**File Permissions**

File Permissions describe the allowed operations by various users.

With respect to file permissions, all users are categorized into the following 4 types.

**User Categories->**

user/owner Represented by 'u'

group Represented by 'g'

others Represented by 'o'

all Represented by 'a'

**Permission Types->**

*Number Permission:*

Follow link:- https://ftp.kh.edu.tw/Linux/Redhat/en\_6.2/doc/gsg/s1-navigating-chmodnum.htm

0 No permission

1 Execute

2 Write

3 Execute and Write

4 Read

5 Read and Execute

6 Read and Write

7 Read, Write and Execute

*character Permissions:*

Read (`r`): Allows the viewing of file contents and listing of directory contents.

Write (`w`): Permits the modification of a file's contents or the creation, deletion, and renaming of files in a directory.

Execute (`x`): Allows the execution of a file (for executable programs) or traversal of a directory.

**Operations related to permissions:**





We can perform the following 3 operations.

+ Add a particular permission to user|group|other|all

- Remove a particular permission to user|group|other|all

= Assignment a particular permission to user|group|other|all

**Example->** `drwxr-xr-x 1 john john 512 Nov 2 01:48 xyz`

The string `drwxr-xr-x 1 john john 512 Nov 2 01:48 xyz` represents the metadata (permissions, ownership, size, modification time, and name) of a file or directory in a Linux or Unix-like operating system. Let's break down the meaning of each character :-

1. `d`: This indicates that it is a directory. If it were a regular file, it would be represented by a hyphen (`-`) in this position.

2. `rwx` (first set): These characters represent the permissions for the owner of the file/directory.

- `r`: Read permission (the owner can read the contents of the file/directory).

- `w`: Write permission (the owner can write or modify the file/directory).

- `x`: Execute permission (the owner can execute the file or enter the directory).

3. `r-x` (second set): These characters represent the permissions for the group associated with the file/directory.

- `r`: Read permission for the group.

- `-`: No write permission for the group.

- `x`: Execute permission for the group.

4. `r-x` (third set): These characters represent the permissions for others (users not in the owner's group).

- `r`: Read permission for others.

- `-`: No write permission for others.

- `x`: Execute permission for others.

5. `1`: This indicates the number of hard links to the file or directory. In this case, there is one hard link.

6. `john`: This is the name of the owner of the file/directory.

7. `john`: This is the name of the group associated with the file/directory.

8. `512`: This is the size of the file or directory in bytes.

9. `Nov 2 01:48`: This represents the last modification time of the file/directory (November 2nd, 01:48).

10. `xyz`: This is the name of the file or directory.

So, in summary, the example you provided is a directory named "xyz" owned by the user "john," belonging to the group "john," with read, write, and execute permissions for the owner, read and execute permissions for the group, and read and execute permissions for others.

**File permissions commands:-**

**`chmod` Command:**

Symbolic Mode:

```bash

Example: Give read and write permissions to the owner

chmod u+rw filename

```

Numeric Mode:

```bash

Example: Give read, write, and execute permissions to the owner

chmod 700 filename

```

**File Ownership Commands:**

Only root user can perform this activity.

**`chown` Command:**

Change Owner:

```bash

Example: Change the owner of a file to a newuser

chown newuser filename

```

Change Owner and Group:

```bash

Example: Change the owner to newuser and the group to newgroup

chown newuser:newgroup filename

```

**`chgrp` Command:**

Only root user can perform this activity.

Change Group:

```bash

Example: Change the group of a file to newgroup

chgrp newgroup filename

```

**Examples->**

User Permissions + Group Permissions + Others Permissions

order is important

Eg 1: $ chmod u+x demo.txt

adding execute permission to the user

Eg 2: $ chmod u+w,g+rw,o+r demo.txt

adding write permission to the user

adding read and write permissions to the group

adding read permission to the others

Eg 3: $chmod u+x,g-w,o+w demo.txt

adding execute permission to the user

removing write permission from the group

adding write permission to the others

Eg 4: $ chmod u=rw,g=rw,o=r demo.txt

Now user permissions: rwgroup

permission: rwothers

permission: r—

Eg 5: $ chmod a=- demo.txt

Now user permissions: ---

group permission: ---

others permission: ---

Eg 6: $ chmod a=rwx demo.txt

Now user permissions: rwx

group permission: rwx

others permission: rwx

**Read Permission to the File:-**

If the file not having read permission then we are not allowed to view content of the file.

Hence cat, head, tail, more, less commands won't work.

**Write Permission to the File:-**

If the file not having write permission, then we cannot modify the content of the file.

