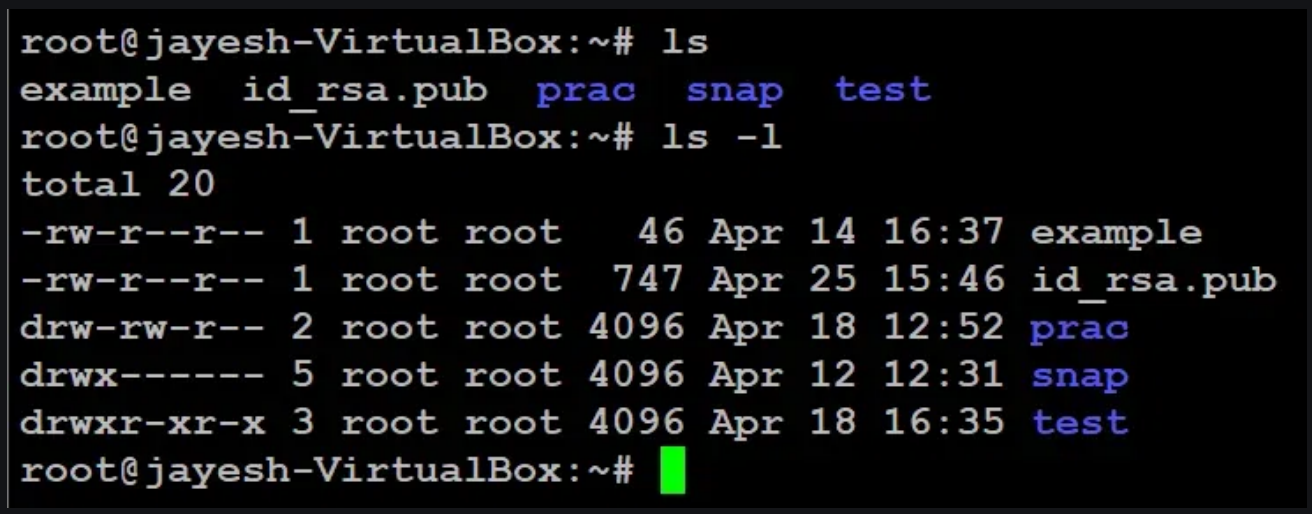
Permissions in Linux

Linux is a multi-user operating system, so it has security to prevent people from accessing each other's confidential files.
When you execute a "ls" command, you are not given any information about the security of the files, because by default "ls" only lists the names of files. You can get more information by using an "option" with the "ls" command. All options start with a '-'. For example, to execute "ls" with the "long listing" option, you would type ls -l When you do so, each file will be listed on a separate line in a long format. There is an example in the window below.

Syntax:

```
ls
```

```
ls -l
```



ls -l

There's a lot of information in those lines.

1. The first character = **‘-‘**, which means it’s a file
   ‘d’, which means it’s a directory.
2. The next nine characters = (rw-r–r–) show the security
3. The next column shows the owner of the file. (Here it is `root`)
4. The next column shows the group owner of the file. (Here it is `root` which has special access to these files)
5. The next column shows the size of the file in bytes.
6. The next column shows the date and time the file was last modified.
7. Last Column = File_name or Directory_name. (For example, here are: prac, snap, test, example)

Security permissions in Linux

First, you must think of those nine characters as three sets of three characters (see the box at the bottom). Each of the three “rwx” characters refers to a different operation you can perform on the file.

```
---     ---     ---
rwx     rwx     rwx
user    group   other
```

Read, write, and execute.

Letters	Definition
‘r’	“read” the file’s contents.
‘w’	“write”, or modify, the file’s contents.
‘x’	“execute” the file. This permission is given only if the file is a program.

Symbols: `+`, `-` and `=`

Operators	Definition
`+`	Add permissions

Operators	Definition
`-`	Remove permissions
`=`	Set the permissions to the specified values

User, group, and others

Reference	Class	Description
`u`	user	The user permissions apply only to the owner of the file or directory, they will not impact the actions of other users.
`g`	group	The group permissions apply only to the group that has been assigned to the file or directory, they will not affect the actions of other users.
`o`	others	The other permissions apply to all other users on the system, this is the permission group that you want to watch the most.
`a`	All three	All three (owner, groups, others)

Reading the security permissions

For example: “rw- r-x r–“

- **“rw-“**: the first three characters `rw-`. This means that the owner of the file can “read” it (look at its contents) and “write” it (modify its contents). we cannot execute it because it is not a program but a text file.
- **“r-x”**: the second set of three characters “r-x”. This means that the members of the group can only read and execute the files.
- **“r–“**: The final three characters “r–” show the permissions allowed to other users who have a UserID on this Linux system. This means anyone in our Linux world can read but cannot modify or execute the files’ contents.

