1. Review of Linux file structure

Linux file system is based on a tree hierarchy. There is a top-level with other sublevels branching beneath it.

The tree hierarchy offers storage and quick access. Unlike the Windows file system, where it uses **drive letters**, Linux stores everything within a single directory structure called a **virtual directory**.

Comparison between the Windows and Linux file systems.

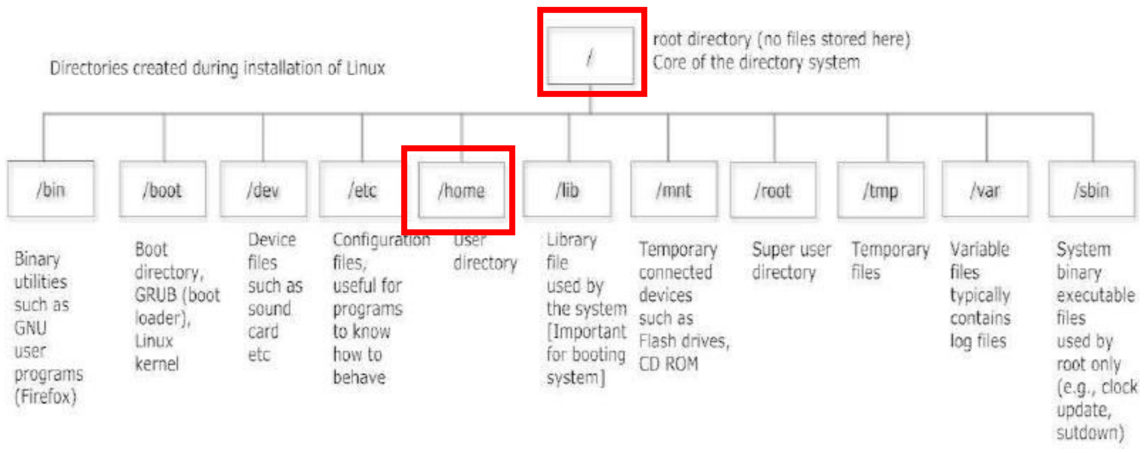
WINDOWS	**LINUX**
Separator: Backslash \	Separator: Forward slash /
C:\Users\Rich\Documents\test.doc	/home/rich/Documents/test.doc
Indicates that test.doc is located in Documents, which itself is located in directory Rich. Rich is contained in directory Users, which is located on the hard drive partition assigned letter C.	Indicates only that the file test.doc is in directory Documents, under the directory rich, which is contained in the directory home.
C is usually a volume placed in the first hard drive on the PC.	It does not provide information as to which physical disk on the PC the file is stored.

First hard drive installed in a Linux PC is called **root drive**. Root drive contains core of the virtual directory. Everything else builds from there. On the root drive, Linux creates mount points (these are special directories where you assign additional storage devices).

The virtual directory causes files and directories to appear within these mount points, even though they are physically stored on a different drive.

One hard drive partition is associated with the root directory (normally in the first hard drive).

Other hard drives can be mounted anywhere in the directory structure. For example, second hard drive partitions can be mounted on /home (where user directories are located).


Directories created during installation of Linux
/
root directory (no files stored here)
Core of the directory system
/bin
Binary utilities such as GNU user programs (Firefox)
/boot
Boot directory, GRUB (boot loader), Linux kernel
/dev
Device files such as sound card etc
/etc
Configuration files, useful for programs to know how to behave
/home
User directory
/lib
Library file used by the system [Important for booting system]
/mnt
Temporary connected devices such as Flash drives, CD ROM
/root
Super user directory
/tmp
Temporary files
/var
Variable files typically contains log files
/sbin
System binary executable files used by root only (e.g., clock update, sutdown)

In the picture above, the root directory (/) and /home can be different hard drive partitions.

2. Traversing directories

We can refer to a file by **absolute** or **relative path** names.

