On Unix, device drivers for hardware (such as hard disk drives) and special device files (such as /dev/zero and /dev/random) appear in the file system just like normal files.

dd can also read and/or write from/to these files

To backup an entire copy of a hard disk to another hard disk connected to the same system, execute the dd command as shown. In this dd command example, the UNIX device name of the source hard disk is /dev/hda, and device name of the target hard disk is /dev/hdb.

# dd if = /dev/sda of = /dev/sdb

If you give the parameter “conv=noerror” then it will continue to copy if there are read errors.

sync option allows you to copy everything using synchronized I/O.

# dd if = /dev/sda of = /dev/sdb conv=noerror, sync

ifconfig:

ifconfig is used to configure, or view the configuration of, a network interface.

ere, eth0, lo and wlan0 are the names of the active network interfaces on the system.

eth0 is the first Ethernet interface. (Additional Ethernet interfaces would be named eth1, eth2, etc.)

lo is the loopback interface. This is a special network interface that the system uses to communicate with itself.

wlan0 is the name of the first wireless network interface on the system. Additional wireless interfaces would be named wlan1, wlan2, etc

ifconfig -a

This will produce output similar to running ifconfig, but if there are any inactive interfaces on the system, their configuration will also be displayed.

Enabling and disabling an interface

When a network interface is active, it can send and receive data; when it is inactive, it is not able to transmit or receiveEnabling and disabling an interface

When a network interface is active, it can send and receive data; when it is inactive, it is not able to transmit or receive

sudo ifconfig eth1 up

sudo ifconfig eth1 down

modprobe:

modprobe utility is used to add loadable modules to the Linux kernel. You can also view and remove modules using modprobe command.

modprobe -l will display all available modules as shown below.

$ modprobe -l | less

In order to insert a new module into the kernel, execute the modprobe command with the module name.

$ sudo modprobe vmhgfs

If you like to know the full file location of a specific Linux kernel module, use modprobe command and do a grep of the module name as shown below.

$ modprobe | grep vmhgfs

If for some strange reasons, the module name you are trying to load into the kernel is getting used (with the same name) by a different module, then you can load the new module using a different name.

To load a module with a different name, use the modprobe option -o as shown below.

$ sudo modprobe vmhgfs -o vm\_hgfs

Remove the Currently Loaded Module

modprobe -r vmhgfs

insmod command in Linux systems is used to insert modules into the kernel

This is used to view the version of the insmod command.

insmod -V

sudo insmod /home/mukkesh/Desktop/geeksforgeeks.ko

LKM files can have parameters/arguments passed to them through the insmod command. These arguments essentially act as inputs for the LKMs. Depending on how the LKM is written, the arguments are used.

sudo insmod geeks4geeks.ko user="Mukkesh"

rmmod:

rmmod command in Linux system is used to remove a module from the kernel.

rmmod bluetooth

rmmod -v: This option prints messages about what the program is being doing. Usually rmmod only prints messages only if something went wrong.

Example:

rmmod -v bluetooth

This option can be extremely dangerous. It takes no effect unless CONFIG\_MODULE\_FORCE\_UNLOAD is being set when the kernel was compiled. With this option, you can remove the specified modules which are being used, or which are not being designed to be removed or have been marked as not safe.

Example:

sudo rmmod -f bluetooth

This option is going to send errors to syslog instead of standard error.

Example:

rmmod -s bluetooth

lsmod:

smod command is used to display the status of modules in the Linux kernel

lsmod

grep:

This is the basic usage of grep command. It searches for the given string in the specified file.

grep “linux” index.html

The below grep command searches for the words like “LINUX”, “Linux”, “linux” case insensitively.

grep -i “linux” index.html

This command will search for "linux" string in multiple files at a time. It searches in all files with file1.txt, file2.txt and along with different extensions too like file1.html, file2.php and so on.

grep “linux” file\*.\*

You can use this grep command to display the line number which contains the matched string in a file using the -n option

grep -n “word\*” file.txt

List all the lines of the file /etc/passwd that does not contain specific word “string”.

grep -v linux /etc/passwd

The below command will search linux in the “/etc” directory recursively.

grep -r linux /etc/The below command will search linux in the “/etc” directory recursively.

grep -r linux /etc/

This grep command can report the number of times the pattern matches for each file by using -c (count) option.

​grep -c 'test' /home/example/test.txt

scp:

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

From your local system to a remote system.

-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 will force scp to compresses the data as it is sent to the destination machine.

-r This option will tell scp to recursively copy directories

scp file.txt remote\_username@10.10.0.2:/remote/directory

The /remote/directory is the path to the directory you want to copy the file to, if you don’t specify a remote directory,

the file will be copied to the remote user home directory.

ssh:

ssh stands for “Secure Shell”. It is a protocol used to securely connect to a remote server/system. ssh is secure in the sense that it transfers the data in encrypted form between the host and the client.

ssh user\_name@host(IP/Domain\_name)

tar:

The below example command will create a tar archive file

# tar -cvf tecmint-14-09-12.tar /home/tecmint/

c – Creates a new .tar archive file.

v – Verbosely show the .tar file progress.

f – File name type of the archive file.

To create a compressed gzip archive file we use the option as z

# tar cvzf MyImages-14-09-12.tar.gz /home/MyImages

Untar files in Current Directory ##

# tar -xvf public\_html-14-09-12.tar

## Untar files in specified Directory ##

# tar -xvf public\_html-14-09-12.tar -C /home/public\_html/videos/

# tar -xvf thumbnails-14-09-12.tar.gz

chmod:

the chmod command sets the permissions of files or directories.

chmod u=rwx,g=rx,o=r myfile

fdisk:

fdisk also known as format disk is a dialog-driven command in Linux used for creating and manipulating disk partition table. It is used for the view, create, delete, change, resize, copy and move partitions on a hard drive using the dialog-driven interface.

A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing. The Unix/Linux systems allow stdout of a command to be connected to stdin of another command. You can make it do so by using the pipe character ‘|’.

ls -l | more

make:

Why do we need “Make”?

The reason we need “Make” is because it enables the end user to build and install your package without knowing the details of how it’s done. Every project comes with its own rules and nuances, and it can get quite painful every time you have a new collaborator. That’s the reason we have this makefile. The details of the build process are actually recorded in the makefile

diff

To execute make we need Makefile. CMake which stands for cross Makefile is tool for generating architecture dependent Makefile. So, after it(Makefile) is generated

While make is you directly writing makefile for a specific platform that you are working with.

"The build process has one step if you use a Makefile, namely typing “make” at the command line. For CMake, there are two steps: First, you need to setup your build environment (either by typing cmake <source\_dir> in your build directory or by running some GUI client). This creates a makefile or something equivalent, depending on the build system of your choice (e.g. Make on \*nix, VC++ or MinGW on Windows, etc). The build system can be passed to CMake as a parameter.

ststic&dynamic:

Static and dynamic linking are two processes of collecting and combining multiple object files in order to create a single executable.

Firstly, you need to know what a library is. Basically, a library is a collection of functions. You may have noticed that we are using functions which are not defined in our code, or in that particular file. To have access to them, we include a header file, that contains declarations of those functions. After compile, there is a process called linking, that links those function declarations with their definitions, which are in another file. The result of this is the actual executable file.

Now, the linking as I described it is a static linking. This means that every executable file contains in it every library (collection of functions) that it needs. This is a waste of space, as there are many programs that may need the same functions. In this case, in memory there would be more copies of the same function. Dynamic linking prevents this, by linking at the run-time, not at the compile time. This means that all the functions are in a special memory space and every program can access them, without having multiple copies of them. This reduces the amount of memory required