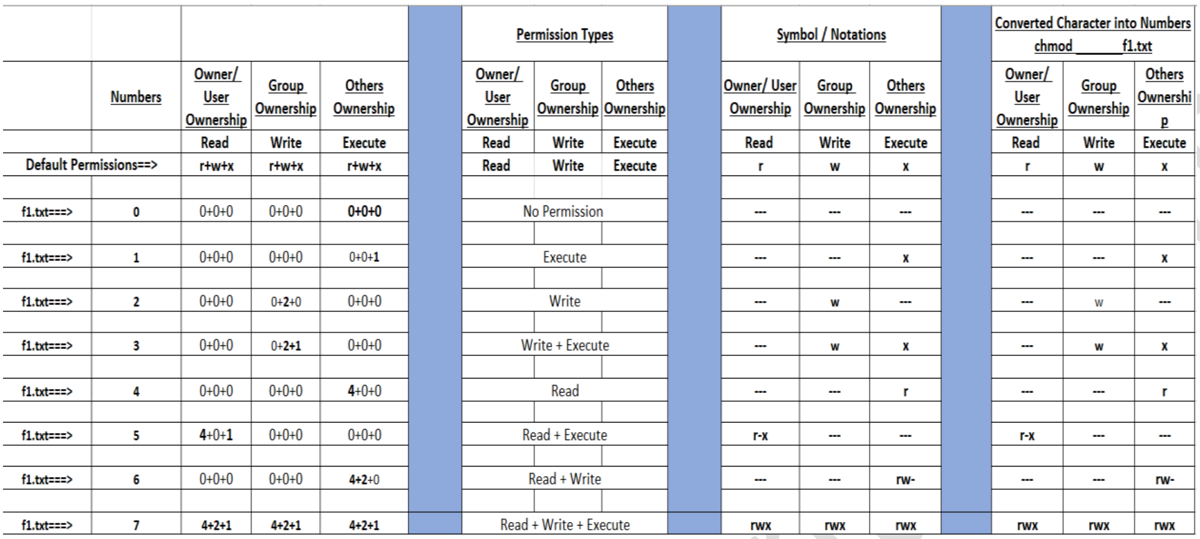
Red Hat
Enterprise
Linux 9

RHCSA-SA1 LAB-Book

Chapter 7 – Controlling Access to Files

A] Simple Permission:

Chart as below:



					Permission Types			Symbol / Notations			Converted Character into Numbers chmod ______ f1.txt		
	Numbers	Owner/ User Ownership	Group Ownership	Others Ownership	Owner/ User Ownership	Group Ownership	Others Ownership	Owner/ User Ownership	Group Ownership	Others Ownership	Owner/ User Ownership	Group Ownership	Others Ownership
		Read	Write	Execute	Read	Write	Execute	Read	Write	Execute	Read	Write	Execute
Default Permissions==>		r+w+x	r+w+x	r+w+x	Read	Write	Execute	r	w	x	r	w	x
f1.txt===>	0	0+0+0	0+0+0	0+0+0	No Permission			---	---	---	---	---	---
f1.txt===>	1	0+0+0	0+0+0	0+0+1	Execute			---	---	x	---	---	x
f1.txt===>	2	0+0+0	0+2+0	0+0+0	Write			---	w	---	---	w	---
f1.txt===>	3	0+0+0	0+2+1	0+0+0	Write + Execute			---	w	x	---	w	x
f1.txt===>	4	0+0+0	0+0+0	4+0+0	Read			---	---	r	---	---	r
f1.txt===>	5	4+0+1	0+0+0	0+0+0	Read + Execute			r-x	---	---	r-x	---	---
f1.txt===>	6	0+0+0	0+0+0	4+2+0	Read + Write			---	---	rw-	---	---	rw-
f1.txt===>	7	4+2+1	4+2+1	4+2+1	Read + Write + Execute			rwx	rwx	rwx	rwx	rwx	rwx

Practical based on above chart:

1) To set the permission rwx to the owner / user, rw- to the Group and r—to the others use the below command and calculation:

➔

Step 1:

Owner / User: rwx

Group: rw-

Others: r—

whereas,

- Represents nothing means in digit form it is showing as 0.
-

Step 2: Convert character into digits:

Values: r=4, w=2 and x=1

i.e., rwx = 4+2+1 = 7

rw = 4+2+0 = 6

r-- = 4+0+0 = 4

Hence the whole digit form is 764

Step 3: Now to set the above permission in the given format as below

Command: chmod 764 file1.txt / college

B] To change the permission using the characters only. instead of using or converting into them in digit formats:

chmod u+rwx, g+rwx, o+rwx file.txt

OR

chmod u+rwx

chmod g+rwx

chmod o+rwx

Practical based on above:

Example1: To set the permission rwx to the owner / user.

