# OPERATING SYSTEM 1

Lecture 4

Eng. Joud Khattab

# FILE PERMISSIONS

## File Permissions

- Definition.
- The Permission Indicators.
- File/Directory Access Modes.
- Changing Permissions.
- Changing Owners and Groups.

## Definition

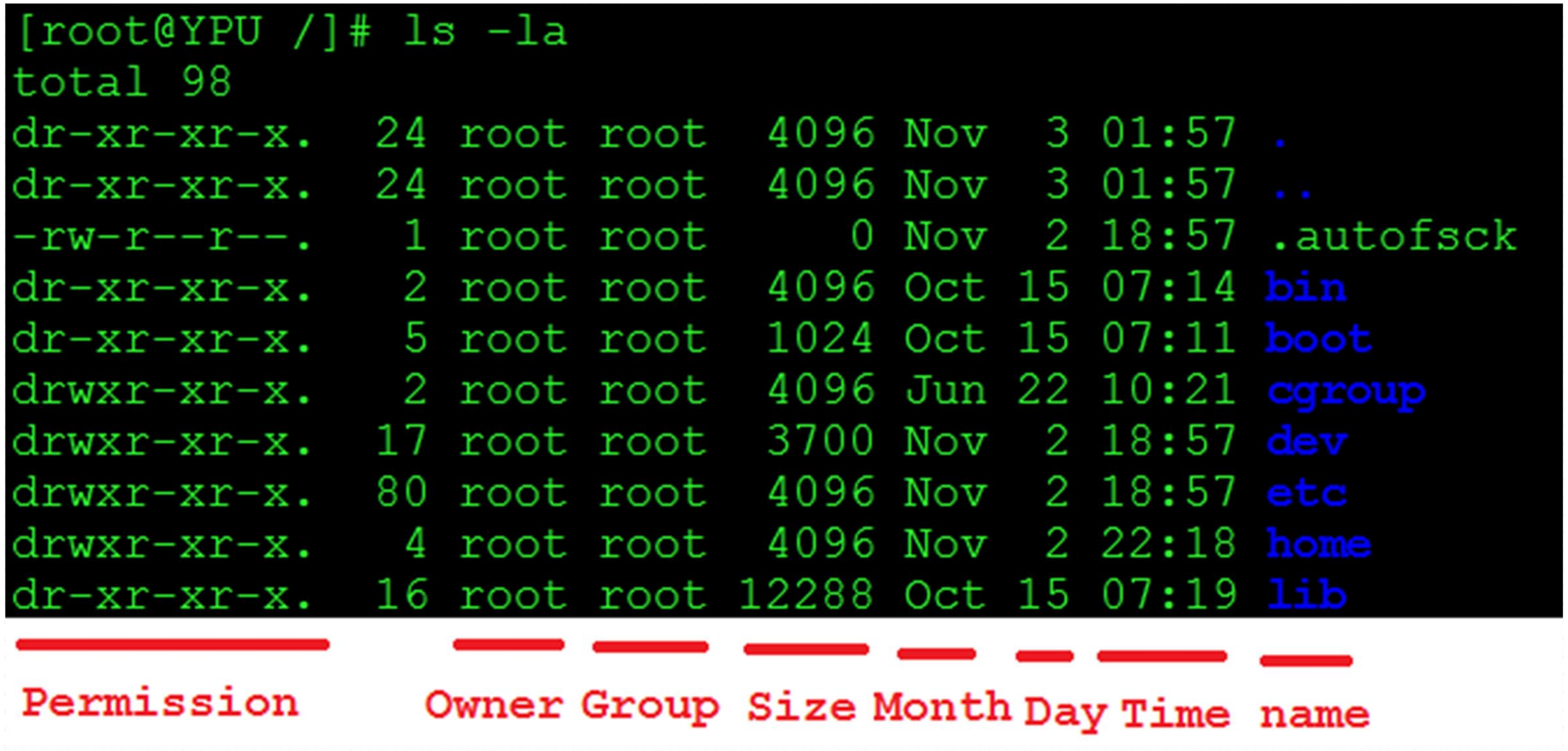
- File ownership is an important component of Unix that provides a secure method for storing files. Every file in Unix has the following attributes:
  - **Owner permissions:**
    - Actions the owner of the file can perform on the file.
  - **Group permissions:**
    - Actions a user, who is a member of the group that a file belongs to, can perform on the  file.
  - **Other (world) permissions:**
    - Actions all other users can perform on the file.

## The Permission Indicators

- While using ls -l command, it displays various information related to file permission as follows:
  - -rwxr-xr-- 1 lab2 users 1024 Nov 2 00:10 myfile
  - drwxr-xr--- 1 lab2 users 1024 Nov 2 00:10 mydir

- Here, the first column represents different access modes, the permission associated with a file or a directory.