Absolute pathname: Specifies full path from the root to the desired directory or file.

- Absolute pathname to my home directory is **/afs/ec.auckland.ac.nz/users/a/m/amah811/unixhome**
- To refer to file1 in my home directory, the absolute pathname: **/afs/ec.auckland.ac.nz/users/a/m/amah811/unixhome/file1.txt**

Relative pathname: Path from current directory to a file or directory

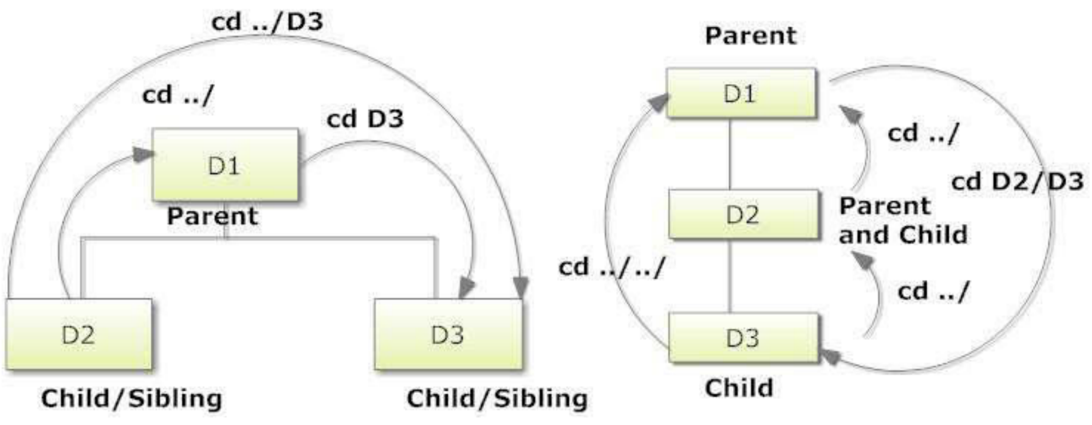
- If my current directory is **/afs/ec.auckland.ac.nz/users/a/m/amah811**
- Relative path to file1.txt in my home directory is **unixhome/file1.txt**
- And relative path to my home directory is **unixhome**

3. Changing directories (cd command)

A relative file path starts with either a directory name or a special character indicating a relative location to your current directory location.

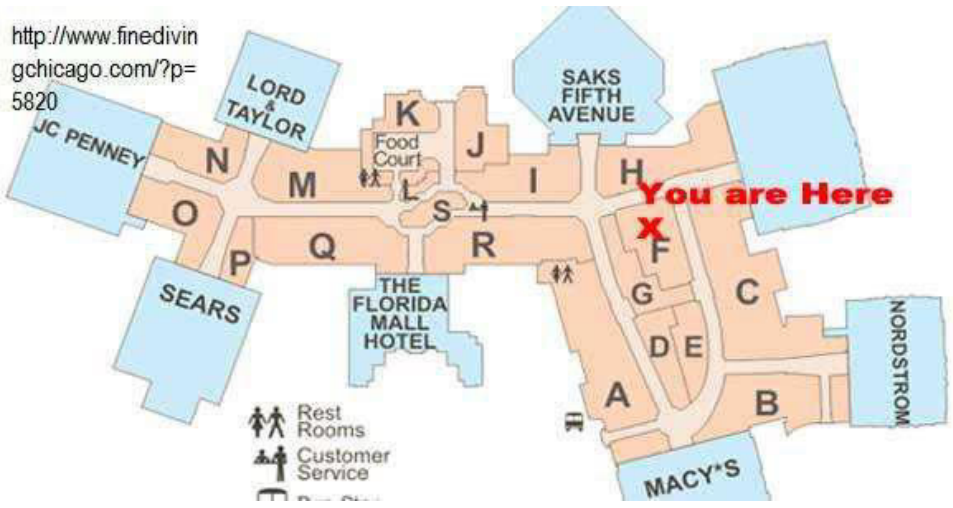
- The dot (.) represents current directory
- The dot dot (..) represents the parent directory

You can also use absolute paths. For example: cd /home/user/directory



4. The pwd command

It displays the current working directory.

