

CHAPTER - 2

BASICS OF LINUX

BASIC COMMANDS:

1. pwd

When we first open the terminal, we are in the home directory of our user. To know which directory we are in, we can use the “**pwd**” command. It gives us the Absolute Path, which means the path that starts from the root. The root is the base of the Linux file system. It is denoted by a forward slash (/). The user directory is usually something like /home/username.

```
nayso@Alok-Aspire:~$ pwd
/home/nayso
```

2. ls

The “**ls**” command is used to know what files are there in the directory we are in. We can see all the hidden files by using the command “**ls -a**”.

```
nayso@Alok-Aspire:~$ ls
Desktop          itsuserguide.desktop  reset-settings        VCD_Copy
Documents        Music                  School_Resources      Videos
Downloads        Pictures               Students_Works_10
examples.desktop Public                 Templates
GplatesProject   Qgis Projects         TuxPaint-Pictures
```

3. cd

“**cd**” is the command used to go to a directory. For example, if we are in the home folder, and we want to go to the Downloads folder, then we can type in “**cd Downloads**”. Remember, this command is case sensitive and we have to type in the name of the folder exactly as it is. But there is a problem with these commands. Imagine we have a folder named “**Raspberry Pi**”. In this case, when we type in “**cd Raspberry Pi**”, the shell will take the second argument of the command as a different one, so we will get an error saying that the directory does not exist. Here, we can use a backward slash. That is, we can use “**cd Raspberry\ Pi**” in this case. Spaces are denoted like this: If we just type “**cd**” and press Enter, it takes us to the home directory. To go back from a folder to the folder before that, we can type “**cd ..**”. The two dots represent back.

```
nayso@Alok-Aspire:~$ cd Downloads
nayso@Alok-Aspire:~/Downloads$ cd
nayso@Alok-Aspire:~$ cd Raspberry\ Pi
nayso@Alok-Aspire:~/Raspberry Pi$ cd ..
nayso@Alok-Aspire:~$
```

4. mkdir & rmdir

The **mkdir** command is used when we need to create a folder or a directory. For Example, if we want to make a directory called “**DIY**”, then we can type “**mkdir DIY**”. Remember, as told before, if we want to create a directory named “**DIY Hacking**”, then we can type “**mkdir DIY\ Hacking**”.

rmdir is the command used for deleting a directory. But, **rmdir** can only be used to delete an empty directory. To delete a directory containing files, **rm** is used.

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ mkdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
DIY
nayso@Alok-Aspire:~/Desktop$ rmdir DIY
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

5. rm

The **rm** command is used to delete files and directories. **rm** cannot simply delete a directory. “**rm -r**” is used to delete a directory. In this case, it deletes both the folder and the files in it.

```
nayso@Alok-Aspire:~/Desktop$ ls
newer.py  New Folder
nayso@Alok-Aspire:~/Desktop$ rm newer.py
nayso@Alok-Aspire:~/Desktop$ ls
New Folder
nayso@Alok-Aspire:~/Desktop$ rm -r New\ Folder
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$
```

6. touch

The **touch** command is used to create a file. It can be anything, from an empty txt file to an empty zip file. For example: “**touch new.txt**”

```
nayso@Alok-Aspire:~/Desktop$ ls
nayso@Alok-Aspire:~/Desktop$ touch new.txt
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
```

7. man & -help

To know more about a command and on how to use it, the **man** command is used. It shows the manual pages of the command. For Example, “**man cd**” shows the manual pages of the **cd** command. Typing in the command name and the argument helps it show which ways the command can be used (Example – **cd -help**).

```
TOUCH(1)                                User Commands                                TOUCH(1)
NAME
    touch - change file timestamps
SYNOPSIS
    touch [OPTION]... FILE...
DESCRIPTION
    Update the access and modification times of each FILE to the current
    time.

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

    A FILE argument string of - is handled specially and causes touch to
    change the times of the file associated with standard output.

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

    -a      change only the access time

Manual page touch(1) line 1 (press h for help or q to quit)
```

8. cp

The **cp** command is used to copy files through the command line. It takes two arguments, the first one is location of the file to be copied, and the second is where to copy.

```
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ cp new.txt /home/nayso/Music/
nayso@Alok-Aspire:~/Desktop$ ls /home/nayso/Music/
new.txt
```

9. mv

The **mv** command is used to move files through the command line. We can also use the **mv** command to rename a file. For example, if we want to rename the file “text” to “new”, we can use “**mv text new**”. It takes the two arguments just like the **cp** command.

```
nayso@Alok-Aspire:~/Desktop$ ls
new.txt
nayso@Alok-Aspire:~/Desktop$ mv new.txt newer.txt
nayso@Alok-Aspire:~/Desktop$ ls
newer.txt
```

10. locate

The **locate** command is used to locate a file in a Linux System, just like the search command in Windows. This command is useful when we don’t know where a file is saved or the actual name of the file. Using the **-i** argument with the command, helps to ignore the case (it doesn’t matter if it is Capital or Small). So, if we want a file that has the word “hello”, it gives the list of all the files in our Linux System containing the word “hello” when we type in “**locate -i hello**”. If we remember two words, we can separate it using asterisk (*). For example, to locate a file containing the words “hello” and “this”, we can use the command “**locate -i *hello*this**”

```
nayso@Alok-Aspire:~$ locate newer.txt
/home/nayso/Desktop/newer.txt
nayso@Alok-Aspire:~$ locate *DIY*Hacking*
/home/nayso/DIY Hacking
```

INTERMEDIATE COMMANDS:

1. echo

“**echo**” is a command that helps us move some data, usually text into a file. For example, if we want to create a new text file or add into an already made text file, then we just need to type in “**echo hello, my name is alok >> new.txt**”. We do not need to separate the spaces by using the backward slash here because we put in two triangular brackets when we finish what we need to write.

