable format.
`-d`	`--nodeps`	Don't print device holders or slaves.
`-D`	`--discard`	Print information about the discard (TRIM, UNMAP) capabilities for each device.
`-E`	`--dedup column`	Use column as a de-duplication key to de-duplicate output tree. If the key is not available for the device, or the device is a partition and parental whole-disk device provides the same key than the device is always printed.
`-e`	`--exclude list`	xclude the devices specified by a comma-separated list of major device numbers. Note that RAM disks (major=1) are excluded by default. The filter is applied to the top-level devices only.
`-f`	`--fs`	Displays information about filesystem.
`-h`	`--help`	Print a help text and exit.
`-l`	`--include list`	Displays all the information in List Format.
`-J`	`--json`	Displays all the information in JSON Format.
`-l`	`--list`	Displays all the information in List Format.
`-m`	`--perms`	Displays info about device owner, group and mode.
`-M`	`--merge`	Group parents of sub-trees to provide more readable output for RAIDs and Multi-path devices. The tree-like output is required.
`-n`	`--noheadings`	Do not print a header line.
`-o`	`--output list`	Specify which output columns to print. Use `--help` to get a list of all supported columns.
`-O`	`--output-all`	Displays all available columns.
`-p`	`--paths`	Displays absolute device paths.
`-P`	`--pairs`	Use key="value" output format. All potentially unsafe characters are hex-escaped (\x)
`-r`	`--raw`	Use the raw output format. All potentially unsafe characters are hex-escaped (\x) in NAME, KNAME, LABEL, PARTLABEL and MOUNTPOINT columns.
`-S`	`--scsi`	Output info about SCSI devices only. All partitions, slaves and holder devices are ignored.
`-s`	`--inverse`	Print dependencies in inverse order.
`-t`	`--topology`	Output info about block device topology. This option is equivalent to "-o NAME,ALIGNMENT,MIN-IO,OPT-IO,PHY-SEC,LOG-SEC,ROTA,SCHED,RQ-SIZE".
`-T`	`--tree[=column]`	Displays all the information in Tree Format.
`-V`	`--version`	Output version information and exit.
`-w`	`--width`	pecifies output width as a number of characters. The default is the number of the terminal columns, and if not executed ona terminal, then output width is not restricted at all by default.
`-x`	`--sort [column]`	Sort output lines by column. This option enables `--list` output format by default. It is possible to use the option `--tree` to force tree-like output and than the tree branches are sorted by the column.
`-z`	`--zoned`	Print the zone model for each device.
-	`--sysroot directory`	Gather data for a Linux instance other than the instance from which the lsblk command is issued. The specified directory is the system root of the Linux instance to be inspected.

Exit Codes

Like every Unix / Linux Program, `lslbk` returns an exit code to the environment. Here is a table of all the exit codes.

Exit Code	Meaning
0	Exit with success.
1	Exit with failure.
32	Specified device(s) not found.
64	Some of the specified devices were found while some not.

The `cmatrix` command

This command doesn't come by default in Linux. It has to be installed, and as seen in chapter 052 we need to run the following command:

```
sudo apt-get install cmatrix
```

And after everything is installed, you have become a 'legit hacker'. In order to use this command, just type in `cmatrix` and press enter:

```
cmatrix
```

And this is what you should see:

As you can see you have access to the matrix now. Well, not really.

What this actually is just a fun little command to goof around with. There are actually a few options you can use. For examle you can change the text colour. You can choose from **green, red, blue, white, yellow, cyan, magenta and black**.

```
cmatrix -C red
```

And the falling characters will be red. This command isn't really something that will help you with your job or anything, but it is fun to know that you can have some fun in Linux.

The `chmod` command

The `chmod` command allows you to change the permissions on a file using either a symbolic or numeric mode or a reference file.

Examples:

1. Change the permission of a file using symbolic mode:

```
chmod u=rwx,g=rx,o=r myfile
```

The command above means :

- user can read, write, execute `myfile`
- group can read, execute `myfile`
- other can read `myfile`

2. Change the permission of a file using numeric mode

```
chmod 754 myfile user:group file.txt
```

The command above means :

- user can read, write, execute `myfile`
- group can read, execute `myfile`
- other can read `myfile`

3. Change the permission of a folder recursively

```
chmod -R 754 folder
```

Syntax:

```
chmod [OPTIONS] MODE FILE(s)
```

- `[OPTIONS]` : `-R`: recursive, mean all file inside directory

- `MODE`: different way to set permissions:

- **Symbolic mode explained**

    - u: user
    - g: group
    - o: other
    - =: set the permission
    - r: read
    - w: write
    - x: execute
    - example `u=rwx` means user can read write and execute

- **Numeric mode explained**:

The **numeric mode** is based off of a binary representation of the permissions for user, group, and others, for more information please look at this explanation from Digital Ocean's community section:

- 4 stands for "read",
- 2 stands for "write",
- 1 stands for "execute", and
- 0 stands for "no permission."
- example 7 mean read + write + execute

The `grep` command

The `grep` filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. grep stands for globally search for regular expression and print out. The pattern that is searched in the file is referred to as the regular expression.