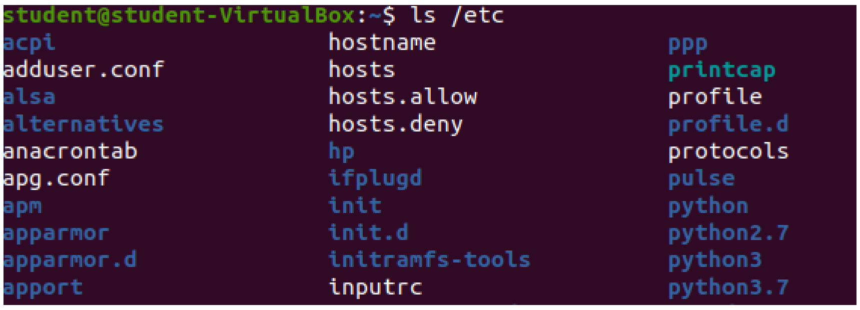


Does not have many practical options, but one of the most important commands as it lets you know where you are in the tree hierarchy.

```
pwd /afs/ec.auckland.ac.nz/users/a/m/amah811/unixhome
```

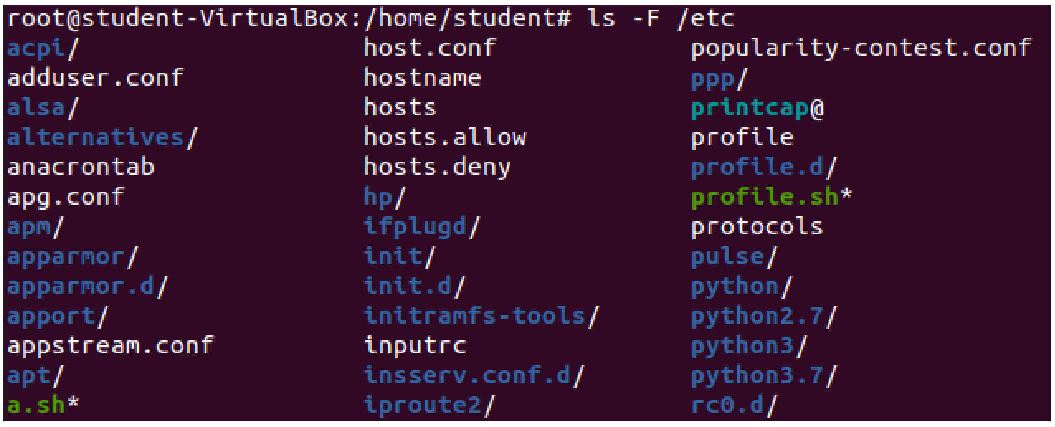
5. File and Directory Listing (ls)

Basic listing (ls): Displays files and directories in your current directory. It produces listing in alphabetic order (columns).



```
student@student-VirtualBox:~$ ls /etc
acpi                    hostname                ppp
adduser.conf            hosts                   printcap
alsa                    hosts.allow             profile
alternatives            hosts.deny              profile.d
anacrontab              hp                      protocols
apg.conf                ifplugd                 pulse
apm                     init                    python
apparmor                init.d                  python2.7
apparmor.d              initramfs-tools         python3
apport                  inputrc                 python3.7
```