➔**Command:** chmod u+rwx college

Example 2: To set permission rw- to the Group.

➔**Command:** chmod g+rw-x college

Example 3: To set permission -wx to the others.

➔**Command:** chmod o+r-w-x college

C] To change ownership name of the User and Group

For Owner: chown <username> < filename>

For Group: chgrp <username> <filename>

Practical based on above:

Example 1: To change the ownership name of Owner/User column.

➔

Syntax: chown <write /give the name which you want to set> <name of the file / directory>

Command: chown student file1.txt

whereas,

1. Before setting the name of the Owner/user name first create the user in the /home directory of that name and then change the ownership of User and Group.
2. chown: Means ch: change and own: ownership.
3. chgrp: Means ch: change and grp: Group

Example 2: To change the ownership name of Group column.

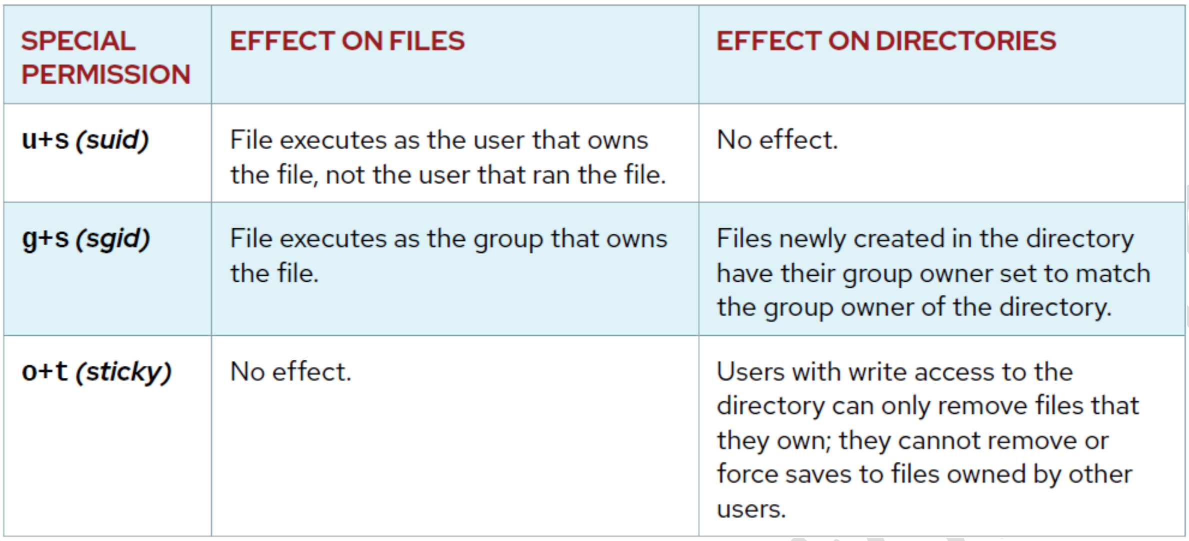
➔

Syntax: chown <write /give the name which you want to set> <name of the file / directory>

Command: chgrp root file1.txt

D] Special Permissions:

Chart as below:



SPECIAL PERMISSION	EFFECT ON FILES	EFFECT ON DIRECTORIES
u+s *(suid)*	File executes as the user that owns the file, not the user that ran the file.	No effect.
g+s *(sgid)*	File executes as the group that owns the file.	Files newly created in the directory have their group owner set to match the group owner of the directory.
o+t *(sticky)*	No effect.	Users with write access to the directory can only remove files that they own; they cannot remove or force saves to files owned by other users.