Examples:

1. To search the contents of the destination.txt file for a string("KeY") case insensitively.

```
grep -i "KeY" destination.txt
```

2. Displaying the count of number of matches

```
grep -c "key" destination.txt
```

3. We can search multiple files and only display the files that contains the given string/pattern.

```
grep -l "key" destination1.txt destination2.txt
destination3.xt destination4.txt
```

4. To show the line number of file with the line matched.

```
grep -n "key" destination.txt
```

5. If you want to grep the monitored log files, you can add the `--line-buffered` to search them in real time.

```
tail -f destination.txt | grep --line-buffered "key"
```

Syntax:

The general syntax for the grep command is as follows:

```
grep [options] pattern [files]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
-c	--count	print a count of matching lines for each input file
-h	--no-filename	Display the matched lines, but do not display the filenames
-i	--ignore-case	Ignores, case for matching
-l	--files-with-matches	Displays list of a filenames only.
-n	--line-number	Display the matched lines and their line numbers.
-v	--invert-match	This prints out all the lines that do not matches the pattern
-e	--regexp=	Specifies expression with this option. Can use multiple times
-f	--file=	Takes patterns from file, one per line.
-F	--fixed-strings=	Interpret patterns as fixed strings, not regular expressions.
-E	--extended-regexp	Treats pattern as an extended regular expression (ERE)
-w	--word-regexp	Match whole word
-o	--only-matching	Print only the matched parts of a matching line, with each such part on a separate output line.
	--line-buffered	Force output to be line buffered.

The `screen` command

`screen` - With screen you can start a screen session and then open any number of windows (virtual terminals) inside that session. Processes running in Screen will continue to run when their window is not visible even if you get disconnected. This is very handy for running long during session such as bash scripts that run very long.

To start a screen session you type `screen`, this will open a new screen session with a virtual terminal open.

Below are some most common commands for managing Linux Screen Windows:

Command	Description
`Ctrl+a+ c`	Create a new window (with shell).
`Ctrl+a+ "`	List all windows.
`Ctrl+a+ 0`	Switch to window 0 (by number).
`Ctrl+a+ A`	Rename the current window.
`Ctrl+a+ S`	Split current region horizontally into two regions.
`Ctrl+a+ '`	Split current region vertically into two regions.
`Ctrl+a+ tab`	Switch the input focus to the next region.
`Ctrl+a+ Ctrl+a`	Toggle between the current and previous windows
`Ctrl+a+ Q`	Close all regions but the current one.
`Ctrl+a+ X`	Close the current region.

Restore a Linux Screen

To restore to a screen session you type `screen -r`, if you have more than one open screen session you have to add the session id to the command to connect to the right session.

Listing all open screen sessions

To find the session ID you can list the current running screen sessions with:

```
screen -ls
```

There are screens on:

```
18787.pts-0.your-server   (Detached)
15454.pts-0.your-server   (Detached)
2 Sockets in /run/screens/S-yourserver.
```

If you want to restore screen 18787.pts-0, then type the following command:

```
screen -r 18787
```

The `nc` command

The `nc` (or netcat) command is used to perform any operation involving TCP (Transmission Control Protocol, connection oriented), UDP (User Datagram Protocol, connection-less, no guarantee of data delivery) or UNIX-domain sockets. It can be thought of as swiss-army knife for communication protocol utilities.

Syntax:

```
nc [options] [ip] [port]
```

Examples:

1. Open a TCP connection to port 80 of host, using port 1337 as source port with timeout of 5s:

```
$ nc -p 1337 -w 5 host.ip 80
```

2. Open a UDP connection to port 80 on host:

```
$ nc -u host.ip 80
```

3. Create and listen on UNIX-domain stream socket:

```
$ nc -lU /var/tmp/dsocket
```

4. Create a basic server/client model:

This creates a connection, with no specific server/client sides with respect to nc, once the connection is established.

```
$ nc -l 1234 # in one console

$ nc 127.0.0.1 1234 # in another console
```

5. Build a basic data transfer model:

After the file has been transferred, sequentially, the connection closes automatically

```
$ nc -l 1234 > filename.out # to start listening in one
console and collect data

$ nc host.ip 1234 < filename.in
```

6. Talk to servers:

Basic example of retrieving the homepage of the host, along with headers.

```
$ printf "GET / HTTP/1.0\r\n\r\n" | nc host.ip 80
```

7. Port scanning:

Checking which ports are open and running services on target machines. `-z` flag commands to inform about those rather than initiate a connection.

```
$ nc -zv host.ip 20-2000 # range of ports to check for
```

Flags and their Functionalities:

Short Flag	Description
`-4`	Forces nc to use IPv4 addresses
`-6`	Forces nc to use IPv6 addresses
`-b`	Allow broadcast
`-D`	Enable debugging on the socket
`-i`	Specify time interval delay between lines sent and received
`-k`	Stay listening for another connection after current is over
`-l`	Listen for incoming connection instead of initiate one to remote

Short Flag	Description
`-T`	Specify length of TCP
`-p`	Specify source port to be used
`-r`	Specify source and/or destination ports randomly
`-s`	Specify IP of interface which is used to send the packets
`-U`	Use UNIX-domain sockets
`-u`	Use UDP instead of TCP as protocol
`-w`	Declare a timeout threshold for idle or unestablished connections
`-x`	Should use specified protocol when talking to proxy server
`-z`	Specify to scan for listening daemons, without sending any data

The `make` command

The `make` command is used to automate the reuse of multiple commands in certain directory structure.

An example for that would be the use of `terraform init`, `terraform plan`, and `terraform validate` while having to change different subscriptions in Azure. This is usually done in the following steps:

```
az account set --subscription "Subscription - Name"
terraform init
```

How the `make` command can help us is it can automate all of that in just one go: `make tf-init`

Syntax:

```
make [ -f makefile ] [ options ] ... [ targets ] ...
```

Example use (guide):

1. Create `Makefile` in your guide directory

2. Include the following in your `Makefile` :

```
hello-world:
	echo "Hello, World!"

hello-bobby:
	echo "Hello, Bobby!"