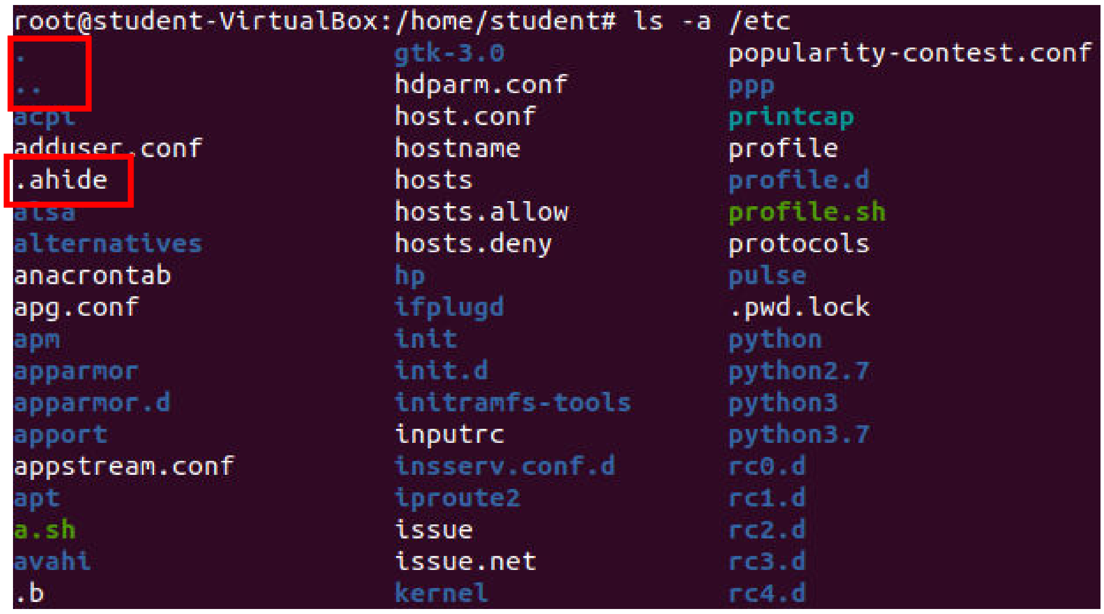
To distinguish between files and directories use the -F parameter.



```
root@student-VirtualBox:/home/student# ls -F /etc
acpi/                   host.conf               popularity-contest.conf
adduser.conf            hostname                ppp/
alsa/                   hosts                   printcap@
alternatives/           hosts.allow             profile
anacrontab              hosts.deny              profile.d/
apg.conf                hp/                     profile.sh*
apm/                    ifplugd/                protocols
apparmor/               init/                   pulse/
apparmor.d/             init.d/                 python/
apport/                 initramfs-tools/        python2.7/
appstream.conf          inputrc                 python3/
apt/                    insserv.conf.d/         python3.7/
a.sh*                   iproute2/               rc0.d/
```

