Tishk International University
Science Faculty
IT Department

Open Source OS (Linux)

Lecture 3:
Users, Groups, and Permissions Management

4th Grade - Fall Semester 2020-2021

Instructor: Alaa Ghazi

Lecture 3
Users, Groups and Permissions Management

Introduction

- Linux uses groups to help you manage users, set permissions on those users.
- Normally Linux computers have two user accounts—

1. root account, which is the super user that can access everything on the PC, make system changes, and administer other users.
2. normal users

User Accounts Files

- **/etc/passwd** This file contains the user account information for the system.
- **/etc/shadow** This file contains encrypted passwords for the user accounts.
- **/etc/group** This file contains the list of groups.
- **/etc/gshadow** each line in this file represents a record for a single group.

The Superuser (root)

- By default, one account has elevated privileges to issue any command, access any file, and perform every function, it is the Superuser, which is called **root**
- root User ID is 0 and group number is 0
- Why root account should be limited?
  - Inexperienced users can cause serious harm
  - Use of root for non-privileged tasks unnecessary and can be open to attack
  - Security and privacy violations – root can look at anyone's files
- Recommended Settings for root:
  - ❖ Disable root account locally and remotely
  - ❖ If not then disable or limit what root can do remotely
  - ❖ Ensure a strong password

Superuser Privileges

- What usually works best is short periods of superuser privilege, only when necessary
- Obtain privileges, complete task, relinquish privileges
- Most common ways are su and sudo
- Some Linux distributions such as Ubuntu disable the root account by default
- Must rely on `sudo` to obtain privilege.

su

- Short for *substitute* or *switch user*
- Syntax: `su [options]`
- After issuing command, prompted for that root's password
- A new shell opened with the superuser privileges
- Once done issuing commands, must type exit

sudo

- Allows user to issue a single command as root
- Syntax:
  ```
  sudo command
  ```
- In Ubuntu the root account is disabled by default.
- In Ubuntu the user created during installation will have certain administrative privileges, since it will be member of sudo group by default so it can run commands with superuser privileges
- The files and folders created with sudo will be owned by root

Creating and Managing User and Groups

Creating a User

Syntax: **adduser** username

example: **adduser** azad

- You will be asked for certain information which you can keep empty except full name (use same username in this course) and provide password. (recommended to use 12345)
- Whenever a new user is created a group with same name will be created automatically.

Deleting a User

- ***Syntax:*** **userdel –r username**
- ***example:*** **userdel –r azad**

Use the ***–r*** option in the command line to remove the home directory when you delete the user.

Creating/Deleting a Group

- **To create a group use groupadd like below**
- **Syntax:**

groupadd *options* *groupname*

- ***Options:***

–g Specifies a GID for the new group.
–p Specifies a password for the group.
–r Specifies that the group being created is a system group

example: **groupadd** ***groupc***

To delete a group use groupdel like below

Syntax: **groupdel** ***group_name***
example: **groupdel** ***test2***

Add/Remove a User to/from a Group

- To add an existing user account to a group on your system, use the usermod command,

```
sudo usermod -a -G groupname username
```

- For example, to add the user azad to the group groupc , use the following command:

```
sudo usermod -a -G groupc azad
```

- To view the groups the current user account is assigned to, run the groups command. You'll see a list of groups.

```
groups
```

- To remove a user from a group, use the gpasswd command with the -d option as follows.

```
sudo gpasswd -d username groupname
```

Managing ownership

Anytime a user creates a new file or directory, his or her user account is assigned as that file or directory's "owner." and the group corresponding to that user will be the file or directory's "group owner."

For example, suppose the azad user logs in to her Linux system and creates a file named azadfile1 in home directory. Because he created this file, azad is automatically assigned ownership of azadfile1.