touch-letter:
	echo "This is a text that is being inputted into our
letter!" > letter.txt

clean-letter:
	rm letter.txt
```

3. Execute `make hello-world` - this echoes "Hello, World" in our terminal.

4. Execute `make hello-bobby` - this echoes "Hello, Bobby!" in our terminal.

5. Execute `make touch-letter` - This creates a text file named `letter.txt` and populates a line in it.

6. Execute `make clean-letter`

References to lenghtier and more contentful tutorials:

(linoxide - linux make command examples)[https://linoxide.com/linux-make-command-examples/] (makefiletutorial.com - the name itself gives it out)[https://makefiletutorial.com/]

The `basename` command

The `basename` is a command-line utility that strips directory from given file names. Optionally, it can also remove any trailing suffix. It is a simple command that accepts only a few options.

Examples

The most basic example is to print the file name with the leading directories removed:

```
basename /etc/bar/foo.txt
```

The output will include the file name:

```
foo.txt
```

If you run basename on a path string that points to a directory, you will get the last segment of the path. In this example, /etc/bar is a directory.

```
basename /etc/bar
```

Output

```
bar
```

The basename command removes any trailing `/` characters:

```
basename /etc/bar/foo.txt/
```

Output

```
foo.txt
```

Options

1. By default, each output line ends in a newline character. To end the lines with NUL, use the -z (--zero) option.

```
$ basename -z /etc/bar/foo.txt
foo.txt$
```

2. The `basename` command can accept multiple names as arguments. To do so, invoke the command with the `-a` (`--multiple`) option, followed by the list of files separated by space. For example, to get the file names of `/etc/bar/foo.txt` and `/etc/spam/eggs.docx` you would run:

```
basename -a /etc/bar/foo.txt /etc/spam/eggs.docx
```

```
foo.txt
eggs.docx
```

Syntax

The basename command supports two syntax formats:

```
basename NAME [SUFFIX]
basename OPTION... NAME...
```

Additional functionalities

Removing a Trailing Suffix: To remove any trailing suffix from the file name, pass the suffix as a second argument:

```
basename /etc/hostname name
host
```

Generally, this feature is used to strip file extensions

Help Command

Run the following command to view the complete guide to `basename` command.

```
man basename
```

The `banner` command

The `banner` command writes ASCII character Strings to standard output in large letters. Each line in the output can be up to 10 uppercase or lowercase characters in length. On output, all characters appear in uppercase, with the lowercase input characters appearing smaller than the uppercase input characters.

Note: If you will define more than one NUMBER with sleep command then this command will delay for the sum of the values.

Examples :

1. To display a banner at the workstation, enter:

```
banner LINUX!
```

2. To display more than one word on a line, enclose the text in quotation marks, as follows:

```
banner "Intro to" Linux
```

This displays Intro to on one line and Linux on the next

3. Printing “101LinuxCommands” in large letters.

```
banner 101LinuxCommands
```

It will print only 101LinuxCo as banner has a default capacity of 10

The `alias` command

The `alias` command lets you create shortcuts for commands or define your own commands.
This is mostly used to avoid typing long commands.

Examples:

1. To show the list of all defined aliases in the reusable form `alias NAME=VALUE` :

```
alias -p
```

2. To make `ls -A` shortcut:

```
alias la='ls -A'
```

Syntax:

```
alias [-p] [name[=value]]
```

Setting Persistent Options:

As with most Linux custom settings for the terminal, any alias you defined is only applied to the current opening terminal session.

For any alias to be active for all new sessions you need to add that command to your rc file to be executed in the startup of every new terminal. this file can be as follows:

- **Bash**: ~/.bashrc
- **ZSH**: ~/.zshrc

- **Fish** – ~/.config/fish/config.fish

you can open that file with your favorite editor as follows:

```
vim ~/.bashrc
```

type your commands one per line, then save the file and exit. the commands will be automatically applied in the next session.

If you want to apply it in the current session, run the following command:

```
source ~/.bashrc
```

Opposite command:

To remove predefined alias you can use `unalias` command as follows:

```
unalias alias_name
```

to remove all aliases

```
unalias -a
```

The `which` command

`which` command identifies the executable binary that launches when you issue a command to the shell. If you have different versions of the same program on your computer, you can use which to find out which one the shell will use.

It has 3 return status as follows:

```
0 : If all specified commands are found and executable.
1 : If one or more specified commands is nonexistent or not
executable.
2 : If an invalid option is specified.
```

Examples

1. To find the full path of the ls command, type the following:

```
which ls
```

2. We can provide more than one arguments to the which command:

```
which netcat uptime ping
```

The which command searches from left to right, and if more than one matches are found in the directories listed in the PATH path variable, which will print only the first one.

3. To display all the paths for the specified command:

```
which [filename] -a
```

4. To display the path of node executable files, execute the command:

```
which node
```

5. To display the path of Java executable files, execute:

```
which java
```

Syntax

```
which [filename1] [filename2] ...
```

You can pass multiple programs and commands to which, and it will check them in order.

For example:

```
which ping cat uptime date head
```

Options

-a : List all instances of executables found (instead of just the first one of each).

-s : No output, just return 0 if all the executables are found, or 1 if some were not found

The `date` command

The `date` command is used to print the system current date and time.

`date` command is also used to set the date and time of the system, but you need to be the super-user *(root)* to do it.

Examples:

1. To show the current date and time:

```
date
```

2. You can use -u option to show the date and time in UTC *(Coordinated Universal Time)* time zone

```
date -u
```

1. To display any given date string in formatted date:

```
date --date="2/02/2010"
date --date="2 years ago"
```

Syntax:

```
date [OPTION]... [+FORMAT]
date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-d`	`--date=STRING`	convert the provided string into formatted date
`-f`	`--file=DATEFILE`	like `--date` but for files
`-I[FMT]`	`--iso-8601[=FMT]`	Display date and time in ISO 8601 format
`-r`	`--reference=FILE`	Display the last modification time of FILE
`-s`	`--set=STRING`	sets the time to the one described by STRING
`-u`	`--universal`	show the date and time in UTC *(Coordinated Universal Time)* time zone
`-R`	`--rfc-email`	Display date and time in ISO 8601 format Example: (Fri, 22 Oct 2021 05:18:42 +0200)
-	`rfc-3339=FMT`	Display date and time in RFC 3339 format
-	`--debug`	Usually used with `--date` to annotate the parsed date and warn about questionable usage to stderr

Control The output:

You can use Format specifiers to control the output date and time.

