# Linux Basic Commands

### - pwd

Print Working Directory

### - /

Root

### - ls -lah

Full meaning list long all human readable output.

ls - list

l - long

a - all plus hidden files

h - human readable output (size show friendly)

For the output, if a letter start with d that means it is a directory, if it shows a “**-”** dash that means this is a file.

### Please note

Order doesn’t matter when exchange the parameters

For example: **ls -hal**

**Permission Owner Group Size**

**drwxrwxr-x2 mguiro mguiro 4096 ene...**

**-rwxrwxr-x2 miguiro mguiro 4096 ene...**

As you can see from above. From the “**Permission**”, The first line first character starts with a “d”, which means it is a directory, the second line starts with a dash “-”, which means it is a file. Sometimes it starts with a “l”, which means this is a symbolic link.

### - man chown

Search the manual page

man have different sections

Section 1 is user command

Section 2 is system calls

Section 3 is library reference

Section 4 is device drivers

Section 5 is configuration files and other file formats

Section 6 is games

Section 7 is miscellaneous

Section 8 is system command

E.g. man 3 newsgroups

To search the man pages for “newsgroups”

### - cd~

Change to your home directory

### - cd -

Change to previous directory

### - cd bas [TAB]

For e.g. The directory “**bas**”. If you press “TAB” key. It will get possible path completion.

### - touch command

Creates a file that does not exists.

### Example:

### touch text.txt

## - File/Directory Permissions

When a user creates a file, it will be assigned

* r - read
* w - write
* x - execute

In order to describe the file or directory permissions, lets create a text file “text.txt” in “D” drive. If you are at the home directory, I assume you are at the “C” drive. In order to switch to the “D” drive. Please type in:

### cd /mnt/d

Please take a note about the slash, it must be a **forward** slash, not ***backward*** slash (which is used by Microsoft), also there must be a slash at the beginning, because you are starting from the **root** directory.

After creating the text.txt file, browse it by typing in the “**ls -lah**”

**Permission Owner Group Size**

**-rwxrwxrwx alantang alantang 0 text.txt**

As you can see from above, the first character of the permission column starts with a dash, which means this is a file **not** directory or symbolic link. Let’s deal with the permission column first.

For the permission, forget about the first character, the 9 characters “rwxrwxrwx”, can be group as 3 groups.

(rwx) - First group are **user owner** permission, that is the second column “**Owner**”.

(rwx) - Second group are permissions that apply to the **group** where the user belongs to, that is the third column “**Group**”.

(rwx) - Third group are permissions that applies to **everyone**.

To change the Permission of each specific group, you need to use the command “**chmod**”.

Let’s take an example of how this can be done. For instance, I am a user “alantang” (forget about the group), I just need to have “read” and “execute” file permission, but not “write” permission (that means edit the file), how could this be done? The answer is by typing in the command “**sudo chmod 577 text.txt**”.

Why 577? Below explains.

Let’s first ask, how do we distinguish “rwx”? which represent read, write and execute. In order to represent each character, we will assign each with a number.

x

w

r

1

2

4

For the above, as you can see each single character, has the base of 2 and to the power of 0, 1, 2. Power of 0 must be “x”, power of 1 must be “w” and the last power of 2 is “r”. Therefore, **order matters**. For instance, if we want to grant all access (read, write, execute) of a file to a **user** but not a **group** or **everyone**. That means full access is 7 for a user but **0** for “group” and “everyone”, when we type in the command, we type in “**sudo chmod 700**”. Now back to above, if we just need to grant “**read**” and “**execute**” privilege, but not **write** for the **user** and **group** but not everyone. That means the above situation will be:

0

4

x

w

r

1

For read is “4”, write is zero (as we want to revoke this privilege) and the last execute is “1”. Now by adding the numbers (4 + 0 + 1), we will get a number 5. We just mentioned this privilege is applying to user and group but not everybody. So, for the command we type in “**sudo chmod 550**”.

Everyone

Group

User

0

5

5

Remind , don’t forget the order:

### Owner User Owner Group Everyone

The final results will be shown below, remember a **dash** sign means the permission is being revoke or does not have gain this permission.

**Permission Owner Group Size**

**-r-xr-x--- alantang alantang 0 text.txt**

## Alternative Ways to change permission

There is another way to change permission instead of assigning numbers, which will be shown below:

**sudo chmod text.txt -rw-rw-r--**

For the above file as you can see the first character means it is a file and the first group represent it is a user; second group is a group, both has the privilege of “read and write” access only but not execute “-”. Talking about **execute** privilege, means it cannot be run or compile as program, or even **open the file for editing**. The third group has only “read” access but not write and execute.

Let’s take an example of how you will modify the permission of the file without assigning numbers:

### sudo chmod u+x text.txt

Just using the command **chmod**, however, the difference is this time you are not assigning numbers, instead a bunch of **letters** characters.

* chmod - to change the file permission
* u - represents user, which means you are dealing with the user group.

g - represents the group of the user belongs.

o - represents others, you can think of everyone.

