To find files

find

It is used to search for files and directories within a specified directory hierarchy/Tree based on various criteria, such as file name, size, modification time, and more.
Syntax: find [path] [expression(Can be regex)]
Examples:

1) find . -name mycode.py or find -name mycode.py→ find a file in perticular directory
2) find . -iname mycode.py → -iname ingnore case
3) If we don't specify a type we will get both files and folders
4) find . -type f/d -name f* → to specify a file or directory in find command
5) find . -perm 777 → to find files with a particular permission
6) find . ! -perm 664 → to find files which does not have a perticular permission
7) find . -type f -perm /u=r → find files where user has read permission
8) find . -type f -empty → find empty files
9) find . -type f - perm -g=w -o=x → find files with more specific permission
10) find . -regex (regex of path) → to find files using regex (.*bar* → ./foobar)
11) -inum N → Search for files with inode number 'N'.
12) -links N → Search for files with 'N' links.
13) -newer file → Search for files that were modified/created after 'file'.
14) -user name → Search for files owned by username or ID 'name'.
15) find ./ -type f -name "*.txt" -exec grep 'Geek' {} \; → Run Second command on found files.

locate

The locate command and find command is used to search a file by name. But, the difference between both commands is that locate command is a background process and searches the file in the database whereas, find command searches in the filesystem.
The locate command is much faster than find command.
If you are unable to find a file with the locate command, then it means that your database is out of date, and you can update your database with the "updatedb" command.

Syntax: locate [OPTION]... PATTERN…

File Manipulation

rm

rm stands for remove here. rm command is used to remove objects such as files, directories, symbolic links and so on from the file system like UNIX. To be more precise, rm removes references to objects from the filesystem, where those objects might have had multiple references (for example, a file with two different names). By default, it does not remove directories.
Note: In Linux operating system, there is no way to perform undo operation. Once we delete a file or directory, it is impossible to retrieve that. Hence while using rm command we have to take special care.

Syntax: rm [OPTION]... FILE...

1) rm file1 [file2 ….] → Simply delete the mentioned files.
2) rm -r mydir or rm -R folder → To delete non-empty folders. This command is the most dangerous command in linux, because it removes the total file system.
3) interactive Option(-i) → While removing files and directories, if we want confirmation then we have to use -i
4) verbose Option(-v) → If we want to know the sequence of removals on the screen we should go for -v option.
5) -f (Force Deletion) → rm prompts for confirmation removal if a file is write protected. The -f option overrides this minor protection and removes the file forcefully.
6) -d → to remove empty directory.

Note: To remove a file whose name begins with a dash (“-“), you can specify a double dash (“–“) separately before the file name. This extra dash is necessary so that rm does not misinterpret the file name as an option.

Copy (cp)

cp stands for a copy. This command is used to copy files or groups of files or directories. It creates an exact image of a file on a disk with a different file name. cp command requires at least two filenames in its arguments.

Syntax: cp [OPTION] Source-1 [Source-2 Source-3 Source-n] Directory/Destination

1) To Copy from File1 to File2 (File to File)

$ cp source_file destination_file
$ cp file1 file2
Total content fo file1 will be copied to file2.

If file2 is not already available, then this command will create that file.
If file2 is already available and contains some data, then this data will be over write with file1 content.

2) To Copy File to Directory:

$ cp file1 file2 output
file1 and file2 will be copied to output directory.
Here we can specify any number of files, but the last argument should be directory.
output directory should be available already as cp command won't create it.

3) To Copy all Files of One Directory to another Directory:

$ cp dir1/* dir2
All files of dir1 will be copied to dir2
But dir2 should be available already.

4) To Copy Total Directory to another Directory:

$ cp dir1 dir2
cp: -r not specified; omitting directory 'dir1'
Whenever we are copying one directory to another directory, compulsory we should use -r option.
$ cp -r dir1 dir2
In the above command, cp behavior depends upon whether Dest_directory exists or not. If the Dest_directory doesn’t exist, cp creates it and copies content of Src_directory recursively as it is. But if Dest_directory exists then a copy of Src_directory becomes a sub-directory under Dest_directory.

5) To Copy Multiple Directories into a Directories:
$ cp -r dir1 dir2 dir3 dir4 dir5
dir1,dir2,dir3 and dir4 will be copied to dir5

Options:

a) i → interactive for user. Outputs a prompt for copying a file.
b) b → makes a back of the copied file in the source folder.
c) -f → forceful copy. If we cant open destination file, this option will forcefully open it and copy the contents.
d) -r or R → Copying directory recursively.
e) -p → preserves the access and modification time. Should be the root user.
f) * → We can have regex for source file and folders.

Question: How to copy all txt files in the current folder to a diff folder?

mv

As its name(move) suggests this command is used to rename file directories and move files from one location to another within a file system.

Syntax: mv [options(s)] [source_file_name(s)] [Destination_file_name]

1) Renaming of files:

$ mv oldname newname
Eg: $ file1.txt file2.txt
file1.txt will be renamed to file2.txt

2) Renaming of Directories:

$ mv dir1 dir2
dir1 will be renamed to dir2

3) Moving files to directory:

$ mv a.txt b.txt c.txt output
a.txt,b.txt and c.txt will be moved to output directory.