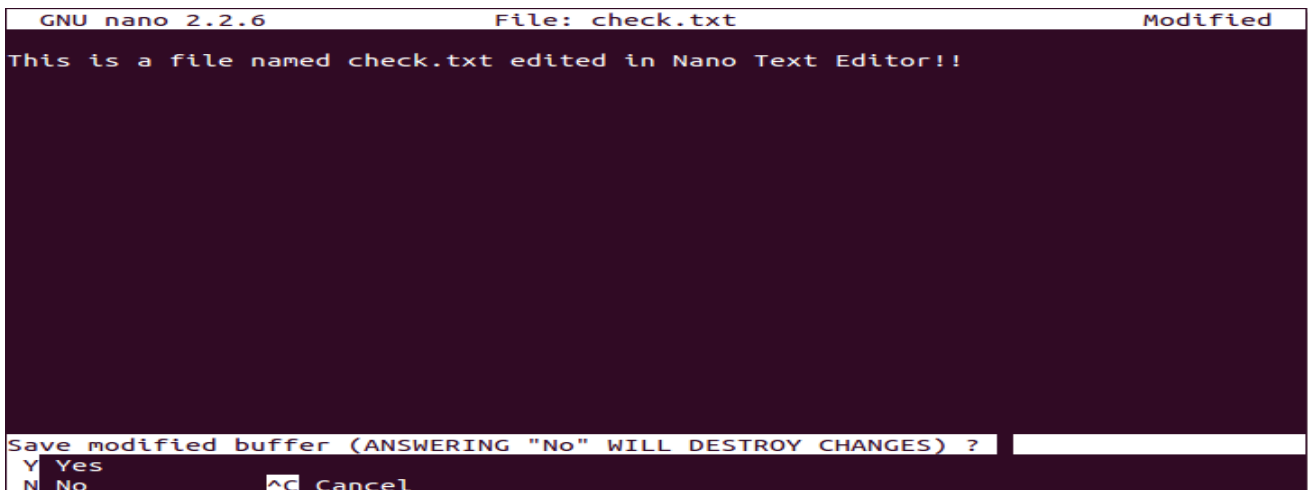
2. cat

The **cat** command is used to display the contents of a file, usually used to easily view programs.

```
nayso@Alok-Aspire:~/Desktop$ echo hello, my name is alok >> new.txt
nayso@Alok-Aspire:~/Desktop$ cat new.txt
hello, my name is alok
nayso@Alok-Aspire:~/Desktop$ echo this is another line >> new.txt
nayso@Alok-Aspire:~/Desktop$ cat new.txt
hello, my name is alok
this is another line
```

3. nano, vi, jed

nano and **vi** are already installed text editors in the Linux command line. **nano** is a good text editor which denotes keywords with color and can recognize most of the languages. **vi** is simpler than **nano**. We can create a new file or modify one using this editor. For example, if we need to make a new file named “**check.txt**”, we can create it by using the command “**nano check.txt**”. We can save our files after editing by using the sequence, Ctrl+X, then Y (or N for no). In my experience, using **nano** for HTML editing doesn’t seem so good, because of its color, so I recommend jed text editor. We will come to installing packages soon.



```
GNU nano 2.2.6           File: check.txt           Modified
This is a file named check.txt edited in Nano Text Editor!!

Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ?
Y Yes
N No      ^C Cancel
```

4. sudo

sudo is a widely used command in the Linux command line. **sudo** stands for “SuperUser Do”. So, if we want any command to be done with administrative or root privileges, then we can use the **sudo** command. For Example, if we want to edit a file like **viz. alsa-base.conf** which needs root permissions, we can use the command – **sudo nano alsa-base.conf**. We can enter the root

command line using the command “**sudo bash**”, then type in our user password. We can also use the command “**su**” to do this, but we need to set a root password before that. For that, we can use the command “**sudo passwd**”(it wasn’t misspelled, it is **passwd**). Then type in the new root password.

```
nayso@Alok-Aspire:~/Desktop$ sudo passwd
[sudo] password for nayso:
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
nayso@Alok-Aspire:~/Desktop$ su
Password:
root@Alok-Aspire:/home/nayso/Desktop#
```

5. df

The **df** command is used to see the available disk space in each of the partitions in our system. We can just type in **df** in the command line and we can see each mounted partition and their used/available space in % and in KBs. If we want it shown in megabytes, we can use the command “**df -m**”

```
root@Alok-Aspire:/home/nayso/Desktop# df -m
Filesystem      1M-blocks  Used Available Use% Mounted on
udev             940         1      940     1% /dev
tmpfs            191         2      189     1% /run
/dev/sda5        96398 23466     68013   26% /
none              1           0         1    0% /sys/fs/cgroup
none              5           0         5    0% /run/lock
none             951         1      950     1% /run/shm
none             100         1      100     1% /run/user
```

6. du

du is a command to know the disk usage of a file in our System. If we want to know the disk usage for a particular folder or file in Linux, then we can type in the command **df** and the name of the folder or file. For example, if we want to know the disk space used by the folder Documents in Linux, we can use the command “**du Documents**”. We can also use the command “**ls -lah**” to view the file sizes of all the files in a folder.

```
nayso@Alok-Aspire:~$ du Documents
516    Documents/DIYHacking
548    Documents
```

7. tar

tar is a command used to work with tarballs (or files compressed in a tarball archive) in the Linux Command Line. It has a long list of uses. It can be used to compress and uncompress different types of **tar** archives like **.tar**, **.tar.gz**, **.tar.bz2**.etc. It works on the basis of the arguments given to it. For example, **tar -cvf** for creating a **.tar** archive, **-xvf** to untar a tar archive, **-tvf** to list the contents of the archive.etc.