Setting Special Permissions:

Symbolically: setuid = u+s; setgid = g+s; sticky = o+t

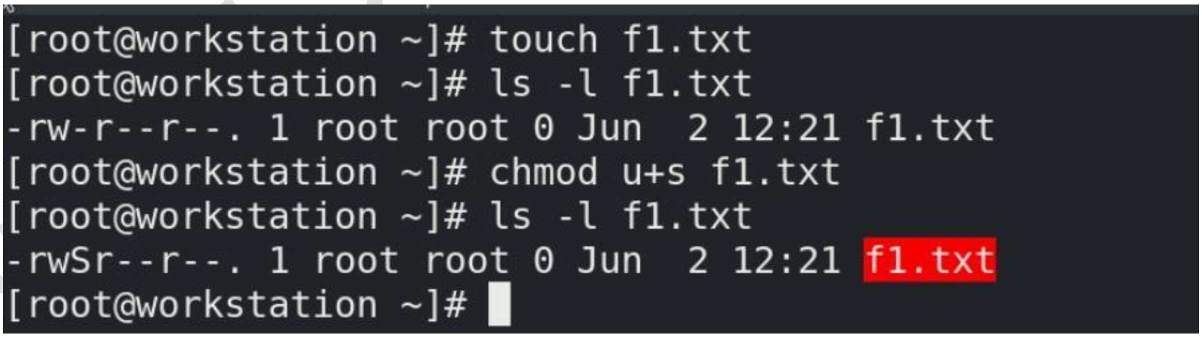
Numerically (fourth preceding digit): setuid = 4; setgid = 2; sticky = 1

Practical based on above:

SUID (u+s):

Example1: If you want to set SUID bit by passing u+s to the chmod command:

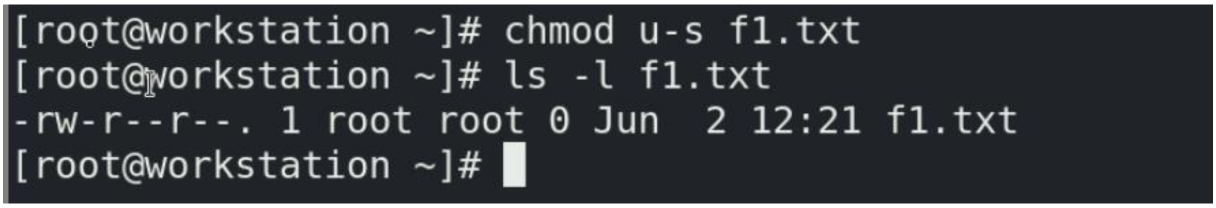
➔



```
[root@workstation ~]# touch f1.txt
[root@workstation ~]# ls -l f1.txt
-rw-r--r--. 1 root root 0 Jun  2 12:21 f1.txt
[root@workstation ~]# chmod u+s f1.txt
[root@workstation ~]# ls -l f1.txt
-rwSr--r--. 1 root root 0 Jun  2 12:21 f1.txt
[root@workstation ~]# 
```

Example 2: If you want to remove SUID bit by passing u-s to the chmod command:

➔

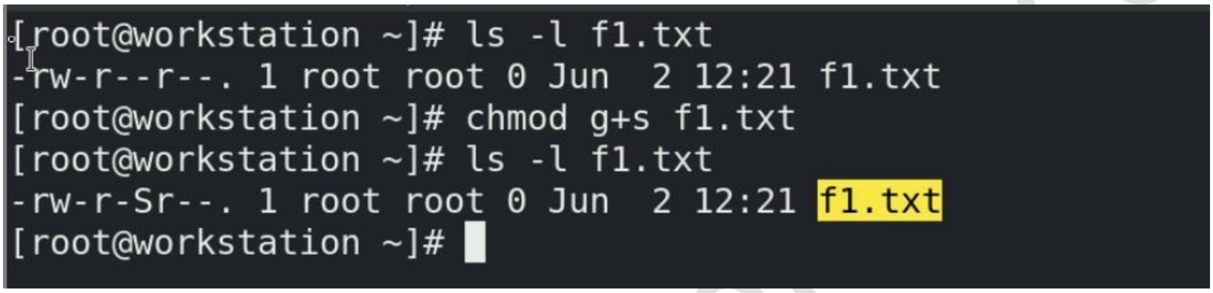


```
[root@workstation ~]# chmod u-s f1.txt
[root@workstation ~]# ls -l f1.txt
-rw-r--r--. 1 root root 0 Jun  2 12:21 f1.txt
[root@workstation ~]# 
```

GUID (g+s):

Example1: If you want to set GUID bit by passing g+s to the chmod command:

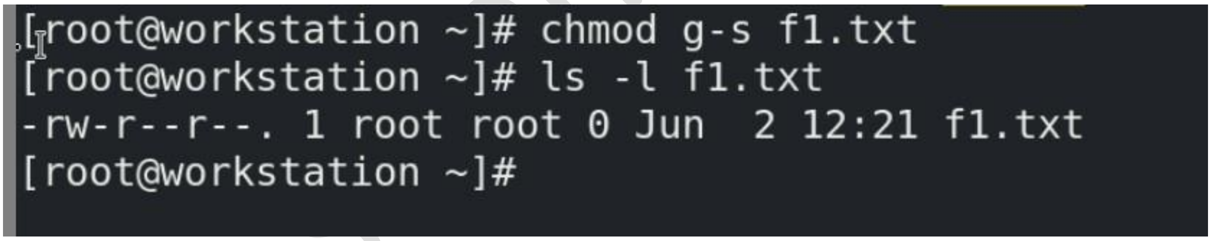
➔



```
[root@workstation ~]# ls -l f1.txt
-rw-r--r--. 1 root root 0 Jun  2 12:21 f1.txt
[root@workstation ~]# chmod g+s f1.txt
[root@workstation ~]# ls -l f1.txt
-rw-r-Sr--. 1 root root 0 Jun  2 12:21 f1.txt
[root@workstation ~]# 
```

Example 2: If you want to remove to set GUID bit by passing g-s to the chmod command:

➔



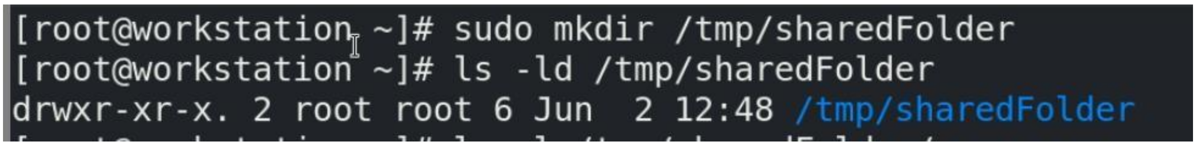
```
[root@workstation ~]# chmod g-s f1.txt
[root@workstation ~]# ls -l f1.txt
-rw-r--r--. 1 root root 0 Jun  2 12:21 f1.txt
[root@workstation ~]#
```

The Sticky Bit:

Example 1:

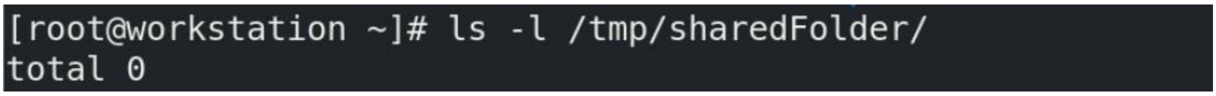
➔

Step 1: Let start by creating a shared folder where everyone has read, write, and execute permission



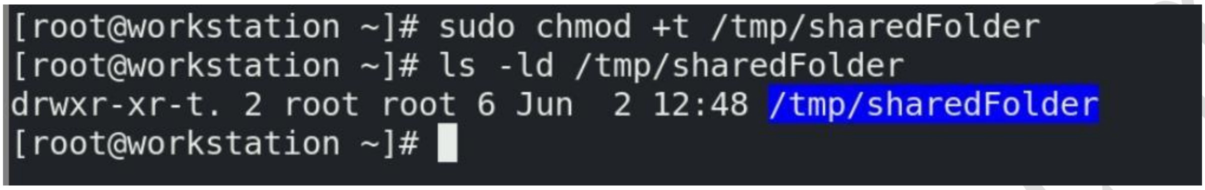
```
[root@workstation ~]# sudo mkdir /tmp/sharedFolder
[root@workstation ~]# ls -ld /tmp/sharedFolder
drwxr-xr-x. 2 root root 6 Jun  2 12:48 /tmp/sharedFolder
```

Step 2: Inside this shared folder, it is possible to remove directory/files of other users



```
[root@workstation ~]# ls -l /tmp/sharedFolder/
total 0
```

Step 3: Now let's set the sticky bit on the sharedFolder.