How ownership works

- You can specify a different user and/or group as the owner of a given file or directory. To change the user who owns a file, you must use superuser privileges with sudo command.
- Using chown

✓ Using chgrp

✓ You can also view file ownership from the command line using the ls –l command

Using chown

- The chown utility can be used to change the **user** or **group** that owns a file or **directory**.

  <u>*Syntax*</u> **chown** user file or directory.

  Example: If I wanted to change the file's owner to the **azad** user, I would enter

```
chown azad myfile
```

  Note: You can use the –R option with chown to change ownership on many files at once recursively.

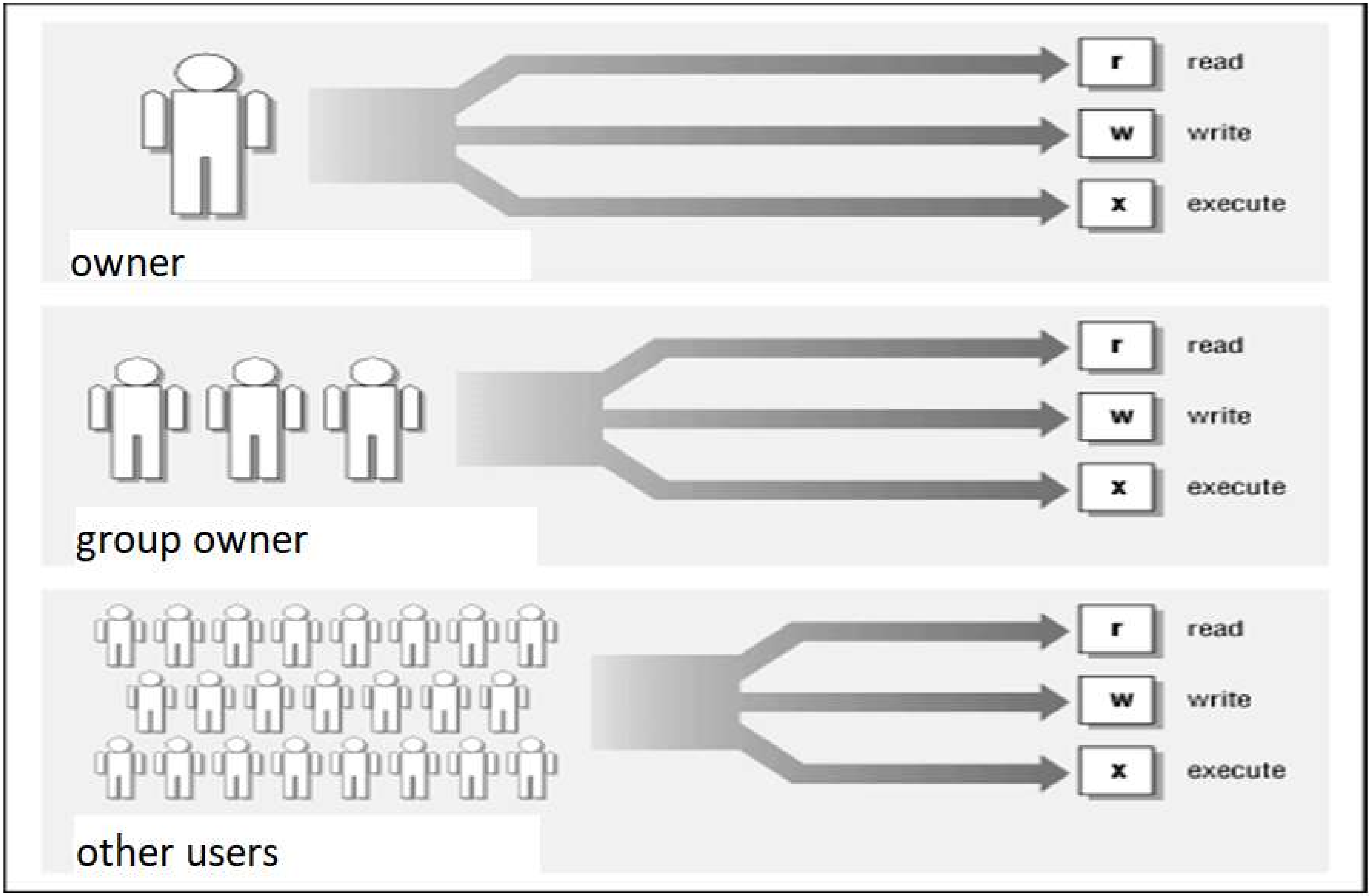
Using chgrp