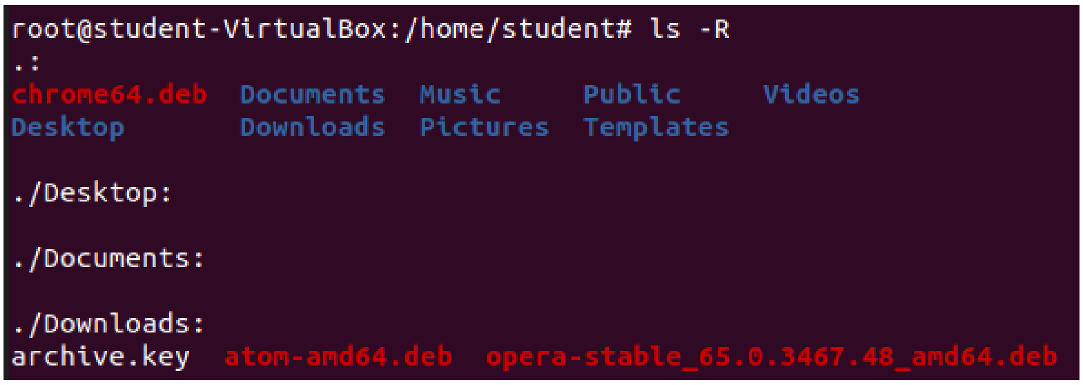
/ → Directory, * → Executable

Use the -a parameter to show *hidden files* starting with a dot (.). Notice the (.) and (..) special directories to specify the current and parent folder.



```
root@student-VirtualBox:/home/student# ls -a /etc
.                        gtk-3.0                 popularity-contest.conf
..                       hdparm.conf             ppp
acpi                     host.conf               printcap
adduser.conf             hostname                profile
.ahide                   hosts                   profile.d
alsa                     hosts.allow             profile.sh
alternatives             hosts.deny              protocols
anacrontab               hp                      pulse
apg.conf                 ifplugd                 .pwd.lock
apm                      init                    python
apparmor                 init.d                  python2.7
apparmor.d               initramfs-tools         python3
apport                   inputrc                 python3.7
appstream.conf           insserv.conf.d          rc0.d
apt                      iproute2                rc1.d
a.sh                     issue                   rc2.d
avahi                    issue.net               rc3.d
.b                       kernel                  rc4.d
```

Use the -R parameter to recursively show the contents of all directories contained in the directory where you do the listing.



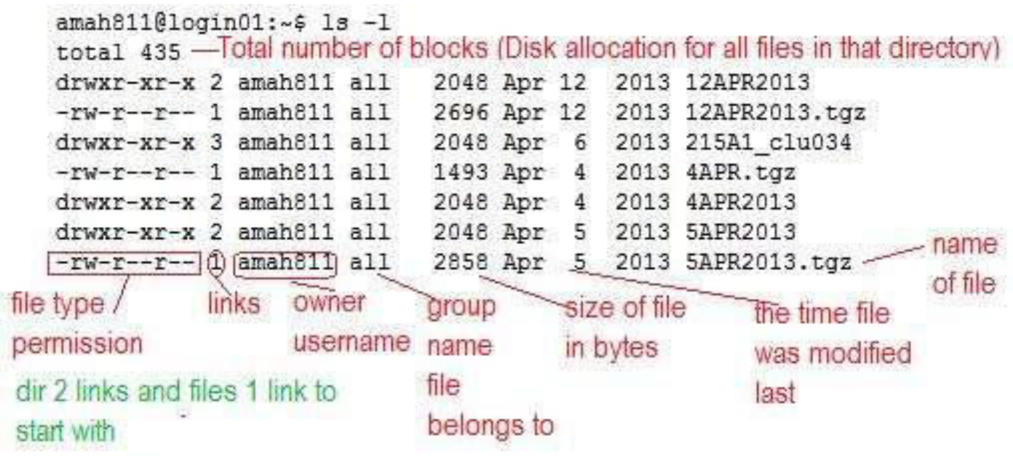
```
root@student-VirtualBox:/home/student# ls -R
.:
chrome64.deb  Documents  Music     Public     Videos
Desktop       Downloads  Pictures  Templates

./Desktop:

./Documents:

./Downloads:
archive.key  atom-amd64.deb  opera-stable_65.0.3467.48_amd64.deb
```