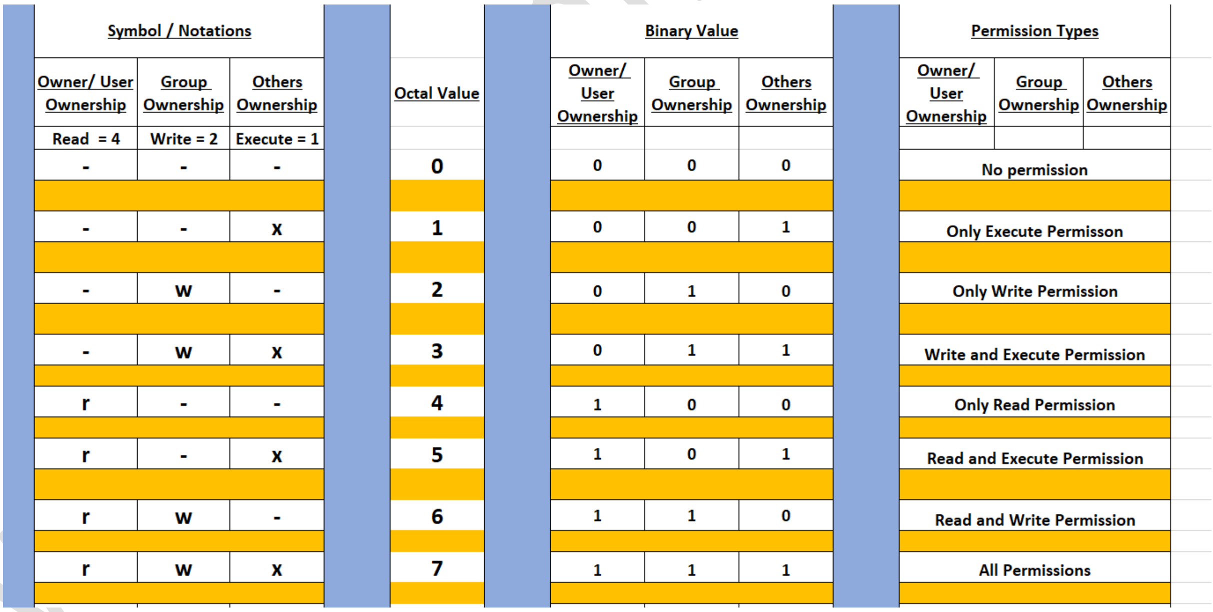


```
[root@workstation ~]# sudo chmod +t /tmp/sharedFolder
[root@workstation ~]# ls -ld /tmp/sharedFolder
drwxr-xr-t. 2 root root 6 Jun  2 12:48 /tmp/sharedFolder
[root@workstation ~]#
```

As you notice "t" letter instead of usual "x" in execute permission for the others. This letter "t" indicates that a sticky bit has been set for the file or directory in question.

E] Default Permissions using umask:

Chart as below:



Symbol / Notations			Octal Value	Binary Value			Permission Types		
Owner/ User Ownership	Group Ownership	Others Ownership		Owner/ User Ownership	Group Ownership	Others Ownership	Owner/ User Ownership	Group Ownership	Others Ownership
Read = 4	Write = 2	Execute = 1							
-	-	-	0	0	0	0	No permission		
-	-	x	1	0	0	1	Only Execute Permisson		
-	w	-	2	0	1	0	Only Write Permission		
-	w	x	3	0	1	1	Write and Execute Permission		
r	-	-	4	1	0	0	Only Read Permission		
r	-	x	5	1	0	1	Read and Execute Permission		
r	w	-	6	1	1	0	Read and Write Permission		
r	w	x	7	1	1	1	All Permissions		

Practical based on above:

Example 1 (Umask for Directory): Set and Update the default umask value for directory?

➔

Step 1: Firstly, set the umask value such as umask 543.

Now, it doesn't mean that 543 value is allocated as

5: for the owner,

4: for the group members and

3: for the others.

But the value we pass an argument is subtracted from the max/full permission set.

Note 1: Below are the two full permission sets:

- **File = The full permission set for a file is 666 (read/write permission for all)**
- **Directory = the full permission set for a directory is 777 (read/write/execute)**

Note 2: The files cannot be given execution permissions by default as it can cause a security concern, and Linux systems are pretty much known for their amazing security, so that wouldn't be good.

