# Chmod (change mode): permission sets

Chmod (change mode) is one of the most frequently used commands in unix or linux operating system. The chmod command is used to change the file or directory access permissions.
To know about the access permissions of a file or directory, use the ls -l command as shown below:

$ ls -l  filename.sh
-rwx-rw-r—1 student student 94 Oct  4 03:12 filename.sh

The **syntax** of chmod command is:

**chmod [options] mode/permissions filename**

*permissions* defines the permissions for the **owner of the file (the "user"), members of the group who owns the file (the "group")**, and **anyone else ("others").**
There are two ways to represent these permissions:

- with symbols (alphanumeric characters), or
- with octal numbers (the digits **0** through **7**).

**Symbolic Representation of Permissions**:

The following symbols are used to represent the users, groups and others:

- u : User
- g : Group
- o : Others
- a : All (user, group and others)

The following symbols represent the permissions:

- r : read
- w : write
- x : execute

The following symbols represent the permissions grant or revoke:

- + : Additional permissions. Selected permissions are added.
- - : Revoke the permissions. Selected permissions are revoked.

- = : Specific permissions. Only selected permissions are assigned.

Let's say you are the owner of a file named **filename.txt**, and you want to set its permissions so that:

1. the **u**ser can **r**ead, **w**rite, and e**x**ecute it;
2. members of your **g**roup can **r**ead and e**x**ecute it; and
3. **o**thers may only **r**ead it.

This command will do the trick:

$ Chmod ugo+rwx filename.txt

$ Chmod ug-wx filename.txt

```
chmod u=rwx,g=rx,o=r filename.txt
```

This example uses symbolic permissions notation. The letters **u**, **g**, and **o** stand for "**user**", "**group**", and "**other**". The equals sign ("**=**") means "set the permissions exactly like this," and the letters "**r**", "**w**", and "**x**" stand for "read", "write", and "execute", respectively. The commas separate the different classes of permissions, and there are no spaces in between them.

Here is the equivalent command using octal permissions notation:

```
chmod 754 filename.txt
```

Here the digits **7**, **5**, and **4** each individually represent the permissions for the user, group, and others, in that order. Each digit is a combination of the numbers **4**, **2**, **1**, and **0**:

- **7** stands for "read, write and execute"(1 1 1 =7)
- **6 stands for(read, write) (110=6)**
- **4 stands for "read"(100=4)**
- **2** stands for "write",(010=2)
- **1** stands for "execute"(001=1), and
- **0** stands for "no permission."(000=0)

So **7** is the combination of permissions **4**+**2**+**1** (read, write, and execute), **5** is **4**+**0**+**1** (read, no write, and execute), and **4** is **4**+**0**+**0** (read, no write, and no execute).

To make a file and run it, use the following commands :

$chmod u+x filename (set file as exectable)

$./filename (To execute shell program or Run)

```
#Example of for LOOP
#!/bin/bash
for ((i=1;i<=5;i++))
do
        echo -n -e "$i\t"
done
ouput:
debasis@LAPTOP-H3N6JCNE:~$ chmod u+x for.sh
debasis@LAPTOP-H3N6JCNE:~$ ./for.sh
1       2       3       4       5
```