8. zip, unzip

zip is a command used to compress files into a zip archive, **unzip** is used to extract files from a zip archive.

9. uname

uname is a command used to show the Information about the system our Linux distro is running. Using the command “**uname -a**” prints most of the information about the system. This prints the Kernel release date, version, processor type, etc.

```
nayso@Alok-Aspire:~$ uname -a
Linux Alok-Aspire 4.4.0-22-generic #40~14.04.1-Ubuntu SMP Fri May 13 17:27:18 UTC
2016 i686 i686 i686 GNU/Linux
```

10. apt-get

apt is a command used to work with packages in the Linux command line. **apt-get** is a command used to install packages. This requires root privileges, so we use the **sudo** command with it. For example, if we want to install the text editor jed (as mentioned earlier), we can type in the command “**sudo apt-get install jed**”. Similarly, any packages can be installed like this. It is good to update our repository each time we try to install a new package. We can do that by typing “**sudo apt-get update**”. We can upgrade the system by typing “**sudo apt-get upgrade**”. We can also upgrade the distro by typing “**sudo apt-get dist-upgrade**”. The command “**apt-cache search**” is used to search for a package. If we want to search for one, we can type in “**apt-cache search jed**”(This doesn’t require root).

```
nayso@Alok-Aspire:~$ sudo apt-get install jed
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  jed-common libslang2-modules slsh
Suggested packages:
  gpm
The following NEW packages will be installed:
  jed jed-common libslang2-modules slsh
0 upgraded, 4 newly installed, 0 to remove and 419 not upgraded.
Need to get 810 kB of archives.
After this operation, 2,992 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

11. chmod

chmod is the command used to make a file executable and to change the permissions granted to it in Linux. Imagine we have a python code named **numbers.py** in our computer, we’ll need to run “**python numbers.py**” every time we need to run it. Instead of that, when we make it executable, we’ll just need to run “**numbers.py**” in the terminal to run the file. To make a file executable, we can use the command “**chmod +x numbers.py**” in this case. We can use “**chmod 755 numbers.py**” to give it root permissions or “**sudo chmod +x numbers.py**” for root executable.


```
nayso@Alok-Aspire:~/Desktop$ ls
numbers.py
nayso@Alok-Aspire:~/Desktop$ chmod +x numbers.py
nayso@Alok-Aspire:~/Desktop$ ls
numbers.py
```

12.hostname

hostname is a command used to know our name in our host or our network. Basically, it displays our hostname and IP address. Typing just “**hostname**” gives the output, our hostname. Typing in “**hostname -I**” gives us our IP address in our network.

```
nayso@Alok-Aspire:~/Desktop$ hostname
Alok-Aspire
nayso@Alok-Aspire:~/Desktop$ hostname -I
192.168.1.36
```

13.ping

ping is a command used to check our connection to a server. Wikipedia says that “**Ping** is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network”. Simply, when we type in, for example, “**ping google.com**”, it checks if it can connect to the server and come back. It measures this round-trip time and gives us the details about it. The use of this command for simple users like us is to check our internet connection. If it pings the Google server (in this case), we can confirm that our internet connection is active!