So, once we have set the umask value to 543, let's see what happens when we make a directory (7-7-7) and a file (6-6-6)

Step 2: Now make a directory using the below command

Command: mkdir dir1

When we make a new directory, the permissions will be calculated as (full permissions for directory) – (umask value) i.e.,

```
  777
– 543
-----
  234
```

Whereas,

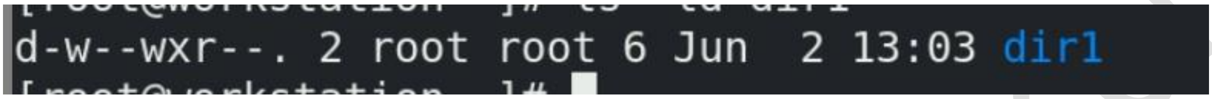
2: for the owner. The Binary value for the owner is "010". Hence, permission type will be "Only Write Permission".

3: for the group members. The Binary value for the group is “011”. Hence, permission type will be “Write and Execute Permission”.

4: for the everyone else. The Binary value for this is “100”.

Hence, permission type will be “Only Read Permission”.

Step 3: Hence, the output of above directory is as below:



```
[root@workstation ~]# ls -ld dir1
d-w--wxr--. 2 root root 6 Jun  2 13:03 dir1
[root@workstation ~]# 
```

Example 2: (Umask for File): Set and Update the default umask value for file?

➔

Step 1: Firstly, set the umask value such as umask 543.

Now, it doesn’t mean that 543 value is allocated as

5: for the owner,

4: for the group members and

3: for the others.

But the value we pass an argument is subtracted from the max/full permission set.

Note 1: Below are the two full permission sets:

- **File = The full permission set for a file is 666 (read/write permission for all)**
- **Directory = The full permission set for a directory is 777 (read/write/execute)**

Note 2: The files cannot be given execution permissions by default as it can cause a security concern, and Linux systems are pretty much known for their amazing security, so that wouldn’t be good.

So, once we have set the umask value to 543, let’s see what happens when we make a directory (7-7-7) and a file (6-6-6)

Step 2: Now make a file using the below command

Command: mkdir dir1

When we make a new directory, the permissions will be calculated as (full permissions for file) – (umask value) i.e.,

$$\begin{array}{r} 666 \\ -\ 543 \\ \hline 123 \end{array}$$

Whereas,

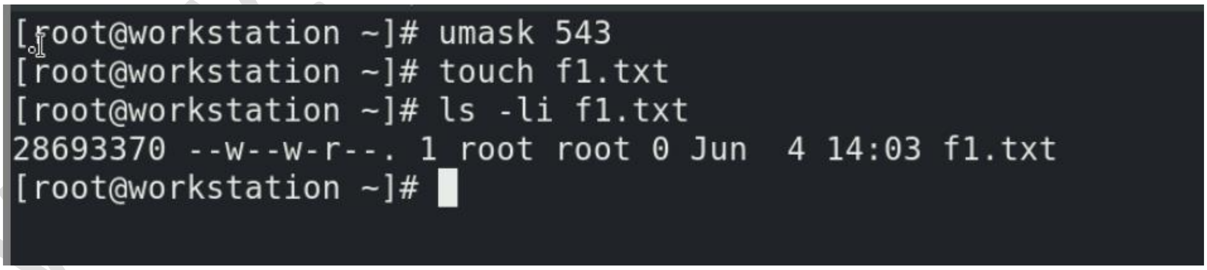
1: for the owner: The Binary value for the owner is "001" in binary, but in Linux doesn't give execute permission to the file. Hence, value promoted by one and we get 010. And finally, "write" permission will be granted to the owner.

2: for the group members: The Binary value for the group is "010". Hence, permission type will be "Write Permission".

3: for the everyone else. The Binary value for this is "011" in binary, but in Linux again execute permission cannot be provided.

Hence, value promoted one more time, and we will get "100". And finally, "read" permission will be granted to everyone.

Step 3: Hence, the output of above directory is as below:



```
[root@workstation ~]# umask 543
[root@workstation ~]# touch f1.txt
[root@workstation ~]# ls -li f1.txt
28693370 --w--w-r--. 1 root root 0 Jun  4 14:03 f1.txt
[root@workstation ~]# 
```

Difference between "chmod" and "umask" command are as below:

Command: chmod	Command: umask
This command must be used on files that are already present, it	It can only be used on new files i.e., while creating new files, any

is used to change the access permissions of files that have been created earlier.	files created prior to using the umask command will have no effect.

END