Changing security permissions

The command you use to change the security permissions on files is called "**chmod**", which stands for "change mode" because the nine security characters are collectively called the security "mode" of the file.
An example will make this clearer.

For example, if you want to give "execute" permission to the world ("other") for file "xyz.txt", you will start by typing.

```
chmod o
```

Now you would type a '+' to say that you are "adding" permission.

```
chmod o+
```

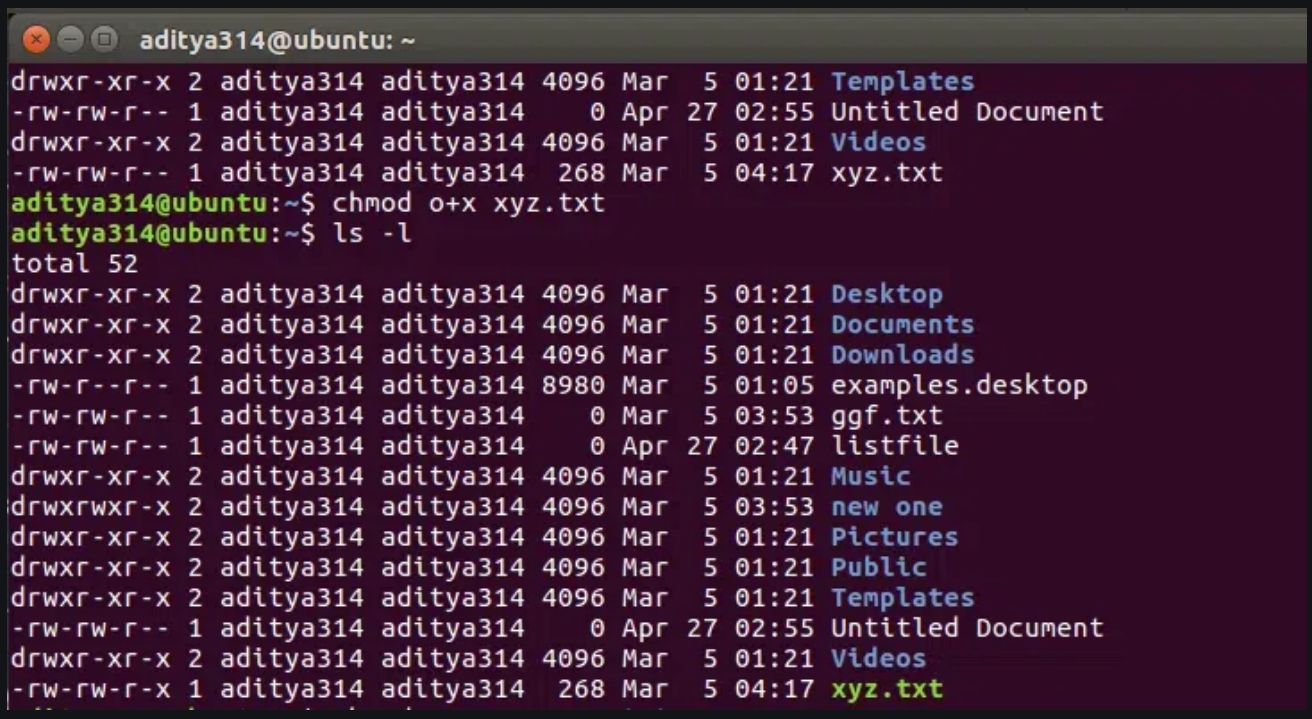
Then you would type an 'x' to say that you are adding "execute" permission.

```
chmod o+x
```

Finally, specify which file you are changing.

```
chmod o+x xyz.txt
```

You can see the change in the picture below.