```
nayso@Alok-Aspire:~/Desktop$ ping google.com
PING google.com (172.217.26.206) 56(84) bytes of data.
64 bytes from google.com (172.217.26.206): icmp_seq=1 ttl=56 time=51.2 ms
64 bytes from google.com (172.217.26.206): icmp_seq=2 ttl=56 time=47.9 ms
64 bytes from google.com (172.217.26.206): icmp_seq=3 ttl=56 time=48.9 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2000ms
rtt min/avg/max/mdev = 47.959/49.388/51.299/1.417 ms
```

SOME MORE COMMANDS:

1. Aspell

GNU Aspell is a free and open source spell checker designed to replace Ispell. It can either be used as a library or as an independent spell checker.

2. chown

Chown changes file or group ownership and has the option to change ownership of all objects within a directory tree, as well as having the ability to view information on objects processed.

3. cmp

The cmp utility compares two files of any type and writes the results to the standard output. By default, cmp is silent if the files are the same; if they differ, the byte and line number at which the first difference occurred is reported.

4. comm

Comm compares lines common to file1 and file2. The output is in three columns; from left to right: lines unique to file1, lines unique to file2 and lines common to both files.

5. cpio

Cpio copies files into or out of a cpio or tar archive. A tar archive is a file that contains other files, plus information about them, such as their file name, owner, timestamps and access permissions. The archive can be another file on the disk, a magnetic tape or a pipe. Cpio has three operating modes and is a more efficient alternative to tar.

6. CRON

CRON is a Linux system process that will execute a program at a preset time. To use CRON, a user must prepare a text file that describes the program to be executed and the times at which CRON should execute them. Then the cron tab program can be used to load the text file that describes the CRON jobs into CRON.

7. date

Date sets a system's date and time. This is also a useful way to output/print current information when working in a script file.

8. declare

Declare declares variables, gives them attributes or modifies properties of variables.

9. enable

Enable will stop or start printers or classes.

10. env

Env runs a program in a modified environment or displays the current environment and its variables.

11. eval

Eval evaluates several arguments, concatenates them into a single command and then reports on that argument's status.

12. exec

Exec replaces the parent process with whatever command is typed. This command treats its arguments as the specification of one or more sub-processes to execute.

13. exit

The exit command terminates a script and can return a value to the parent script.

14.expect

Expect talks to other interactive programs according to a script and waits for a response, often from any string that matches a given pattern.

15.export

Export converts a file into a different format than the one in which it is currently. Once a file is exported, it can be accessed by any application that uses its format.

16.find

Find searches the directory tree to find particular groups of files that meet specified conditions, including --name and --type, -exec and --size and --mtime and --user.

17.for, while

For and while are used to execute or loop items repeatedly as long as certain conditions are met.

18.free

Free displays the total amount of free and used physical memory and swap space in the system, as well as the buffers and cache used by the kernel.

19.grep

Grep searches files for a given character string or pattern and can replace the string with another. This is one method of searching for files within Linux.

20.gzip

Gzip is the GNU project's open source program used for file compression, compressing web pages on the server end for decompression in the browser. This is popular for streaming media compression and can concatenate and compress several streams simultaneously.

21.ifconfig

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

22.ifup

Ifup configures a network interface/enables a network connection.

23.ifdown

Ifdown shuts down a network interface/disables a network connection.

24.less, more

The less command lets an admin scroll through configuration and error log files, displaying text files one screen at a time, with backward or forward moving available in files. There is more mobility within files. Similar to less, more pages through text one screen at a time, but is more limited in moving in files.

25.lft

Lft is similar to traceroute in determining connection routes, but provides a lot more information for debugging connections or finding where a box/system is. Lft also displays route packets and file types.

26.ln

The ln command creates a new name for a file through hard linking, allowing multiple users to share one file.

27.mc

A visual shell, text-based file manager for UNIX systems.

28.neat

Neat is a GNOME GUI admin tool which allows admins to specify the information needed to set up a network card, among other features. Setting up an NTL Cable Modem using neat.

29.netconfig/netcfg

Netconfig configures a network, enables network products and displays a series of screens that ask for configuration information.

30.netstat

Netstat provides information and statistics about protocols in use and current TCP/IP network connections. It is a helpful forensic tool in figuring out which processes and programs are active on a computer and are involved in network communications.

31.nslookup

Nslookup allows a user to enter a host name and find the corresponding IP address. A reverse of that process to find the host name is also possible.

32.od

Od is used to dump binary files in octal (or hex/binary) format to standard output.

33.passwd

Passwd updates a user's authentication tokens (changes their current password).

34.read

Read is used to read lines of text from standard input and to assign values of each field in the input line to shell variables for further processing.

35.RPM

Red Hat Package Manager (RPM) is a command-line-driven program capable of installing, uninstalling and managing software packages in Linux.

36.rsync

Rsync syncs data from one disk or file to another across a network connection. Rsync is similar to rcp, but has more options.

37.screen

The GNU screen utility is a terminal multiplexor in which a user can use a single terminal window to run multiple terminal applications or windows.

38.sdiff

Sdiff finds differences between two files by producing a side-by-side listing indicating lines that are dissimilar. Sdiff then merges the files and outputs results to the outfile.

39.sed

Sed is a stream editor that is used to filter text in a pipeline, distinguishing it from other editors. Sed takes text input, performs operations on it and outputs the modified text. Sed is typically used to extract part of a file using pattern matching or to substitute multiple occurrences of a string within a file.

40.shutdown

Shutdown is a command that turns off the computer and that can be combined with variables such as -h, for halt after shutdown, or -r, for reboot after shutdown.

41.Snort

Snort is an open source network intrusion detection system and packet sniffer that monitors network traffic, looking at each packet to detect dangerous payloads or suspicious anomalies. Snort is based on libpcap. Stopping hackers with Snort.

42.sort

Used to sort lines of text alphabetically or numerically according to fields; multiple sort keys can also be used.

43.SSH

SSH is a command interface used for securely gaining access to a remote computer and is used by network admins to control servers remotely.

