Basic Linux Commands & Symbols

There are several flags associated with each command which can be used to set options and to pass in arguments to the command. A command's behaviour changes based on the flags set. This document doesn't present an exhaustive list of all the flags available but is only concerned with helping you getting up-to-speed with Linux. For a more exhaustive list of all the flags and their uses RTFM.

1. **pwd**
   Use the **pwd** (short for **P**rint **W**orking **D**irectory) command to get the absolute path of the directory (folder) you're currently in.
   Absolute path – path from the root directory (the root directory denoted by a forward slash (**/**) is the "ancestor" of all the other directories, it's the top-most directory in the hierarchy of the Linux file structure)
   Relative path – path from your current working directory

2. **cd**
   To navigate through the Linux directory structure, use the **cd** (short for **C**hange **D**irectory) command.
   Syntax: **cd [path_to_directory]**
   The directory path can be either absolute or relative. There are some shortcuts to help you navigate quickly:
   - **cd ..**
     to move one directory up
   - **cd**
     to go straight to the home folder
   - **cd –**
     to move to your previous directory

3. **ls**
   The **ls** (short for **L**i**s**t) command is used to view the contents of a directory. By default, this command will display only the visible (not hidden) contents of your current working directory. If you want to see the contents of other directories, type **ls** followed by the directory's path. Syntax: **ls [path_to_directory]**
   The directory path can be either absolute or relative.
   Some of the most commonly used flags have been mentioned below:
   - **-R**
     will list all the files in the sub-directories as well
   - **-a**
     will show the hidden files
     (instead of **ls -a [path_to_directory]** you can also use **la [path_to_directory]**)

- **-l** will list the visible (not hidden) files and directories with detailed information like the permissions, size, owner, etc. (Can you guess why the size of each directory is only 4kB?)

4. **cat**
   The **cat** (short for Con**cat**enate) command is one of the most frequently used commands in Linux. It is used to list the contents of a file on the standard output (stdout). To run this command, type **cat** followed by the name of the file.

Syntax: **cat [path_to_file]**

Here are a couple of other things you can do with the **cat** command:

- **cat > filename** creates a new file named “filename” [file names are without quotes]
- **cat filename1 filename2 > filename3** joins two files (“filename1” and “filename2”) and stores the output of them in a new file (“filename3”) [file names are without quotes]
- **cat filename1 filename2 >> filename3** joins two files (“filename1” and “filename2”) and appends the contents to “filename3”. [file names are without quotes]

5. **cp**
   Use the **cp** (short for **C**o**p**y) command to copy files from one directory to a different directory.
   Synatx: **cp [path_to_source_file] [path_to_destination_file]** (copy + rename)
   Or
   cp [path_to_source_file] [path_to_destination_directory] (copy only)

6. mv
   The primary use of the **mv** (short for **M**o**v**e) command is to move files. The syntax of the **mv** is the same as that of the **cp** command.
   Example: **mv file.txt /home/username/Documents**.
   There’s no explicit command to rename a file. The **mv** command can be used to rename a file. For example the command **mv oldname.ext newname.ext** would rename “oldname.ext” file to “newname.ext” file. [file names are without quotes]

7. mkdir
   Use the **mkdir** (short for **M**a**k**e **Dir**ectory) command to make a new directory. If you type in the command **mkdir Music,** it will create a directory **Music** in the current directory.
   Synatx: **mkdir [directories…]**
   There are extra **mkdir** commands as well:
   - To create a new directory inside another directory, use this Linux command **mkdir Music/anotherdirectory**

- use the **-p** (parents) option to create a directory in between two existing directories. For example, **mkdir -p Music/2020/Newfile** will create a new directory with name “2020” (without the quotes).

8. rm

The **rm** (short for **R**e**m**ove) command is used to delete directories and the contents within them. If you only want to delete the directory — as an alternative to **rmdir** (remove directory) — use **rm -r** (remove recursively).

Synatx: **rm FILE…**

Note: Be very careful with this command and double-check the which directory you are currently in. If used carelessly this can cause irreversible damage.

9. touch

The **touch** command allows you to create one or more new blank files through the Linux command line.

Syntax: **touch FILE…**

As an example, enter **touch file1.txt file2.txt file3.txt** to create 3 files with names “file1.txt”, “file2.txt”, and “file3.txt” (without the quotes) in your current working directory. The name of the file can also be replaced by the path to the file (path to directory + name of the file) if you want to create the files in a directory different from you current working directory.

10. man

To know more about a command and how to use it, use the **man** (short for **Man**ual) command. It shows the manual pages of the command.

Syntax: **man [command_name/tool_name]**

For example, “**man cd**” will show the manual pages for the **cd** command.

11. echo

This command is used to display text/string (put them in the standard output) that are passed in as arguments.

Syntax: **echo [string]**

12. **diff**

Short for difference, the **diff** command compares the contents of two files line by line. After analyzing the files, it outputs the lines that do not match.

Syntax: **diff [file1] [file2]**

13. sudo

Short for “**Super User Do**”, this command enables you to perform tasks that require administrative or root permissions. It is not advisable to use this command often.

14. **df**

Use **df** (short for **D**isk **F**ree) command to get a report on your system's disk space usage, shown in percentage and KBs. If you want to see the sizes in megabytes, type **df -m**.

15. **grep**

Another basic Linux command that is undoubtedly helpful for everyday use is the **grep** (short for **G**lobal **R**egular **E**xpression **P**rint) command. It lets you search through a file using regular expressions.
Synatx: **grep [pattern] FILES…**

16. **scp**

The **scp** (short for Secure Copy) command is used to copy files between servers in a secure way.
Syntax: **scp [[[user1@]host1:]file1] [[[user2@]host2:]file2]**
For example, if you want to copy a file from you local machine to your mars server at IITB, you would do something like,
scp [file_source_path] <your_mars_id>@mars.cse.iitb.ac.in: [file_destination_path]

17. **ssh**

The **ssh** (short for **S**ecure **Sh**ell) command is used to connect to a remote server/system securely. It can be used for things such as accessing a remote server, port forwarding, etc. (Which port does SSH run at?)
Syntax: **ssh [user@host]**
For example, if you want to connect to the mars server at IITB, you would do something like, **ssh <your_mars_id>@mars.cse.iitb.ac.in**

18. **>>** & **>**

Both of these symbols are used to redirected the data in the standard output (stdout) stream to a file.
Examples:
1. **echo Linux > Linus_Torvalds.txt**
2. **echo Linux >> Linus_Torvalds.txt**

The difference between them is that while **>** overwrites the file whose name is specified (if such a file already exists), **>>** appends the text to the file. Both of these symbols create a new file when a file with the specified name doesn't exist.

Resources:

1. https://www.unixtutorial.org/basic-unix-commands