## The Permission Indicators



```
[root@YPU /]# ls -la
total 98
dr-xr-xr-x.  24 root root  4096 Nov  3 01:57 .
dr-xr-xr-x.  24 root root  4096 Nov  3 01:57 ..
-rw-r--r--.   1 root root     0 Nov  2 18:57 .autofsck
dr-xr-xr-x.   2 root root  4096 Oct 15 07:14 bin
dr-xr-xr-x.   5 root root  1024 Oct 15 07:11 boot
drwxr-xr-x.   2 root root  4096 Jun 22 10:21 cgroup
drwxr-xr-x.  17 root root  3700 Nov  2 18:57 dev
drwxr-xr-x.  80 root root  4096 Nov  2 18:57 etc
drwxr-xr-x.   4 root root  4096 Nov  2 22:18 home
dr-xr-xr-x.  16 root root 12288 Oct 15 07:19 lib
```

# The Permission Indicators

- 3 types of file/directory permissions: (read, write, execute)
- 10 bit format from 'ls -l' command

| 1 | 2 3 4 | 5 6 7 | 8 9 10 |
|---|---|---|---|
| File type | Owner | Group | Others |

- eg. drwxrw-r-- means owner has all three permissions, group has read and write, others have only read permission.

- read permission = 4, write = 2, execute = 1
- eg. rwx rw- r-- = 764

  rw- rwx -wx = 673

## File Access Modes

- The permissions of a file are the first line of defense in the security of a Unix system.
- The basic building blocks of Unix permissions are the read, write, and execute permissions, which have been described below:
  - **Read:** Grants the capability to read, i.e., view the contents of the file.
  - **Write:** Grants the capability to modify, or remove the content of the file.
  - **Execute:** User with execute permissions can run a file as a program.

## Directory Access Modes

- Directory access modes are listed and organized in the same manner as any other file.
- There are a few differences that need to be mentioned:
  - **Read:** Access to a directory means that the user can read the contents. The user can look at the filenames inside the directory.
  - **Write:** Access means that the user can add or delete files from the directory.
  - **Execute:** Executing a directory doesn't really make sense, so think of this as a traverse permission. A user must have execute access to the bin directory in order to execute the ls or the cd command.

## CHANGING PERMISSIONS

# Changing Permissions

- chmod : **ch**ange **mode**
  - Changes the permission of a file.
  - **Syntax:** chmod [OPTION][MODE][FILE]

| | |
|---|---|
| **u** | User who owns the file |
| **g** | Group that owns the file |
| **o** | Other |
| **a** | All (User+Group+Other) |
| **r** | Read the file |
| **w** | Write or edit the file |
| **x** | Execute or run the file as a program |

  - **Examples:**
    - chmod 744 myfile.txt
    - chmod u+rwx myfile.txt
    - chmod u-x myfile.txt

## Changing Permissions (First Method)

- The easiest way for a beginner to modify file or directory permissions is to use the symbolic mode.
- With symbolic permissions you can add, delete, or specify the permission set you want by using the operators in the following table.

| chmod Operator | Description |
|---|---|
| + | Adds the designated permission(s) to a file or directory. |
| - | Removes the designated permission(s) from a file or directory. |
| = | Sets the designated permission(s). |

## Changing Permissions (First Method) Example

- Here's an example using testfile. Running ls -l on the testfile shows that the file’s permissions are as follows:
  - $ls -l testfile
  - -rwxrwxr-- 1 amrood users 1024 Nov 2 00:10 testfile
- Then each example chmod command from the preceding table is run on the testfile, followed by ls –l, so you can see the permission changes:

## Changing Permissions (First Method) Example

- $chmod o+wx testfile
- $ls -l testfile
- -rwxrwxrwx 1 amrood users 1024 Nov 2 00:10 testfile

- $chmod u-x testfile
- $ls -l testfile
- -rw-rwxrwx 1 amrood users 1024 Nov 2 00:10 testfile

- $chmod g=rx testfile
- $ls -l testfile
- -rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile

## Changing Permissions (First Method) Example

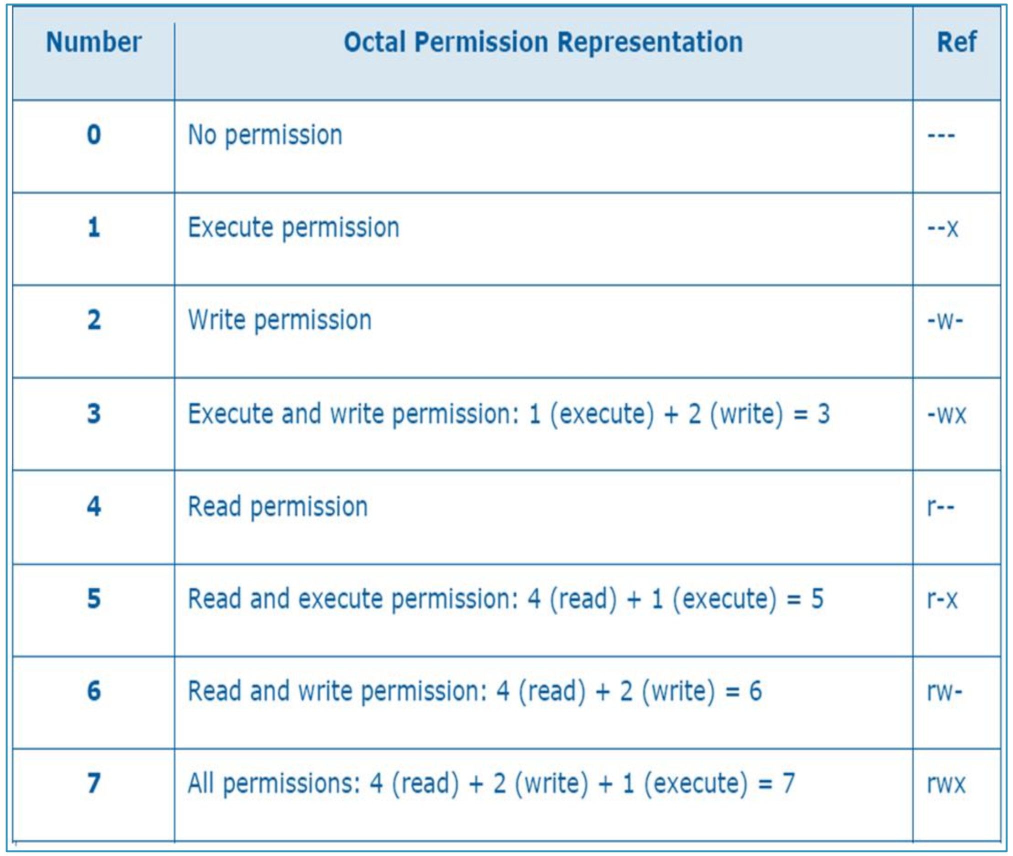
- Here's how you can combine these commands on a single line:
  - `$chmod o+wx,u-x,g=rx testfile`
  - `$ls -l testfile`
  - `-rw-r-xrwx 1 amrood users 1024 Nov 2 00:10 testfile`

## Changing Permissions (Second Method)

- The second way to modify permissions with the chmod command is to use a number to specify each set of permissions for the file.
- Each permission is assigned a value, as the following table shows, and the  total of each set of permissions provides a number for that set.

## Changing Permissions (Second Method)

- The second way to modify permission with the chmod command, is to use a number to specify each set of permissions for the file.
- Each permission is assigned a value, as the following table shows, and the total of each set of permissions provides a number for that set.



| Number | Octal Permission Representation | Ref |
|---|---|---|
| 0 | No permission | --- |
| 1 | Execute permission | --x |
| 2 | Write permission | -w- |
| 3 | Execute and write permission: 1 (execute) + 2 (write) = 3 | -wx |
| 4 | Read permission | r-- |
| 5 | Read and execute permission: 4 (read) + 1 (execute) = 5 | r-x |
| 6 | Read and write permission: 4 (read) + 2 (write) = 6 | rw- |
| 7 | All permissions: 4 (read) + 2 (write) + 1 (execute) = 7 | rwx |

## Changing Permissions (Second Method) Example

- Here's an example using the testfile. Running ls -l on the testfile shows that  the file’s permissions are as follows:
  - $ls -l testfile
  - -rwxrwxr-- 1 amrood users 1024 Nov 2 00:10 testfile
- Then each example chmod command from the preceding table is run on the testfile, followed by ls –l, so you can see the permission changes:

## Changing Permissions (Second Method) Example

- $ chmod 755 testfile
- $ls -l testfile
- -rwxr-xr-x 1 amrood users 1024 Nov 2 00:10 testfile

- $chmod 743 testfile
- $ls -l testfile
- -rwxr---wx 1 amrood users 1024 Nov 2 00:10 testfile

- $chmod 043 testfile
- $ls -l testfile
- ----r---wx 1 amrood users 1024 Nov 2 00:10 testfile

## CHANGING OWNER

# Changing Owner

- chown : **ch**ange **own**er
  - Change file owner and group.
  - **Syntax:** chown [-R] owner:group [FILE]
    - **-R** Change the permission on files that are in the subdirectories of the directory that you are currently in
  - **Examples:**
    - chown -R root /home/ypu
    - chown -R root:root /home/ypu

## EXERCISE

## Exercise

1. First navigate to desktop.
   - cd Desktop
2. Make new file contains the current date.
   - date > d.txt
3. Now we need to make new user with your name and make you owner to the previous file, so enter to the root.
   - Sudo -i
4. Make new user commands:
   - useradd name
   - passwd name
5. Back to the main user:
   - Exit
6. Login as root inside the main user:
   - su root
   - sudo passwd root
   - whoami
7. Change file ownership:
   - chown name:name d.txt
8. Exit user:
   - exit