**Execute Permission to the File:-**

If the user not has executed permission on any file, then he cannot execute that file as a

program.

**Read Permission to the Directory:-**

If the user has read permission on any directory, then he can list out the contents of that

directory. i.e he can use ls command.

**Write Permission on the Directory:-**

If the user has write permission on any directory, then he is allowed to modify the content

of that directory. i.e he can add new files and remove existing files.

**Execute Permission to the Directory:-**

If the user not has executed permission on any directory, then he is not allowed to enter

into that directory. i.e he cannot use cd command.

Note:- If the user not having read permission on any file, then he cannot execute that file

even though he has executed permission.

**Linux vs Security:**

The virus files usually created by others.

others are not having execute permission on our directories. Hence others are not allowed

to add virus files to our directories.

Hackers are not having executed permission on our directories. Hence they cannot read

our file data.

Because of this, Linux is considered as more secured operating system.

Linux follows 2 levels of security.

1st level: login with credentials

2nd level: File and Directory permissions

Note: We are using permission types as r,w,x and these are considered as symbolic

permissions. But we can also specify permissions by using octal number, such type of

permissions are called numeric permissions.

**Notes:**

1. Recursive Changes:

- To apply changes to files and directories recursively, use the `-R` option with `chmod`, `chown`, or `chgrp`.

```bash

Example: Recursively change permissions for all files and subdirectories

chmod -R u+rw directory

```

2. Wildcard Usage:

- You can use wildcards with `chmod` to apply changes to multiple files or directories.

```bash

Example: Give read and write permissions to all text files in the current directory

chmod u+rw .txt

```

3. Preserving Timestamps:

- When changing ownership with `chown`, you can preserve the timestamps with the `--preserve` option.

```bash

Example: Change the owner and preserve timestamps

chown --preserve newuser:newgroup filename

```

These commands and concepts are fundamental to managing file and directory permissions and ownership in Linux. Always exercise caution when modifying permissions and ownership, especially with the use of wildcard characters and recursive options, to avoid unintended consequences.

Class example:-

1286 ls

1287 cd ..

1288 ls

1289 su mukund

1290 sudo -i

1291 exit

1292 groups

1293 sudo -i

1294 cd /

1295 touch j3.txt

1296 sudo touch t5.txt

1297 ls

1298 cat /etc/passwd

1299 ls

1300 cd /home

1301 ls -l

1302 groups

1303 cat /etc/passwd

1304 awk -F":" "{print $1}" /etc/passwd

1305 awk -F ":" "{print $1}" /etc/passwd

1306 awk -F ":" '{print $1}' /etc/passwd

1307 passwd john

1308 awk -F ":" '{print $1}' /etc/passwd

1309 sudo deluser garima

1310 sudo -i

1311 history

1312 who

1313 man who

1314 who

1315 w

1316 last

1317 ls -l

1318 groups

1319 addgroup gg

1320 sudo addgroup group1

1321 groups

1322 awk -F ":" '{print $1}' /etc/passwd

1323 groups

1324 groups root

1325 groups dhruv

1326 id john

1327 ls -l

1328 chgrp group1 xyz

1329 sudo chgrp group1 xyz

1330 ls -l

1331 sudo chgrp group7 xyz

1332 groups

1333 ls groups

1334 chmod u-w , g-r , o-r xyz

1335 chmod u-w,g-r,o-r xyz

1336 ls -l

1337 chmod u+x xyz

1338 ls -l

1339 chmod u-x xyz

1340 ls -l

1341 chmod u-x s.sh

1342 ls -l

1343 delgroup group1

1344 sudo delgroup group1

1345 chgrp group1 test.txt

1346 groups1

1347 groups

1348 cd ..

1349 touch f4.txt

1350 sudo touch f4.txt

1351 ls

1352 ls -l

1353 history

1354 # Example: Edit a system file with elevated privileges

1355 sudo nano /etc/system-file.conf

1356 cat /etc/system-file.conf

1357 sudo cat /etc/system-file.conf

1358 cat /etc/system-file.conf

1359 cd john

1360 ls -l

1361 groups

1362 # Example: Change the owner to newuser and the group to newgroup

1363 chown newuser:newgroup filename

1364 chown john:netdev r1.txt

1365 sudo chown john:netdev r1.txt

1366 ls -l

1367 chmod 444 test.txt

1368 ls -l

1369 chmod 440 test.txt

1370 ls -l

1371 chmod 666 test.txt

1372 ls -l

1373 chmod 555 test.txt

1374 ls -l

1375 exit

1376 passwd

1377 man passwd

1378 exit

1379 history