Examples:

Command	Output
`$ date "+%D"`	`10/22/21`
`$ date "+%D %T"`	`10/22/21 05:33:51`
`$ date "+%A %B %d %T %y"`	`Friday October 22 05:34:47 21`

Syntax:

```
date "+%[format-options ...]"
```

List of Format specifiers to control the output:

Specifiers	Description
`%a`	abbreviated weekday name *(e.g., Sun)*

Specifiers	Description
%A	full weekday name *(e.g., Sunday)*
%b	abbreviated month name *(e.g., Jan)*
%B	full month name *(e.g., January)*
%c	date and time *(e.g., Thu Mar 3 23:05:25 2005)*
%C	century; like %Y, except omit last two digits (e.g., 20)
%d	day of month (e.g., 01)
%D	date; same as %m/%d/%y
%e	day of month, space padded; same as %_d
%F	full date; same as %Y-%m-%d
%g	last two digits of year of ISO week number (see %G)
%G	year of ISO week number (see %V); normally useful only with %V
%h	same as %b
%H	hour (00..23)
%I	hour (01..12)
%j	day of year (001..366)
%k	hour, space padded (0..23); same as %_H
%l	hour, space padded (1..12); same as %_I
%m	month (01..12)
%M	minute (00..59)
%n	a newline
%N	nanoseconds (000000000..999999999)
%p	locale's equivalent of either AM or PM; blank if not known
%P	like %p, but lower case
%q	quarter of year (1..4)
%r	locale's 12-hour clock time (e.g., 11:11:04 PM)
%R	24-hour hour and minute; same as %H:%M
%s	seconds since 1970-01-01 00:00:00 UTC
%S	second (00..60)
%t	a tab
%T	time; same as %H:%M:%S
%u	day of week (1..7); 1 is Monday
%U	week number of year, with Sunday as first day of week (00..53)
%V	ISO week number, with Monday as first day of week (01..53)
%w	day of week (0..6); 0 is Sunday
%W	week number of year, with Monday as first day of week (00..53)
%x	locale's date representation (e.g., 12/31/99)
%X	locale's time representation (e.g., 23:13:48)

Specifiers	Description
%y	last two digits of year (00..99)
%Y	year
%z	+hhmm numeric time zone (e.g., -0400)
%:z	+hh:mm numeric time zone (e.g., -04:00)
%::z	+hh:mm:ss numeric time zone (e.g., -04:00:00)
%:::z	numeric time zone with : to necessary precision (e.g., -04, +05:30)
%Z	alphabetic time zone abbreviation (e.g., EDT)

The `mount` command

The `mount` command is used to mount 'attach' a filesystem and make it accessible by an existing directory structure tree.

Examples:

1. Displays version information:

```
mount -V
```

2. Attaching filesystem found on device and of type type at the directory dir:

```
mount -t type device dir
```

Syntax Forms:

```
mount [-lhV]
```

```
mount -a [-fFnrsvw] [-t vfstype] [-O optlist]
```

```
mount [-fnrsvw] [-t fstype] [-o options] device dir
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-h`	`--help`	Dispaly a help message and exists
`-n`	`--no-mtab`	Mount without writing in /etc/mtab
`-a`	`--all`	Mount all filesystems (of the given types) mentioned in fstab
`-r`	`--read-only`	Mount the filesystem read-only
`-w`	`--rw`	Mount the filesystem as read/write.
`-M`	`--move`	Move a subtree to some other place.
`-B`	`--bind`	Remount a subtree somewhere else *(so that its contents are available in both places)*.

The `nice/renice` command

The `nice/renice` commands is used to modify the priority of the program to be executed. The priority range is between -20 and 19 where 19 is the lowest priority.

Examples:

1. Running cc command in the background with a lower priority than default (slower):

```
nice -n 15 cc -c *.c &
```

2. Increase the priority to all processes belonging to group "test":

```
renice --20 -g test
```

Syntax:

```
nice [  -Increment|  -n Increment ] Command [ Argument ... ]
```

Flags :

Short Flag	Long Flag	Description
`-Increment`	-	Increment is the value of priority you want to assign.
`-n Increment`	-	Same as `-Increment`

The `wc` command

the `wc` command stands for word count. It's used to count the number of lines, words, and bytes *(characters)* in a file or standard input then prints the result to the standard output.

Examples:

1. To count the number of lines, words and characters in a file in order:

```
wc file.txt
```

2. To count the number of directories in a directory:

```
ls -F | grep / | wc -l
```

Syntax:

```
wc [OPTION]... [FILE]...
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-c`	`--bytes`	print the byte counts
`-m`	`--chars`	print the character counts
`-l`	`--lines`	print the newline counts
-	`--files0-from=F`	read input from the files specified by NUL-terminated names in file F. If F is `-` then read names from standard input

Short Flag	Long Flag	Description
`-L`	`--max-line-length`	print the maximum display width
`-w`	`--words`	print the word counts

Additional Notes:

- Passing more than one file to `wc` command prints the counts for each file and the total conuts of them.
- you can combine more whan one flag to print the result as you want.