* + - plus, sign, means you are assigning this privilege.

- - minus sign, means you are revoking this privilege.

* text.txt - the file name

From this command “**sudo chmod u+x text.txt**”, you can see we are assigning “execute” but not “read” and “write” permission to the user of the file text.txt. This only applies to “user” but not “group” and “everyone”.

Let’s take another example:

### sudo chmod o-r text.txt

What this command do is revoking **read** access for everyone, that means everyone (“o” means everyone) except the **user** and **group** who don’t have read access of the “text.txt” file.

## One more example:

### sudo chmod go=rx text.txt

Now you are familiar with the “+” and “-” sign, the “=” equal sign means directly assign the privilege to groups or owner of this file. The “go” means, G “Group of the user belongs to” and the O “Others that means everyone for short”. Now the group of the user it belongs and everyone has “read” and “execute” access, but not “write” access.

## Change the Group or Owner of the file

## - chown

There are sometimes you need to change the owner or group of the file, “chown” command does this function. **Notice** that we dealing with the ownership of the file, that means only **administrator** privilege can perform this function. Therefore, whenever you type in a command and want to perform administrative privileges you need to add a “**sudo**” (**sudo** means **supper user**) before the command.

### sudo chown mysql text.txt

The above command changes the **user and group** of the text file “text.txt” to **mysql**.

### sudo chown :mysql text.txt

The above command changes the **group** of the text file “text.txt” to **mysql**.

### sudo chown mysql:mysql text.txt

The above command changes the user and the group of the text file “text.txt” to **mysql**.

## - whoami command

Specify which user sign-in the system.

## - mkdir

To create a directory. You can know it is a directory by checking if there is a letter “**d**” in the first position of the permission column.

## Relative Path

The definition of relative path is who is related to the **working directory** “you can check your working directory by the pwd command”.

For instance, “**mkdir code/version1**”, means there will be a directory code and a subdirectory version1 created inside your working directory.

To list the content, you can do:

### ls code/ -lah

## Absolute Path

* Use pwd command to print the working directory

The results of the path **always** start from the root that is a “**/**” forward slash at the beginning.

* For example, move to any directory you like not your **working directory**, and if you want to create the code/version1 just like before you can type:

### mkdir /home/alantang/code/version1

## Question: How do I know if it is a relative path or absolute path?

**Answer**: **if it is relative path, it does not start with a forward slash “/”. Otherwise, if it is an absolute path, it starts with a forward slash “/”, because absolute path start counting from the root.**

### \*Remember a relative path is always relates to your working directory.

## - rm command

The remove command removes unwanted files or directory.

For example:

### rm -r code/version1 /home/alantang/code/version2

as shown above the remove command removes 2 **folders**, the first removes the **relative path** of version1 folder, the second removes the **absolute path** version2 folder. The “r” option parameters tell the system to removes all the content (including all files and sub-directories”) inside the folder.

You can also use wildcards when deleting the files

### rm exam?.txt

The above command removes the files which begins with “**exam**”, from your **working directory** you are working on.

### rm /home/alantang/\*.txt

The above command removes all the files that has the “txt” extension, from the **absolute path** “**/home/alantang/**”

**\*Note**: when removing **folder(s)** using the “rm” command, it is a **must** to add a “r” option parameter. How do you know if it is a file or folder? The best way is to list out the file “**ls -lah**” by the command, and check the **beginning** character, if it is a dash “**-**”. it is a file, else if it is a “**d**”, it is a directory.

## - mv command

mv command moves the file(s) or directories(s) from the source to destination.

For example:

### mv /home/alantang/code/version1/\*.txt code/version2/

The above command moves all the files with the “.txt” extension from the absolute path “/home/alantang/code/version1/” to the relative path “code/version2”. when using the mv command, you need to provide the source and target location. Files will not remain in the source location when moved.

**\*Note**: “**mv”** command can be used to **rename** file or directory.

For example:

### mv -r debian unix

The above syntax renames the folder “debian” to “unix”, the “r” indicates all the file(s) and subdirectories(s) inside the debian folder. The option parameters will be discussed below.

## - cp command

Same as “mv” command, which moves the file(s) or directorie(s) from the source to destination, except one differences is the file(s) copied will still remain in the source destination.

For example:

### cp /home/alantang/code/version1/\*.txt code/version2/

Another example:

### cp -ir mydir/ /tmp

The above command copies all the content(s) from the relative directory “**mydir/**” source to the absolute path “**/tmp**” destination. The system will **prompt a message** before copy/overwrite (“i” option parameter). The “r” option indicates all the contents (including files and subdirectories) inside the “**mydir/**” folder. Please reminded all the contents of the “mydir” folder still remains.

One more example:

### cp -ir foo bar baz.txt /tmp

The above command copies the folders “foo”, “bar”, and the file baz.txt from the source to the absolute path “/tmp” destination directory. Folders and files still remain in the source. System message will be prompt before copy/overwrite.