44.TOP

TOP is a set of protocols for networks that performs distributed information processing in offices and displays the tasks on the system that take up the most memory. TOP can sort tasks by CPU usage, memory usage and runtime.

45.tr

Tr is used to translate or delete characters from a text stream. Tr writes to standard output, but does not accept file names as arguments, it only accepts inputs from standard input.

46.traceroute

Traceroute determines and records a route through the internet between two computers and is useful for troubleshooting network/router issues. If the domain does not work or is not available, an IP can be tracerouted.

47.uniq

Uniq compares adjacent lines in a file and removes/reports any duplicate lines.

48.vmstat

Vmstat is used to get a snapshot of everything in a system and to report information on such items as processes, memory, paging and CPU activity. This is a good method for admins to use to determine where issues/slowdown in a system may be occurring.

49.wc

Wc counts the number of words, lines and characters in text files and produces a count for multiple files if several files are selected.

50.wget

Wget is a network utility that retrieves files from the web that support http, https and ftp protocols. Wget works non-interactively in the background while a user is logged off. This can create local versions of remote websites, re-creating directories of original sites.

51.whoami

Whoami prints or writes the user/login name associated with the current user ID to the standard output.

52.xargs

Xargs reads, builds and executes arguments from standard input; blank lines in the input are ignored.

Tips and Tricks in Using Linux Command Line:

- ☑ We can use **clear** command to clear the terminal if it gets filled up with too many commands!
- ☑ **TAB** can be used to fill up in Terminal. For example, we just need to type "**cd Doc**" and then **TAB** and the terminal fills the rest up and makes it "**cd Documents**".
- ☑ **Ctrl+C** can be used to stop any command in terminal safely. If it doesn't stop with that, then **Ctrl+Z** can be used to force stop it.
- ☑ We can exit from the terminal by using the **exit** command.
- ☑ We can power off or reboot the computer by using the command **sudo halt** and **sudo reboot**.

SHELL:

Shell is a UNIX term for the interactive user interface with an operating system. The shell is a user program or it is an environment provided for user interaction. It is a command language interpreter that executes commands read from the standard input device such as keyboard or from a file.

The shell gets started when we log in or open a console (terminal). It is a quick and dirty way to execute utilities. A shell usually implies an interface with a command syntax (think of the DOS operating system and its "C:>" prompts and user commands such as "dir" and "edit"). The shell is not part of system kernel, but uses the system kernel to execute programs, create files etc.

Several shells are available for Linux including:

1. Bourne Shell:

The original Bourne shell is named after its developer at Bell Labs, Stephen Bourne. It was the first shell used for the UNIX operating system, and it has been largely surpassed in functionality by many of the more recent shells. However, all UNIX and many Linux versions allow users to switch to the original Bourne Shell, known simply as "sh," if they choose to forgo features such as file name completion and command histories that later shells have added.

2. C Shell:

The C shell, as its name might imply, was designed to allow users to write shell script programs using a syntax very similar to that of the C programming language. **It is known as "csh."**

3. TC Shell:

TC shell is an expansion upon the C shell. It has all the same features, but adds the ability to use keystrokes from the Emacs word processor program to edit text on the command line. For example, users can press Esc-D to delete the rest of the highlighted word. **It is also known as "tcsh."**

4. Korn Shell:

Korn Shell was also written by a developer at Bell Labs, David Korn. It attempts to merge the features of the C shell, TC shell and Bourne shell under one package. It also includes the ability for developers to create new shell commands as the need arises. **It is known as "ksh."**

5. Bourne-Again Shell:

The Bourne-Again shell is an updated version of the original Bourne shell that was created by the Free Software Foundation for its open source GNU project. For this reason, it is a widely used shell in the open source community.

Its syntax is similar to that used by the Bourne shell, however it incorporates some of the more advanced features found in the C, TC and Korn shells.

Among the added features that Bourne lacked are the ability to complete file names by pressing the TAB key, the ability to remember a history of recent commands and the ability to run multiple programs in the background at once. **It is known as "bash."**

TEXT EDITOR VI AND PICO:

➤ **Vi:**

Vi is often the default editor that pops up when we're ready to write an e-mail message or when we're posting a News message. Vi is complicated and seems difficult to learn at first. However, it is often the default for UNIX and Linux systems.

Start the Vi editor by typing `vi` at the prompt. Typing `vi` followed by a file name will automatically name the file so we don't have to worry about it later.

On a number of other occasions programs use Vi for text editing. When we post to a News group or send e-mail, the system may default to Vi. How do we know when we're in Vi and when we can use Vi commands?

Vi has two modes:

Command mode that lets us use commands to edit, save, or quit; and **Text mode** that lets us type. If we attempt to do something in the wrong mode, the system beeps furiously at us until we either stop pressing keys or scream (the louder the scream the more beeps we muffle). Use the **Esc key** to change from one mode to the other.

Simple Example of Using Vi:

Vi starts in the Command mode. To switch to text mode press `i`. Type out the text. To make corrections, move to a location in the text, or save the file, switch to the Command mode by pressing `Esc`. In command mode we may edit, save, or exit (see Command Mode below for details). To switch back to Text mode, type `i` again. Getting the hang of switching between modes may take a while, so be patient.

Occasionally, when we're typing quickly, some of our text may seem to disappear. Actually our text is still there, but has become blacked out. This black-out is usually caused by a slow screen update, and Vi is notorious for this. Because the information on our screen is coming through our modem, it needs to be updated (refreshed) occasionally, or parts of the information disappear from our screen. To update our screen's information, hold down the `Ctrl` key and press `l` (that's an L not a 1) while in the Command mode. Do this whenever chunks of our text black-out.

Text Writing Mode:

The Vi editor starts in the Command mode. To switch to the Text mode and begin typing, press `i`. If we hear several beeps and we're unable to type, then press `i` twice to switch to the text mode.

Command Mode:

Press the `Esc` key to switch from Text mode to Command mode.

Moving Around:

<i>To move</i>	<i>press</i>
one character <i>left</i>	h
one character <i>right</i>	l
<i>up</i> one line	k
<i>down</i> one line	j
<i>To</i>	<i>hold down</i>
move <i>back</i> one screen	Ctrl and press b
move <i>forward</i> one screen	Ctrl and press f

Saving and Exiting:

- ❖ quit Vi without saving anything (we'll lose any changes we made when using this command) type: **:q!**