4) Moving of all files from one directory to another directory:

$ mv dir1/* dir2
All files of dir1 will be moved to dir2. After executing this command dir1 will become empty.

5) Moving total directory to another directory:

$ mv dir1 dir2
Note: If dir2 is already available then dir1 will be moved to dir2.
If dir1 is not already available then dir1 will be renamed to dir2.

6) Options:
    a) -i → The “-i” option makes the “mv” command ask for confirmation before overwriting an existing file. If the file doesn’t exist, it will simply rename or move it without prompting.
    b) -f → mv prompts for confirmation overwriting the destination file if a file is write-protected. The -f option overrides this minor protection and overwrites the destination file forcefully and deletes the source file.
    c) -n → With -n option, mv prevents an existing file from being overwritten.
    d) -b → With this option, it is easier to take a backup of an existing file that will be overwritten as a result of the mv command. This will create a backup file with the tilde character (~) appended to it.
    e) –version → This option is used to display the version of mv which is currently running on your system.

Creation of Hidden Files and Directories

If any file starts with '.' , such type of file is called hidden file.
If we don't want to display the files then we have to go for hidden files.
Hidden files meant for hiding data. All system files which are internally required by kernel are hidden files.
We can create hidden files just like normal files, only difference is file name should starts with a dot.
touch .securefile1.txt
cat > .securefile1.txt
Even by using editors also we can create hidden files.
We can create hidden directories also just like normal directories.
mkdir .db_info
Note: By using hidden files and directories we may not get full security. To make more secure we have to use proper permissions. For this we should use 'chmod' command.

Interconversion of Normal Files and Hidden Files:

Based on our requirement, we can convert normal file as hidden file and viceversa.
mv a.txt .a.txt
We are converting normal file a.txt as hidden file.
mv .a.txt a.txt
Similarly directories also
mv dir1 .dir1
mv .dir1 dir1

View Content of the Files
We can view content of the file by using the following commands

1) cat
2) tac
3) rev
4) head
5) tail
6) less
7) more

Cat (Contd):

View Files

1) cat < file or cat file (< is optional): Displays the content of the File.
2) cat file1 file2… → Display multiple files
3) Options for better Viewing:
    a) n → Number all output lines.
    b) b → to give numbering to all lines apart from blank lines
    c) s → suppress repeated empty output lines.
    d) v → use ^ and and M- notation for non-Printable characters except LFD and Tab.
    e) T → Display Tab as ^I.
    f) E → Display $ at the end of each Line.
    g) e → Equivalent to vE.
    h) t → Equivalent to vT.
    i) A → Equivalent to vET.

cat command will display total file content at a time. It is best suitable for small files. If the file contains huge lines then it is not recommended to use cat command. We should go for head, tail, less and more commands.

tac

It is the reverse of cat.
It will display file content in reverse order of lines. i.e first line will become last line and the last line will become the first line.
This is vertical reversal.
Options:

1) b → attach the separator before instead of after
2) r → interpret the separator as a regular expression
3) s → Use the following string as a separator instead of newLine

rev

rev means reverse. Here each line content will be reversed. It is a horizontal reversal.

head

We can use the head command to view the top few lines of content.
Syntax: head [Options] FileName1 [FileName2 ..]

1) head file1 → It will display the top 10 lines of file1.txt. 10 is the default value of the number of lines.
2) head -n 30 file1.txt OR head -30 file1.txt. To display top 30 lines of the file. Instead of 30 we can specify any number.
3) head -n -20 file1.txt. To display all lines of file1.txt except the last 20 lines.
4) head -c 100 file1.txt. To display first 100 bytes of file content
5) head -q file1 file2 → Displays both files without any separator.
6) head -v file1 → verbose. Gives file name before displaying file content

tail

We can use tail command to view a few lines from the bottom of the file. It is opposite to the head command.
Syntax: head [Options] FileName1 [FileName2 ..]

1) tail file1.txt → Last 10 lines will be displayed.
2) tail -n 30 file1.txt OR tail -30 file1.txt OR tail -n -30 file1.txt → It will display the last 30 lines.
3) tail -n +4 file1.txt → It will display from 4th line to last line
4) tail -c 200 file1.txt → It will display 200 bytes of content from bottom of the file.
5) tail -q file1 file2 → Displays both files without any separator.
6) tail -v file1 → verbose. Gives file name before displaying file content.

more

We can use the more command to view file content page by page.

1) more file1.txt → It will display the first page. Enter To view the next line. Space Bar To view the next page. q To quit/exit
2) more -d file1.txt → -d option meant for providing details like --More--(5%)[Press space to continue, 'q' to quit.]
3) -n → To specify the number of lines printed at a time. Example: More -n 1 Hello prints one line at a time.
4) -s → squeeze multiple blank lines into one.

less

By using more command, we can view file content page by page only in forward Direction. If we want to move either in forward direction or in backward direction then we should go for less command.
d To go to the next page.(d means down)
b To go to the previous page. (b means backward)