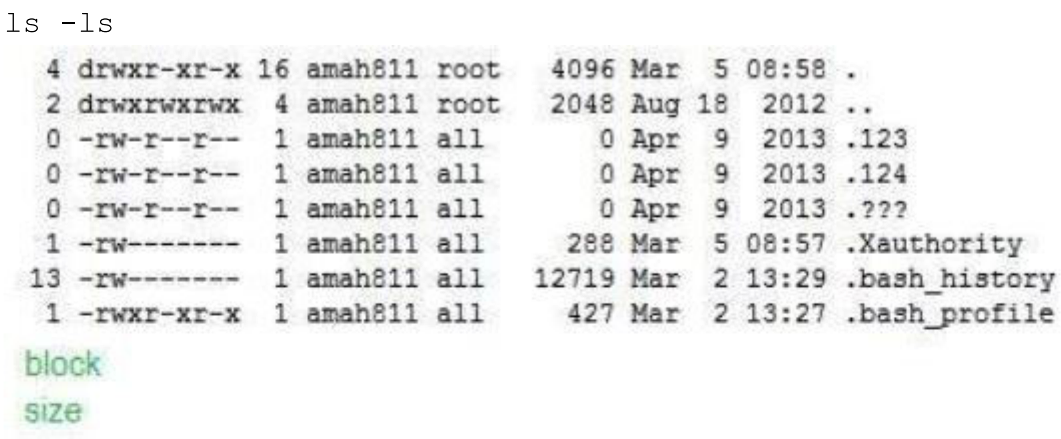
The basic listing does not produce much information. Use the **-l** parameter to produce a long listing.



```
amah811@login01:~$ ls -l
total 435
drwxr-xr-x 2 amah811 all   2048 Apr 12  2013 12APR2013
-rw-r--r-- 1 amah811 all   2696 Apr 12  2013 12APR2013.tgz
drwxr-xr-x 3 amah811 all   2048 Apr  6  2013 215A1_clu034
-rw-r--r-- 1 amah811 all   1493 Apr  4  2013 4APR.tgz
drwxr-xr-x 2 amah811 all   2048 Apr  4  2013 4APR2013
drwxr-xr-x 2 amah811 all   2048 Apr  5  2013 5APR2013
-rw-r--r-- 1 amah811 all   2858 Apr  5  2013 5APR2013.tgz
```

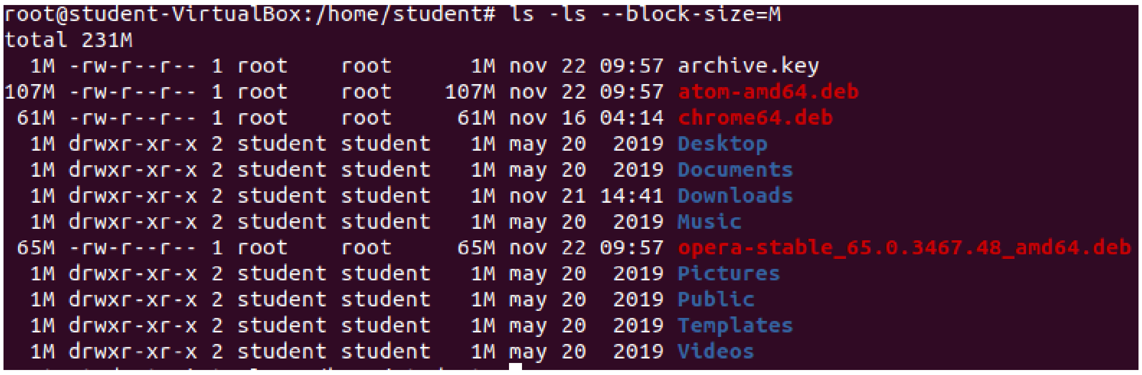
Total number of blocks (Disk allocation for all files in that directory)
name of file
file type / permission
links
owner username
group name file belongs to
size of file in bytes
the time file was modified last
dir 2 links and files 1 link to start with

Use the -s parameter to print block size of each file. In this example, we are combining the output of the options -l and -s (it can be written like **ls -ls**, **ls -l -s** or **ls ls**).



```
ls -ls
 4 drwxr-xr-x 16 amah811 root   4096 Mar  5 08:58 .
 2 drwxrwxrwx  4 amah811 root   2048 Aug 18  2012 ..
 0 -rw-r--r--  1 amah811 all       0 Apr  9  2013 .123
 0 -rw-r--r--  1 amah811 all       0 Apr  9  2013 .124
 0 -rw-r--r--  1 amah811 all       0 Apr  9  2013 .???
 1 -rw-------  1 amah811 all     288 Mar  5 08:57 .Xauthority
13 -rw-------  1 amah811 all   12719 Mar  2 13:29 .bash_history
 1 -rwxr-xr-x  1 amah811 all     427 Mar  2 13:27 .bash_profile
```