- ❖ save/write the file we're working on without exiting type: **:w** followed by a filename
- ❖ save/write the file and quit the vi editor in one step by typing: **:wq**

Editing:

<i>To</i>	<i>type</i>
<i>cut</i> the <i>character</i> directly above the cursor. Ex: test The letter e will be cut.	x
<i>cut</i> the <i>entire line</i> directly above the cursor.	dd
<i>paste</i> the last character or line that you cut on the line directly <i>below</i> the cursor.	p
<i>paste</i> the last character or line that you cut on the line directly <i>above</i> the cursor.	P
<i>search forward</i> through the document for specific text you designate. If Vi finds the text you specified, it will place the cursor below the first letter of the text. If Vi can not find the text, the message <i>Pattern not found</i> appears. Ex: /text string	/
<i>search</i> for the <i>same</i> text again (saves you the time of having to re-type your text string).	n
<i>search backward</i> through the document for specific text you designate. If Vi finds the text you specified it will place the cursor below the first letter of the text. If Vi can not find the text the message <i>Pattern not found</i> appears. Ex: ?text string	?

Vi Troubleshooting:

Trying to use a command if we're trying to use a Move, Save, or Edit command but the command isn't working, switch from Text mode to Command mode by pressing the Esc key. If the backspace key doesn't work, then hold down the Ctrl key and press backspace. Trying to save If we get the message No current filename, type :w followed by a filename. The message appears

only if a filename has not been specified. If we get the message is a directory, we're trying to write to a directory not a file. Use a different name for the file to save it properly.

➤ **Pico:**

Pico is a fairly simple text editor that provides straight-forward options and easy-to-use commands. Although some programmers have frowned at Pico's simplicity and limited options, most folks find that it provides everything necessary to write long documents with minimal hassles. However, Pico is not very good when manipulating certain types of files such as making changes to .cgi files etc.

Start the Pico editor by typing `pico` at the prompt. Typing `pico` followed by a file name automatically names the file so we don't have to worry about it later. Example: `pico newfile.txt`

Using Pico is fairly straight-forward. The blinking cursor indicates where we may begin typing. We type out our message without worrying about line breaks or page breaks. Pico takes care of these for us. When we're finished typing, or anytime we're ready to use a Pico command, refer to the Pico menu options, listed at the bottom of the screen.

To use an option, hold down the Ctrl key and press the letter indicated. The ^ symbol represents the Ctrl key. Example: To use the option ^G Get Help, hold down the Ctrl key (designated by the ^ character) and press g. Always refer to the bottom two lines of Pico to see what options are available to us. Depending on what we're doing in Pico, our options change.

Editing text:

We can edit our document by using the arrow keys and the backspace key on our keyboard. Sometimes, when we're typing quickly, our text may seem to disappear. Our text is still there, but has become blacked out. This black-out is usually caused by a slow screen update. Because the information on our screen is coming through our modem, it needs to be updated (refreshed) occasionally, or parts of the information disappear from our screen. To update our screen's information, hold down the Ctrl key and press l (that's L, not the number 1). Do this whenever chunks of text we're working on become blacked out.

Movement Commands:

Depending on our system, the arrow keys or the backspace key may not work. Instead, we can use these commands to perform the same tasks.

<i>To</i>	<i>hold down the Ctrl key and press</i>	<i>instead of</i>
<i>delete a character</i>	backspace	backspace
<i>move up a line</i>	p	up arrow
<i>move down a line</i>	n	down arrow
<i>move left one space</i>	b	left arrow
<i>move right one space</i>	f	right arrow
<i>move to the end of a line</i>	e	End

Smorgasbord of Pico Options

^C Cancel allows us to stop a process at any time. If we make a mistake, just hold down the Ctrl key and press c.

^G Get Help Get clear and concise assistance from the Pico help, in case something unexpected happens or we need additional information about a command.

^X Exit Exit Pico at any time. If we've made changes to a file or we've worked on a new file, but we haven't saved the changes, we see this message: Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) (y/n)? Answering no (press n) will close Pico and bring us back to the prompt without saving our file. Answering yes (press y) will allow us to save the file we've been working on (see WriteOut section below for details).

^O WriteOut Save our file without hassles or worries. Fill in the name of the file beside the File Name to write: prompt. If our file already has a name, then press enter.

^T To Files option lets us save our text over a file that exists in our directory. By choosing the To Files option, Pico takes us to a directory Browser.

Browser Options:

To alter a file or directory, first use the arrow keys or the optional movement keys to highlight a particular name. We can also press w to find and highlight a file or directory quickly. Once we've highlighted a particular file or directory, we can use any one of these options.

- Type e to Exit the Browser.
- Type r to rename a directory or file.
- Type d to delete a file.
- Type m to create an additional copy of a file.
- Type g to move to another directory where the file is located.
- Type s or press to write over the file with text we just wrote in Pico.

^R Read File Insert text from another file into our current text file. This option allows us to search through our directories for a file that we would like to add to our text. This option is especially handy if we've saved a document and would like to add its content to the new file we're working on. Text from the file we select is placed on the line directly above our cursor. At the Insert file prompt we may either type a file name or use the Browser options.

^T To Files option lets us import a text file directly into the file we're currently typing. By choosing the To Files option, Pico takes us to a directory Browser.

^Y Prev Pg Move quickly to the previous page. Although we could just as easily press the up arrow key several times, this command quickly jumps our cursor up one page.

^V Next Pg Move quickly to the next page. Although we could just as easily press the down arrow key several times, this command quickly jumps our cursor down one page.

^K Cut Text Cut a line of text. This option allows us to cut a full line of text. By using the uncut command and our arrow keys, we can then paste the cut text at another location in our document. To cut specific text in a line or to cut several lines of text, first select the text.

Selecting Text:

To select text for cutting and pasting use the following steps:

- Move the cursor to the beginning of the text we want to select.
- Hold down the Ctrl key and press ^
- Use the right arrow key or hold down Ctrl and press f to highlight text. When we have highlighted the appropriate text, hold down the Ctrl key and press k to cut it. Paste the text we cut, anywhere in our document, using UnCut Text

^U UnCut Text Paste text that we previously cut. We can use this option to undo an accidental cut of text or place cut text at another location in our document. The text we cut is pasted on the line directly above our cursor.