The `tr` command

The tr command in UNIX is a command line utility for translating or deleting characters. It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters and basic find and replace. It can be used with UNIX pipes to support more complex translation. tr stands for translate.

Examples:

1. Convert all lowercase letters in file1 to uppercase.

```
$ cat file1
foo
bar
baz
tr a-z A-Z < file1
FOO
BAR
BAZ
```

2. Make consecutive line breaks into one.

```
$ cat file1
foo


bar


baz
$ tr -s "\n" < file1
foo
bar
baz
```

3. Remove the newline code.

```
$ cat file1
foo
bar
baz
$ tr -d "\n" < file1
foobarbaz%
```

Syntax:

The general syntax for the tr command is as follows:

```
tr [options] string1 [string2]
```

Additional Flags and their Functionalities:

Short Flag	Long Flag	Description
`-C`		Complement the set of characters in string1, that is `-C ab` includes every character except for `a` and `b`.
`-c`		Same as -C.
`-d`		Delete characters in string1 from the input.
`-s`		If there is a sequence of characters in string1, combine them into one.

The `fdisk` command

The `fdisk` command is used for controlling the disk partition table and making changes to it and this is a list of some of options provided by it :

- Organize space for new drives.
- Modify old drives.
- Create space for new partitions.
- Move data to new partitions.

Examples:

1. To view basic details about all available partitions on the system:

```
fdisk -l
```

2. To show the size of the partition:

```
fdisk -s /dev/sda
```

3. To view the help message and all options of the command:

```
fdisk -h
```

Syntax:

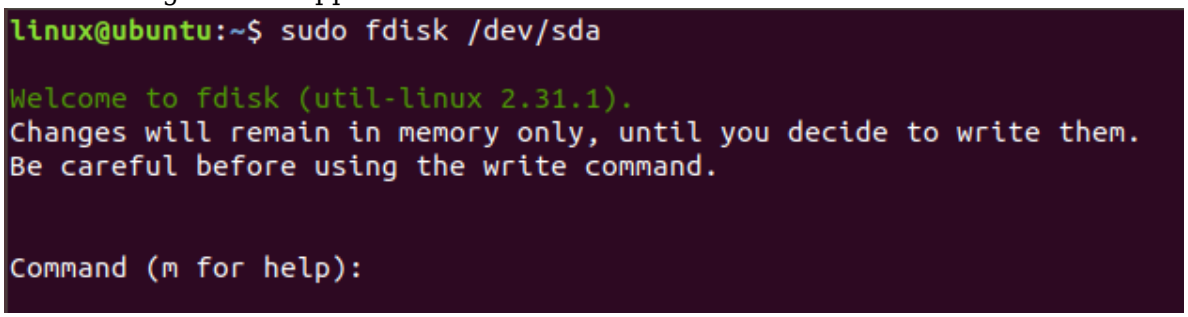
```
fdisk [options] device
```

Some of the command options:

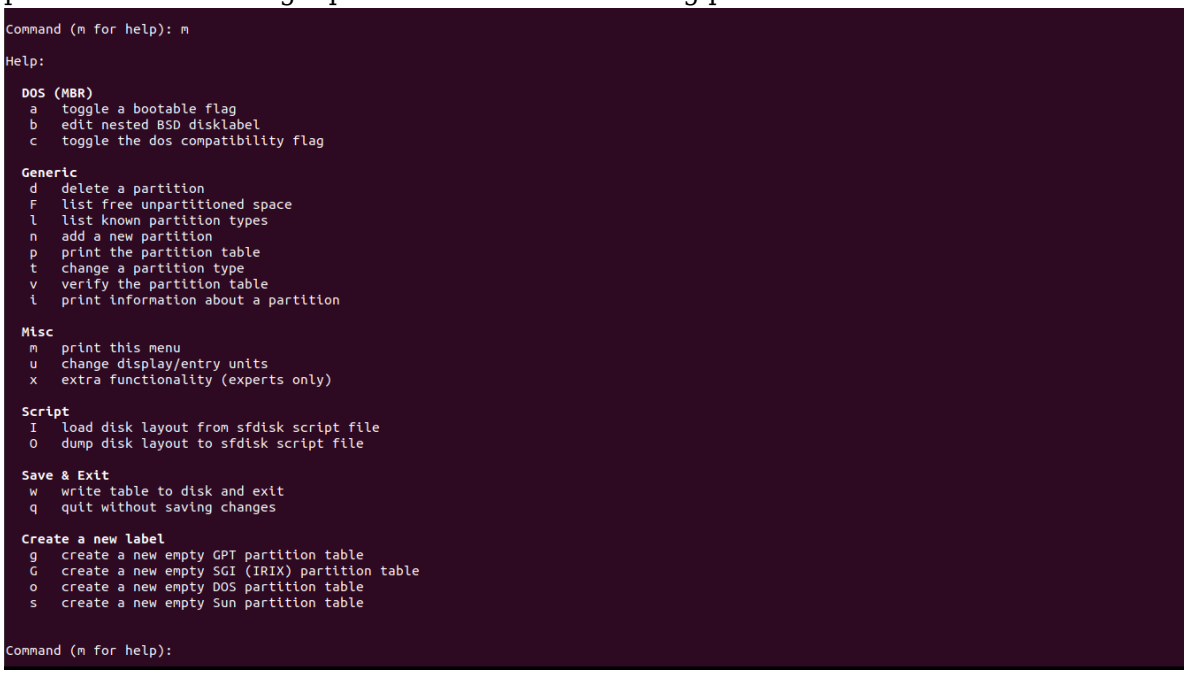
On writing the following command

```
fdisk /dev/sdb
```

the following window appears :



and then you type m which will show you all options you need such as creating new partition and deleting a partition as in the following picture :



The `Wait` commands

It is a command that waits for completing any running process of given id. if the process id is not given then it waits for all current child processes to complete.