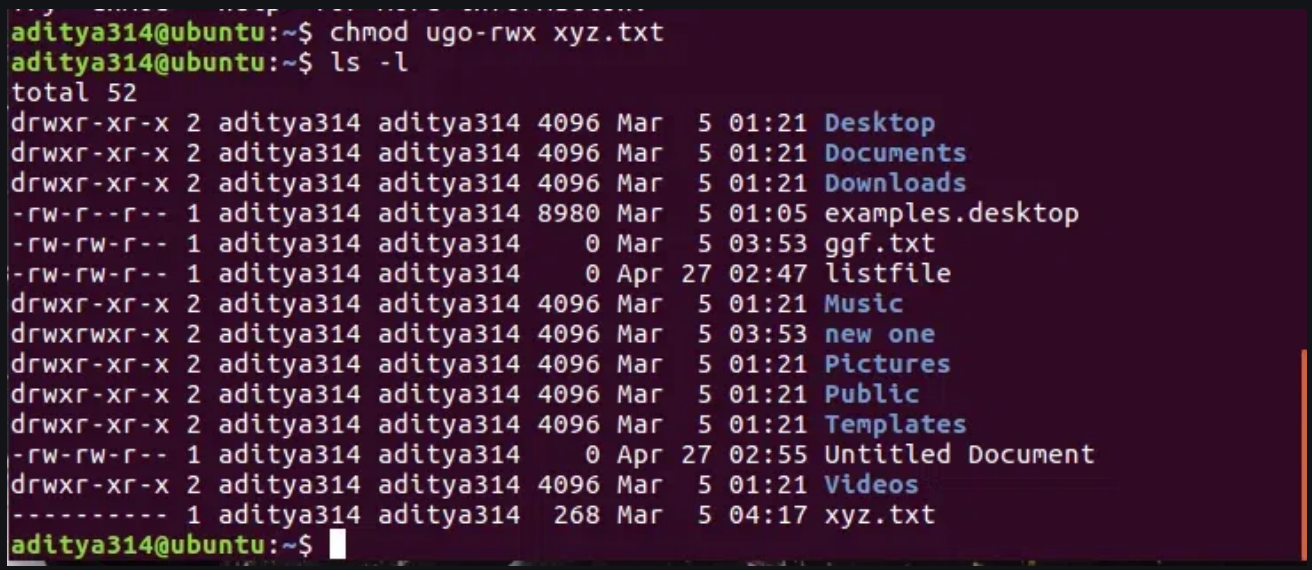


chmod o+x xyz.txt

You can also change multiple permissions at once. For example, if you want to take all permissions away from everyone, you would type.

```
chmod ugo-rwx xyz.txt
```

The code above revokes all the read(r), write(w), and execute(x) permission from all user(u), group(g), and others(o) for the file xyz.txt which results in this.



multiple use

Another example can be this:

```
chmod ug+rw,o-x abc.mp4
```

The code above adds read(r) and write(w) permission to both user(u) and group(g) and revoke execute(x) permission from others(o) for the file abc.mp4.

Something like this:

```
chmod ug=rx,o+r abc.c
```

assigns read(r) and execute(x) permission to both user(u) and group(g) and add read permission to others for the file abc.c.

There can be numerous combinations of file permissions you can invoke revoke and assign. You can try some on your Linux system.

The octal notations

You can also use octal notations like this.

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwx

octal notations

Using the octal notations table instead of ‘r’, ‘w’, and ‘x’. Each digit octal notation can be used for either of the group ‘u’, ‘g’, or’o’.

So, the following work is the same.

```
chmod ugo+rwx [file_name]
chmod 777 [file_name]
```

Both of them provide full read write and execute permission (code=7) to all the group.

The same is the case with this.

```
chmod u=r,g=wx,o=rx [file_name]
chmod 435 [file_name]
```

Both the codes give read (code=4) user permission, write and execute (code=3) for the group and read and execute (code=5) for others.

And even this...

```
chmod 775 [file_name]
chmod ug+rwx,o=rx [file_name]
```

Both the commands give all permissions (code=7) to the user and group, read and execute (code=5) for others.

Conclusion

The article explains how to view and change file permissions in Linux using the “ls” and “chmod” commands. It describes the three sets of permissions (read, write, and execute) and how to apply them to users, groups, and others. The article also provides examples of how to change permissions using both symbolic and octal notation.

Last Updated : 10 May, 2023

Similar Reads

1. SetUID, SetGID, and Sticky Bits in Linux File Permissions