^C Cur Pos Indicate the current position of our cursor, relative to the entire document. This is a helpful option if we'd like to check exactly where in our document we are. The status line indicates the following items: [line 8 of 18 (44%), character 109 of 254 (42%)]

^J Justify Even out lines of text. This command is handy when we accidentally type extra spaces between words or press the key before reaching the end of a line. The option evens the length of our text lines automatically.

^U UnJustify UnJustify lines of text. For the messy line look we can always select the UnJustify option.

^W Where is Find a particular string of text quickly. This option allows us to do a word search in our text. This option is especially handy for longer documents. If the word we designated at the Search: prompt is found, it places the cursor beside it.

^T To Spell Check for spelling errors. The spell check option allows us to correct spelling errors throughout our document. If spell checker finds a misspelled word or a word it doesn't recognize (don't worry, this rarely happens), it will let us correct the word. At the Edit a replacement: prompt, type in the correct spelling of a word. However, if we don't want to make any changes, simply press the enter key.

Any words that we've corrected but re-occur in the document can be automatically replaced. At the Replace a with b? [y]: prompt press y to replace all occurrences of the misspelled word or n to ignore.

FILE SYSTEM OF LINUX:

File systems organize the data stored on computer hard drives, keeping track of the physical locations of all data elements on disk while allowing users to quickly and reliably retrieve files when needed.

The file system acts as a digital index that lets a computer instantly find a specific file, regardless of the size or configuration of the storage drive or where the data bytes associated with the file sit on the drive's storage platters.

Every operating system, from MS-DOS to Windows 95, Windows XP and Linux, has its own file system. But although all file systems perform the same basic functions, they vary in design and sophistication.

VARIOUS FILE SYSTEMS:

1. FAT (File Allocation Table):

Introductions:

This file system is one of most simple types of file systems. It consists of file system descriptor sector (boot sector or superblock), file system block allocation table (referenced as File Allocation Table) and plain storage space to store files and folders. The files on FAT are stored in directories. Each directory is the array of 32-byte records, each defines file or file extended attributes (like long file name). File record references a first block of file. Any next block could be found through block allocation table by using it as linked-list.

Block allocation table contains array of block descriptors. Zero value indicates block is not used and non-zero indicates reference to next block of the file or special value for end of file. The number in FAT12, FAT16, and FAT32 file system name means how many bits are used to number file system block. This means that FAT12 may use up to 4096 different block references, FAT16 - 65536 and FAT32 - 4294967296. Actual maximum count of blocks is even less and depends on file system driver implementation.

Version of FAT file system:

- FAT 12: The initial version of FAT introduced in 1977. Primary file system for Microsoft System upto MS-DOS 4.0
- FAT 16: Introduced in 1988, primary file system for MS-DOS 4.0 upto Windows 95. Support drive size upto 2 GB.
- FAT 32: Latest version of FAT, introduced in 1996 for windows 95 OSRL Users. Support drive size upto 8TB.

Applications:

FAT12 was used for old floppy disks. FAT16 (or simply FAT) and FAT32 are widely used for flash memory cards, USB flash sticks and so on. It is supported by mobile phones, digital cameras and other portable devices. FAT or FAT32 could be identified as file system, used on Windows compatible external storages or disk partitions with size below 2GB (for FAT) or 32GB (for FAT32). Windows cannot create even FAT32 file system over 32GB (however Linux supports FAT32 up to 2TB).

2. NTFS (New Technology File System):

Introductions:

It was introduced in Windows NT and at present is main file system for Windows. It is default file system for disk partitions and the only one file system that is supported for disk partitions over 32GB. The file system is quite extensible and supports many file properties, including access control, encryption etc. Each file on NTFS is stored as file descriptor in Master File Table and file content. Master file table contains all information about file: size, allocation, name and so on. The first and the last sectors of the file system contain file system settings (the boot record or

superblock). The file system uses 48 and 64 bit values to reference files thus it supports quite large disk storages.

Features:

- It uses 64-bit disk addresses and can support disk partitions up to 264 bytes.
- Individual file names in NTFS are limited to 255 characters. Case sensitive names.
- Encryption & Data recovery.
- Compression.
- File level security.
- FILE ENCRYPTION: Encrypting file system is used to encrypt files in NTFS. Generally Public Key cryptography is used
- **Data Recovery:** It offers data recovery mechanism.
- File Compression: NTFS can perform data compression on individual files or on all data files in a directory.
- **Security:** NTFS allows file level security. With NTFS permissions, one can control which users have what kind of access to which files. Security can be assigned at two different levels: Per user basis and On a group basis Network File System (NFS)

3. Network File System (protocol)

Introduction:

Network File System (NFS) is a network file system protocol originally developed by Sun Microsystems in 1984, allowing a user on a client computer to access files over a network in a manner similar to how local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. The Network File System is an open standard defined in RFCs, allowing anyone to implement the protocol.

NFS Protocols:

NFS accomplishes two client-server protocol:

- The Mount Protocol: It handles mounting.
- The NFS Protocols: Which is for directory & file access. Daemons NFS server daemons (nfsd) Accept RPC calls from client.

4. The Extended File System (Ext)

Introduction:

The extended file system (ext), was released in April 1992 as the first file system using the VFS API and was included in Linux version 0.96c.

Features:

- allowed 2 gigabytes of data
- filenames of up to 255 characters
- Limitation of Ext: There was no support for separate access i-node modification and data modification timestamps.

Solution:

A new file systems were developed in January 1993 by Rémy Card.

5. The Second Extended File System (Ext2)

Introduction:

The Second Extended File system was devised as an extensible and powerful file system for Linux. It is also the most successful file system so far in the Linux community and is the basis for all of the currently shipping Linux distributions.

- Ext2 data structures
- Physical Layout of the EXT2 File system
- i-node structure of ext-2:

Features of Ext2:

- POSIX, ACL and extended attribute were first introduced.
- Journaling not allowed with flash drives.

Disadvantages:

- Limit of sublevel directory 32768.
- Cannot handle file larger than 2TB.
- Block size is limited by architecture.

6. The Ext3 Or Third Extended File System

Introduction:

The **ext3** or **third extended file system** is a journaled file system that is commonly used by the Linux kernel. It is the default file system for many popular Linux distributions, including Debian. Stephen Tweedie first revealed that he was working on extending ext2 in *Journaling the Linux ext2fs File system* in a 1998 paper and later in a February 1999 kernel mailing list posting, and the file system was merged with the mainline Linux kernel in November 2001 from 2.4.15 onward.

Disadvantages:

- Functionality
- Defragmentation
- Compression
- No check summing in journal.

7. The Ext4 Or Fourth Extended File System:

Introduction:

Ext4 is the evolution of the most used Linux file system Ext3. In many ways, Ext4 is a deeper improvement over Ext3 than Ext3 was over Ext2. Ext3 was mostly about adding journaling to Ext2, but Ext4 modifies important data structures of the file system such as the ones destined to store the file data. The result is a file system with an improved design, better performance, reliability, and features.

Features:

- Compatibility (the file system can continue to be mounted as ext3). This allows users to still read the file system from other distributions/operating systems without ext4 support (e.g. Windows with ext3 drivers)
- Improved performance (though not as much as a fully-converted ext4 partition)

8. ReiserFS:

Introduction:

The alternative Linux file system with the main purpose to store huge amount of small files. It has good capability of files search and it allows to 'compact' files allocation by storing file tails or small file along with metadata and to not use large file system blocks for this.

9. Some Other File Systems:

XFS: The file system from SGI Company who initially used it for their IRIX servers. Now XFS specifications are open it the file system support was implemented in Linux. The XFS file system has great performance and thus widely used as file storage file system.

JFS: The file system was developed by IBM for their powerful computing systems. Saying JFS one usually mean JFS, second edition (JFS2). Currently this file system is open source and is implemented in most modern Linux distributions.

UFS: The most common file system for these OS is UFS (the Unix File System). It is also often called FFS (the Fast File System; it is 'fast' in comparison with a previous file system used for UNIX). The UFS is the source of ideas for many other file system implementations. Currently UFS (in different editions) is supported by all Unix-family OS and is the main file system of BSD OS and Sun Solaris OS. The modern tendency is to implement replacements for UFS in different OS (ZFS for Solaris, JFS and derived file systems for UNIX and so on).

Clustered File Systems: The clustered file systems specifics is that they are used in computer cluster systems. These file systems have embedded support of distributed storage.

- **ZFS:** Sun Company 'Zettabyte File System' - the new file system developed for distributed storages of Sun Solaris OS.
- **Apple Xsan:** The Apple company evolution of CentraVision and later StorNext file systems.
- **VMFS:** The 'Virtual Machine File System' developed by VMware Company for its VMware ESX Server.
- **GFS:** The Red Hat Linux 'Global File System'.
- **JFS1:** The original (legacy) design of IBM JFS file system used in older AIX storage systems.

DIRECTORIES AND THEIR SPECIAL PURPOSE:

The directory structure of Linux/other Unix-like systems is very intimidating for the new user, especially if he/she is migrating from Windows. In Windows, almost all programs install their files (all files) in the directory named: 'Program Files.' Such is not the case in Linux. The directory

system categories all installed files. All configuration files are in /etc, all binary files are in /bin or /usr/bin or /usr/local/bin. Here is the entire directory structure along with what they contain:

/ (Root):

- Every single file and directory starts from the root directory.
- Only root user has write privilege under this directory.
- Please note that /root is root user's home directory, which is not same as /.

/bin (User Binaries):

- Contains binary executable.
- Common Linux commands we need to use in single-user modes are located under this directory.
- Commands used by all the users of the system are located here.
- For example: ps, ls, ping, grep, cp.

/sbin (System Binaries):

- Just like /bin, /sbin also contains binary executable.
- But, the Linux commands located under this directory are used typically by system administrator, for system maintenance purpose.
- For example: iptables, reboot, fdisk, ifconfig, swapon

/etc (Configuration Files):

- Contains configuration files required by all programs.
- This also contains startup and shutdown shell scripts used to start/stop individual programs.
- For example: /etc/resolv.conf, /etc/logrotate.conf

/dev (Device Files):

- Contains device files.
- These include terminal devices, usb, or any device attached to the system.
- For example: /dev/tty1, /dev/usbmon0

/proc (Process Information):

- Contains information about system process.
- This is a pseudo file system contains information about running process. For example: /proc/{pid} directory contains information about the process with that particular pid.
- This is a virtual file system with text information about system resources. For example: /proc/uptime

/var (Variable Files):

- var stands for variable files.
- Content of the files that are expected to grow can be found under this directory.

- This includes system log files (/var/log); packages and database files (/var/lib); emails (/var/mail); print queues (/var/spool); lock files (/var/lock); temp files needed across reboots (/var/tmp);

/tmp (Temporary Files):

- Directory that contains temporary files created by system and users.
- Files under this directory are deleted when system is rebooted.

/usr (User Programs):

- Contains binaries, libraries, documentation, and source-code for second level programs.
- /usr/bin contains binary files for user programs. If we can't find a user binary under /bin, look under /usr/bin. For example: at, awk, cc, less, scp
- /usr/sbin contains binary files for system administrators. If we can't find a system binary under /sbin, look under /usr/sbin. For example: atd, cron, sshd, useradd, userdel
- /usr/lib contains libraries for /usr/bin and /usr/sbin
- /usr/local contains users programs that we install from source. For example, when we install apache from source, it goes under /usr/local/apache2

/home (Home Directories):

- Home directories for all users to store their personal files.
- For example: /home/john, /home/nikita

/boot (Boot Loader Files):

- Contains boot loader related files.
- Kernel initrd, vmlinuz, grub files are located under /boot
- For example: initrd.img-2.6.32-24-generic, vmlinuz-2.6.32-24-generic

/lib (System Libraries):

- Contains library files that supports the binaries located under /bin and /sbin
- Library filenames are either ld* or lib*.so.*
- For example: ld-2.11.1.so, libncurses.so.5.7

/opt (Optional add-on Applications):

- opt stands for optional.
- Contains add-on applications from individual vendors.
- add-on applications should be installed under either /opt/ or /opt/ sub-directory.

/mnt (Mount Directory):

- Temporary mount directory where sysadmins can mount file systems.

/media (Removable Media Devices):

- Temporary mount directory for removable devices.
- For examples, /media/cdrom for CD-ROM; /media/floppy for floppy drives; /media/cdrecorder for CD writer

/srv (Service Data):

- srv stands for service.
- Contains server specific services related data.
- For example, /srv/cvs contains CVS related data.