Example

This example shows how the `wait` command works :

Step-1:

Create a file named "wait_example.sh" and add the following script to it.

```
#!/bin/bash
echo "Wait command" &
process_id=$!
wait $process_id
echo "Exited with status $?"
```

Step-2:

Run the file with bash command.

```
$ bash wait_example.sh
```

The `zcat` command

The `zcat` allows you to look at a compressed file.

Examples:

1. To view the content of a compressed file:

```
~$ zcat test.txt.gz
Hello World
```

2. It can also Works with multiple files:

```
~$ zcat test2.txt.gz test.txt.gz
hello
Hello world
```

Syntax:

The general syntax for the `zcat` command is as follows:

```
zcat [  -n ] [  -V ] [  File ... ]
```

The `fold` command

The `fold` command in Linux wraps each line in an input file to fit a specified width and prints it to the standard output.

By default, it wraps lines at a maximum width of 80 columns but this is configurable.

To fold input using the fold command pass a file or standard input to the command.

Syntax:

```
fold [OPTION]... [FILE]...
```

Options

-w : By using this option in fold command, we can limit the width by number of columns.

By using this command we change the column width from default width of 80. Syntax:

```
fold -w[n] [FILE]
```

Example: wrap the lines of file1.txt to a width of 60 columns

```
fold -w60 file1.txt
```

-b : This option of fold command is used to limit the width of the output by the number of bytes rather than the number of columns.

By using this we can enforce the width of the output to the number of bytes.

```
fold -b[n] [FILE]
```

Example: limit the output width of the file to 40 bytes and the command breaks the output at 40 bytes.

```
fold -b40 file1.txt
```

-s : This option is used to break the lines on spaces so that words are not broken.

If a segment of the line contains a blank character within the first width column positions, break the line after the last such blank character meeting the width constraints.

```
fold -w[n] -s [FILE]
```

The `quota` command

The `quota` display disk usage and limits.

Installation:

You can simply go ahead and install quota on ubuntu systems by running:

```
sudo apt-get install quota
```

for Debian use the install command without sudo:

```
apt-get install quota
```

Syntax:

The general syntax for the `quota` command is as follows:

```
quota [ -u [ User ] ] [ -g [ Group ] ] [ -v | -q ]
```

The `aplay` command

`aplay` is a command-line audio player for ALSA(Advanced Linux Sound Architecture) sound card drivers. It supports several file formats and multiple soundcards with multiple devices. It is basically used to play audio on command-line interface. aplay is much the same as arecord only it plays instead of recording. For supported soundfile formats, the sampling rate, bit depth, and so forth can be automatically determined from the soundfile header.

Syntax:

```
$ aplay [flags] [filename [filename]] ...
```

Options:

```
-h, –help : Show the help information.
-d, –duration=# : Interrupt after # seconds.
-r, –rate=# : Sampling rate in Hertz. The default rate is 8000
Hertz.
–version : Print current version.
-l, –list-devices : List all soundcards and digital audio
devices.
-L, –list-pcms : List all PCMs(Pulse Code Modulation) defined.
-D, –device=NAME : Select PCM by name.
```

Note: This command contain various other options that we normally don’t need. If you want to know more about you can simply run following command on your terminal.

```
aplay --help
```

Examples :

1. To play audio for only 10 secs at 2500hz frequency.

   ```
   $ aplay -d 10 -r 2500hz sample.mp3
   ```

   Plays sample.mp3 file for only 10 secs at 2500hz frequency.

2. To play full audio clip at 2500hz frezuency.

   ```
   $ aplay -r 2500hz sample.mp3
   ```

   Plays sample.mp3 file at 2500hz frezuency.

3. To Display version information.

   ```
   $ aplay --version
   ```

   Displays version information. For me it shows aplay: vesrion 1.1.0

The `spd-say` command

`spd-say` sends text-to-speech output request to speech-dispatcher process which handles it and ideally outputs the result to the audio system.

Syntax:

```
$ spd-say [options] "some text"
```

Options:

```
-r, --rate
       Set the rate of the speech (between -100 and +100,
default: 0)

-p, --pitch
       Set the pitch of the speech (between -100 and +100,
default: 0)

-i, --volume
       Set the volume (intensity) of the speech (between -100
and +100, default: 0)

-o, --output-module
       Set the output module

-l, --language
       Set the language (iso code)

-t, --voice-type
       Set the preferred voice type  (male1,  male2,  male3,
female1,  female2,  female3,
       child_male, child_female)

-m, --punctuation-mode
       Set the punctuation mode (none, some, all)

-s, --spelling
       Spell the message

-x, --ssml
       Set SSML mode on (default: off)

-e, --pipe-mode
       Pipe from stdin to stdout plus Speech Dispatcher

-P, --priority
       Set  priority  of  the  message  (important, message,
text, notification, progress;
       default: text)

-N, --application-name
```

```
       Set the application name used to establish the
connection to specified string value
       (default: spd-say)

-n, --connection-name
       Set  the connection name used to establish the
connection to specified string value
       (default: main)

-w, --wait
       Wait till the message is spoken or discarded

-S, --stop
       Stop speaking the message being spoken in Speech
Dispatcher

-C, --cancel
       Cancel all messages in Speech Dispatcher

-v, --version
       Print version and copyright info

-h, --help
       Print this info
```

Examples :

1. To Play the given text as the sound.

```
$ spd-say "Hello"
```

Plays "Hello" in sound.