The block size can be changed using the parameter --block-size=X (X means: K for Kilobytes, M for Megabytes, G for Gigabytes, etc).



```
root@student-VirtualBox:/home/student# ls -ls --block-size=M
total 231M
  1M -rw-r--r-- 1 root    root      1M nov 22 09:57 archive.key
107M -rw-r--r-- 1 root    root    107M nov 22 09:57 atom-amd64.deb
 61M -rw-r--r-- 1 root    root     61M nov 16 04:14 chrome64.deb
  1M drwxr-xr-x 2 student student   1M may 20  2019 Desktop
  1M drwxr-xr-x 2 student student   1M may 20  2019 Documents
  1M drwxr-xr-x 2 student student   1M nov 21 14:41 Downloads
  1M drwxr-xr-x 2 student student   1M may 20  2019 Music
 65M -rw-r--r-- 1 root    root     65M nov 22 09:57 opera-stable_65.0.3467.48_amd64.deb
  1M drwxr-xr-x 2 student student   1M may 20  2019 Pictures
  1M drwxr-xr-x 2 student student   1M may 20  2019 Public
  1M drwxr-xr-x 2 student student   1M may 20  2019 Templates
  1M drwxr-xr-x 2 student student   1M may 20  2019 Videos
```

6. The cp command (copying)

Allows you to make copy of a file or directory.

The general format is:

```
cp [option] source destination
```

- -i option for interactive mode to give you a warning before overwriting an already existing file.

Allows any combination of full or partial path.

Copy from current directory to parent directory:

```
   cp source ../
```

Copy from current directory to sibling directory:

```
   cp source ../sibling
```

Copy file from parent to current directory:

```
   cp ../source .
```

Recursively copy contents of a directory to another

```
   cp -R source_dir destination_dir
```

7. The mv command (moving)

Allows you to move or rename a file or directory

```
mv source destination
```

- -i option for interactive

For the mv command, -R parameter does not exist. Linux detects directories and directly moves them.

Move files from current directory to parent directory:

```
   mv source ../
```

Move from current directory to sibling directory:

```
   mv source ../sibling
```

Move file from parent to current directory:

```
   mv ../source .
```

Recursively move contents of a directory to another

```
   mv source_dir destination_dir
```

8. The mkdir command (creating)

To create new directories:

```
mkdir [option] directory_name(s)
```

For example, the following command would create the directory dir_1 into the current one:

```
   mkdir dir_1
```

It is also possible to create more than one directory at the same time:

```
   mkdir dir_1 dir_2 dir_3
```

We can also use absolute paths:

```
   mkdir /home/user/dir_1
```

The -p option creates the intermediate directories if they do not already exist. For example:

```
   mkdir -p food/fruit/citrus/oranges
```

9. Removing files and directories (rm and rmdir)

The **rm** command allows deleting files.

```
rm -i file
```

- Use the -i to get a warning message.
- Use -f option for forcible deletion (no warnings).

You can use both absolute and relative paths.

To delete a file into the current folder

```
   rm file
```

To force deleting a file into /etc (as a root)

```
   rm -f /etc/default/grub
```

The **rmdir** allows deleting of only empty directories.

For example, to delete a directory located in the current one:

```
   rmdir directory
```

You can also specify -p option (like in mkdir) to delete intermediate directories, as long as all of them are empty:

```
rmdir -p dir1/dir2/dir3
```

`rm -rf` directory recursively deletes all contents of a file.

- The -r option is to delete all the files and subfolders from the given directory.
- The -f option is to suppress warnings about descending into subdirectories and deleting contents

For example:

`rm -rf Folder` (it will delete the folder and subfolders).

10. The touch command

Used to update the modification date and time of a file; does not modify the contents.