The `xeyes` command

Xeyes is a graphical user interface program that creates a set of eyes on the desktop that follow the movement of the mouse cursor. It seems much of a funny command, than of any useful use. Being funny is as much useful, is another aspect.

Syntax:

```
xeyes
```

What is the purpose of xeyes?

`xeyes` is not for fun, at least not only. The purpose of this program is to let you follow the mouse pointer which is sometimes hard to see. It is very useful on multi-headed computers, where monitors are separated by some distance, and if someone (say teacher at school) wants to present something on the screen, the others on their monitors can easily follow the mouse with `xeyes`.

The `parted` command

The `parted` command is used to manage hard disk partitions on Linux. It can be used to add, delete, shrink and extend disk partitions along with the file systems located on them. You will need root access to the system to run `parted` commands.

NOTE: Parted writes the changes immediately to your disk, be careful when you are modifying the disk partitions.

Examples:

1. Displays partition layout of all block devices:

```
sudo parted -l
```

2. Display partition table of a specific `disk`

```
sudo parted disk print
```

Examples of `disk` are /dev/sda, /dev/sdb

3. Create a new disk label of `label-type` for a specific disk

```
sudo parted mklabel disk label-type
```

`label-type` can take values "aix", "amiga", "bsd", "dvh", "gpt", "loop", "mac", "msdos", "pc98", or "sun"

4. Create a new partition in a specific `disk` of type `part-time`, file system is `fs-type` and of size `size` Mb.

```
sudo parted disk mkpart part-time fs-type 1 size
```

`part-time` can take values "primary", "logical", "extended".
`fs-type` is optional. It can take values "btrfs", "ext2", "ext3", "ext4", "fat16", "fat32", "hfs", "hfs+", "linux-swap", "ntfs", "reiserfs", "udf", or "xfs"
`size` has to less than the total size of the specified disk. To create a partition of size 50Mb, will take the value of 50

5. `parted` can also be run in an interactive format. Operations to manage the disk partitions can be performed by entering appropriate commands in the interactive session. `help` command in the interactive session shows a list of all possible disk management operations which can be performed.

```
  $ sudo parted
  GNU Parted 3.3
  Using /dev/sda
  Welcome to GNU Parted! Type 'help' to view a list of
commands.
  (parted) print  # prints the partition table of the default
selected disk - /dev/sda
  Model: ATA VBOX HARDDISK (scsi)
  Disk /dev/sda: 53.7GB
  Sector size (logical/physical): 512B/512B
  Partition Table: msdos
  Disk Flags:

  Number  Start   End     Size    Type     File system  Flags
   1      1049kB  53.7GB  53.7GB  primary  ext4         boot

  (parted) select /dev/sdb  # change the current disk on which
operations have to be performed
  Using /dev/sdb
  (parted) quit  # exit the interactive session
```

Syntax Forms:

```
parted [options] [device [command [options...]...]]
```

Options:

Short Flag	Long Flag	Description
-h	--help	displays a help message listing all possible `commands [options]`
-l	--list	lists partition layout on all block devices
-m	--machine	displays machine parseable output
-v	--version	displays the version
-a	--align	set alignment type for newly created partition. It can take the following values: `none`: Use the minimum alignment allowed by the disk type `cylinder`: Align partitions to cylinders `minimal`: Use minimum alignment as given by the disk topology information `optimal`: Use optimum alignment as given by the disk topology information

The `nl` command

The “nl” command enumerates lines in a file. A different way of viewing the contents of a file, the “nl” command can be very useful for many tasks.

Syntax

```
nl [ -b Type ] [ -f Type ] [ -h Type ] [ -l Number ] [ -d
Delimiter ] [ -i Number ] [ -n Format ] [ -v Number ] [ -w
Number ] [ -p ] [ -s Separator ] [ File ]
```

Examples:

1. To number all lines:

```
nl  -ba  chap1
```

2. Displays all the text lines:

```
[server@ssh ~]$ nl states
1 Alabama
2 Alaska
3 Arizona
4 Arkansas
5 California
6 Colorado
7 Connecticut.
8 Delaware
```

3. Specify a different line number format

```
nl  -i10  -nrz  -s::  -v10  -w4  chap1
```

You can name only one file on the command line. You can list the flags and the file name in any order.

The `pidof` command

The `pidof` is a command-line utility that allows you to find the process ID of a running program.

Syntax

```
pidof [OPTIONS] PROGRAM_NAME
```

To view the help message and all options of the command:

```
[user@home ~]$ pidof -h

 -c           Return PIDs with the same root directory
 -d <sep>     Use the provided character as output separator
 -h           Display this help text
 -n           Avoid using stat system function on network
shares
 -o <pid>     Omit results with a given PID
 -q           Quiet mode. Do not display output
 -s           Only return one PID
 -x           Return PIDs of shells running scripts with a
matching name
 -z           List zombie and I/O waiting processes. May cause
pidof to hang.
```

Examples:

To find the PID of the SSH server, you would run:

```
pidof sshd
```

If there are running processes with names matching `sshd`, their PIDs will be displayed on the screen. If no matches are found, the output will be empty.

```
# Output
4382 4368 811
```

`pidof` returns `0` when at least one running program matches with the requested name. Otherwise, the exit code is `1`. This can be useful when writing shell scripts.

To be sure that only the PIDs of the program you are searching for are displayed, use the full pathname to the program as an argument. For example, if you have two running programs with the same name located in two different directories pidof will show PIDs of both running programs.

By default, all PIDs of the matching running programs are displayed. Use the `-s` option to force pidof to display only one PID:

```
pidof -s program_name
```

The `-o` option allows you to exclude a process with a given PID from the command output:

```
pidof -o pid program_name
```

When pidof is invoked with the `-o` option, you can use a special PID named %PPID that represents the calling shell or shell script.