If a file does not exists, then touch can be used to create an empty file.

```
touch filename
```

Use the -t parameter to specify the time.

```
touch -t 201911221200 testfile
```

11. The cat command

The command named "cat" can be used to display/concatenate one or more files, displaying the output all at once.

Example: Display the contents of a file called "assign1.txt".

```
cat assign.txt
```

You can also use it to create files from keyboard input as follows (> is the output redirection operator, which will be discussed later).

```
cat > hello.txt
hello
world!
```

You can write all the lines you want. Once you have finished writing, just type [Ctrl+D]

```
cat hello.txt
hello
world!
```

12. Less is more

```
more target-file(s)
```

The command **more** displays the contents of `target-file(s)`on the screen, pausing at the end of each screenful and asking the user to press a key (useful for long files). It also incorporates a searching facility (press '/' and then type a phrase that you want to look for).

You can also use more to break up the output of commands that produce more than one screenful of output as follows (| is the pipe operator, as we will see later):

```
ls -l | more
```

The orders to manipulate the displayed content are:

- Space bar: To advance to next page
- Enter: To advance to next line

On the other hand, **less** is just like more, except that has a few extra features (such as allowing users to scroll backwards and forwards through the displayed file). less not a standard utility, however and may not be present on all UNIX systems.

Using less, you can also type:

- b: to go back a page Enter Key
- Arrow keys: To scroll horizontally

13. Grep: Seeking lines in files

```
grep [-options] expreg [files]
```

Seeking in files the lines containing strings satisfying certain conditions. The conditions are explained in a regular expression. Options:
-c: Count the number of lines satisfying the condition
-v: Select the non-matching lines
-i: Case insensitive
-n: Return the line numbers

Following characters can be used in the expression:
. For replacing any character
***** Repeat the previous character
^ Starting of a line
$ Ending of a line
[...]: A list or a range of characters
[^..]: A list or a range of excluding characters

Attention: The expression should be between double quotes `""`

```
grep "^t" /etc/passwd
```
Find in the /etc/passwd all lines starting with "t"

```
grep "^[^t]" /etc/passwd
```
Find all lines which do not start with “t”

```
grep -v "tuananh" /etc/passwd
```
Find all lines not containing “tuananh”

14. Pipes and redirection

The **pipe** ('|') operator is used to create concurrently executing processes that pass data directly to one another. It is useful for combining system utilities to perform more complex functions. For example:

```
cat hello.txt | sort | uniq
```

Creates three processes (corresponding to cat, sort and uniq) which execute concurrently. As they execute, the output of the cat process is passed on to the sort process, which is in turn passed on to the uniq process. uniq displays its output on the screen (a sorted list of users with duplicate lines removed). Similarly:

```
cat hello.txt | grep "dog" | grep -v "cat"
```

Finds all lines in hello.txt that contain the string "dog" but do not contain the string "cat".

The output from programs is usually written to the screen, while their input usually comes from the keyboard (if no file arguments are given). In technical terms, we say that processes usually write to standard output (the screen) and take their input from standard input (the keyboard). There is in fact another output channel called standard error, where processes write their error messages; by default error messages are also sent to the screen.

To redirect standard output to a file instead of the screen, we use the **> operator**:

```
echo hello
hello

echo hello > output
cat output
hello
```

In this case, the contents of the file output will be destroyed if the file already exists. If instead we want to append the output of the echo command to the file, we can use the **>> operator**:

```
echo bye >> output
cat output
```

```
hello
bye
```

To capture standard error, prefix the > operator with a 2 (in UNIX the file numbers 0, 1 and 2 are assigned to standard input, standard output and standard error respectively), e.g.:

```
cat nonexistent 2>errors
cat errors
cat: nonexistent: No such file or directory
```

You can redirect standard error and standard output to two different files:

```
ls -R . > errors 2>files
```

or to the same file:

```
ls –R . >output 2>output
```

or to the same file with the abbreviated:

```
ls –R . >& output
```