To return only the PIDs of the processes that are running with the same root directory, use the `-c` option. This option works only pidof is run as `root` or `sudo` user:

```
pidof -c pid program_name
```

Conclusion

The `pidof` command is used to find out the PIDs of a specific running program.

`pidof` is a simple command that doesn't have a lot of options. Typically you will invoke pidof only with the name of the program you are searching for.

The `shuf` command

The `shuf` command in Linux writes a random permutation of the input lines to standard output. It pseudo randomizes an input in the same way as the cards are shuffled. It is a part of GNU Coreutils and is not a part of POSIX. This command reads either from a file or standard input in bash and randomizes those input lines and displays the output.

Syntax

```
# file shuf
shuf [OPTION] [FILE]

# list shuf
shuf -e [OPTION]... [ARG]

# range shuf
shuf -i LO-HI [OPTION]
```

Like other Linux commands, `shuf` command comes with `--help` option:

```
[user@home ~]$ shuf --help
Usage: shuf [OPTION]... [FILE]
  or:  shuf -e [OPTION]... [ARG]...
  or:  shuf -i LO-HI [OPTION]...
Write a random permutation of the input lines to standard
output.

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

Mandatory arguments to long options are mandatory for short
options too.
  -e, --echo                treat each ARG as an input line
  -i, --input-range=LO-HI   treat each number LO through HI as
an input line
  -n, --head-count=COUNT    output at most COUNT lines
  -o, --output=FILE         write result to FILE instead of
standard output
      --random-source=FILE  get random bytes from FILE
  -r, --repeat              output lines can be repeated
  -z, --zero-terminated     line delimiter is NUL, not newline
```

Examples:

shuf command without any option or argument.

```
shuf
```

When `shuf` command is used without any argument in the command line, it takes input from the user until `CTRL-D` is entered to terminate the set of inputs. It displays the input lines in a shuffled form. If `1, 2, 3, 4 and 5` are entered as input lines, then it generates `1, 2, 3, 4 and 5` in random order in the output as seen in the illustration below:

```
[user@home ~]$ shuf
1
2
3
4
5

4
5
1
2
3
```

Consider an example where Input is taken from the pipe:

```
{
seq 5 | shuf
}
```

`seq 5` returns the integers sequentially from `1` to `5` while the `shuf` command takes it as input and shuffles the content i.e, the integers from `1` to `5`. Hence, `1` to `5` is displayed as output in random order.

```
[user@home ~]$ {
> seq 5 | shuf
> }
5
4
2
3
1
```

File shuf

When `shuf` command is used without `-e` or `-i` option, then it operates as a file shuf i.e, it shuffles the contents of the file. The `<file_name>` is the last parameter of the `shuf` command and if it is not given, then input has to be provided from the shell or pipe.

Consider an example where input is taken from a file:

```
shuf file.txt
```

Suppose file.txt contains 6 lines, then the shuf command displays the input lines in random order as output.

```
[user@home ~]$ cat file.txt
line-1
line-2
line-3
line-4
line-5

[user@home ~]$ shuf file.txt
line-5
line-4
line-1
line-3
line-2
```

Any number of lines can be randomized by using `-n` option.

```
shuf -n 2 file.txt
```

This will display any two random lines from the file.

```
line-5
line-2
```

List shuf

When `-e` option is used with shuf command, it works as a list shuf. The arguments of the command are taken as the input line for the shuf.

Consider an example:

```
shuf -e A B C D E
```

It will take `A, B, C, D, E` as input lines, and will shuffle them to display the output.

```
A
C
B
D
E
```

Any number of input lines can be displayed using the `-n` option along with `-e` option.

```
shuf -e -n 2 A B C D E
```

This will display any two of the inputs.

```
E
A
```

Range shuf

When `-i` option is used along with `shuf` command, it acts as a `range shuf`. It requires a range of input as input where `L0` is the lower bound while `HI` is the upper bound. It displays integers from `L0-HI` in shuffled form.

```
[user@home ~]$ shuf -i 1-5
4
1
3
2
5
```

Conclusion

The `shuf` command helps you randomize input lines. And there are features to limit the number of output lines, repeat lines and even generate random positive integers. Once you're done practicing whatever we've discussed here, head to the tool's man page to know more about it.

The `less` command

The less command is a Linux terminal pager which shows a file's content one screen at a time. Useful when dealing with a large text file because it doesn't load the entire file but accesses it page by page, resulting in fast loading speeds.

Syntax

```
less [options] file_path
```

Options

Some popular option flags include:

```
-E  less automatically exits upon reaching the end of file.
-f      Forces less to open non-regular files (a directory or
a device-special file).
-F      Exit less if the entire file can be displayed on the
first screen.
-g      Highlights the string last found using search. By
default, less highlights all strings matching the last search
command.
-G      Removes all highlights from strings found using
search.
```

For a complete list of options, refer to the less help file by running:

```
    less --help
```

Few Examples:

1. Open a Text File

```
less /etc/updatedb.conf
```

2. Show Line Numbers

```
less -N /etc/init/mysql.conf
```

3. Open File with Pattern Search

```
less -pERROR /etc/init/mysql.conf
```

4. Remove Multiple Blank Lines

```
less welcome.txt
```

Here I showed you how to use the less command in Linux. Although there are other terminal pagers, such as most and more, but less could be a better choice as it is a powerful tool present in almost every system.

For more details:
https://phoenixnap.com/kb/less-command-in-linux#:~:text=The%20less%20command%20is%20a,resulting%20in%20fast%20loading%20speeds.

999-wrap-up.md