- In addition to chown, you can also use **chgrp** to change the group that owns a file or directory.
- Syntax: **chgrp group file (or directory)**
- Example: **chgrp student myfile**

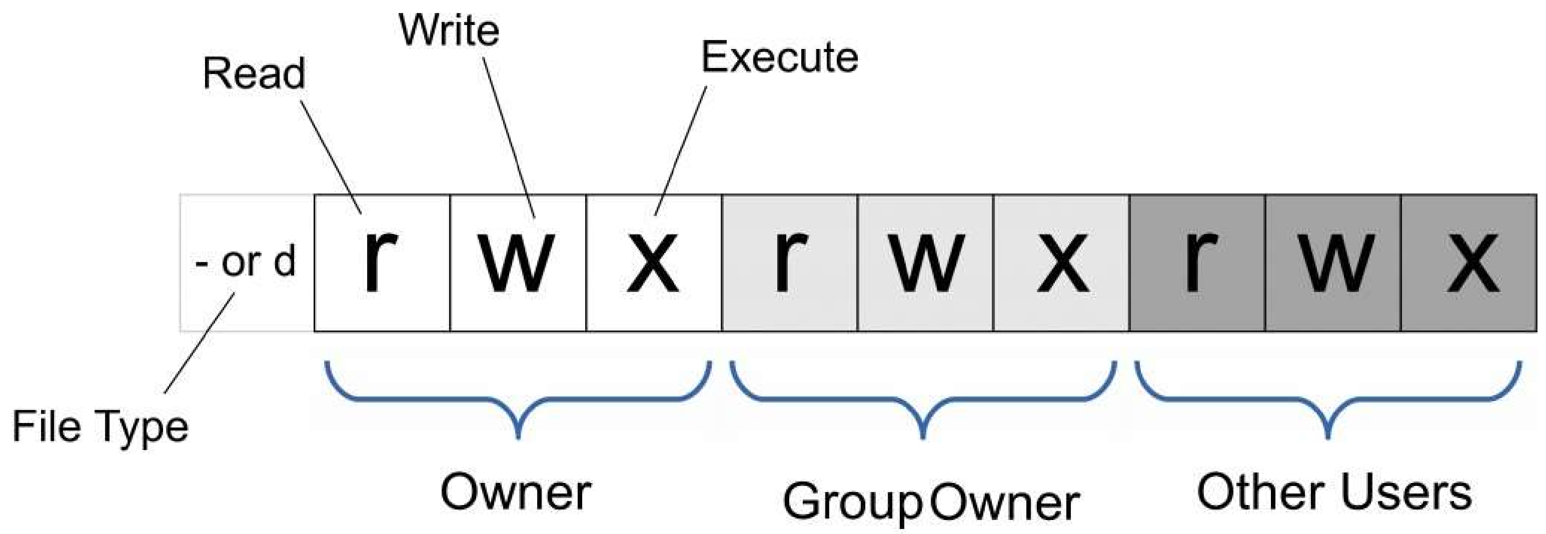
File Permissions

- On a Linux system, each file and directory is assigned access rights for the owner of the file, the members of a group of related users, and everybody else. Rights can be assigned to read a file, to write a file, and to execute a file (i.e., run the file as a program).
- Linux controls access to files on the computer through a system of “permissions.”
- <u>**Permissions**</u> are settings configured to control exactly how files on your computer are accessed and used.
- To see the permission settings for a file, we can use the ls -l command.

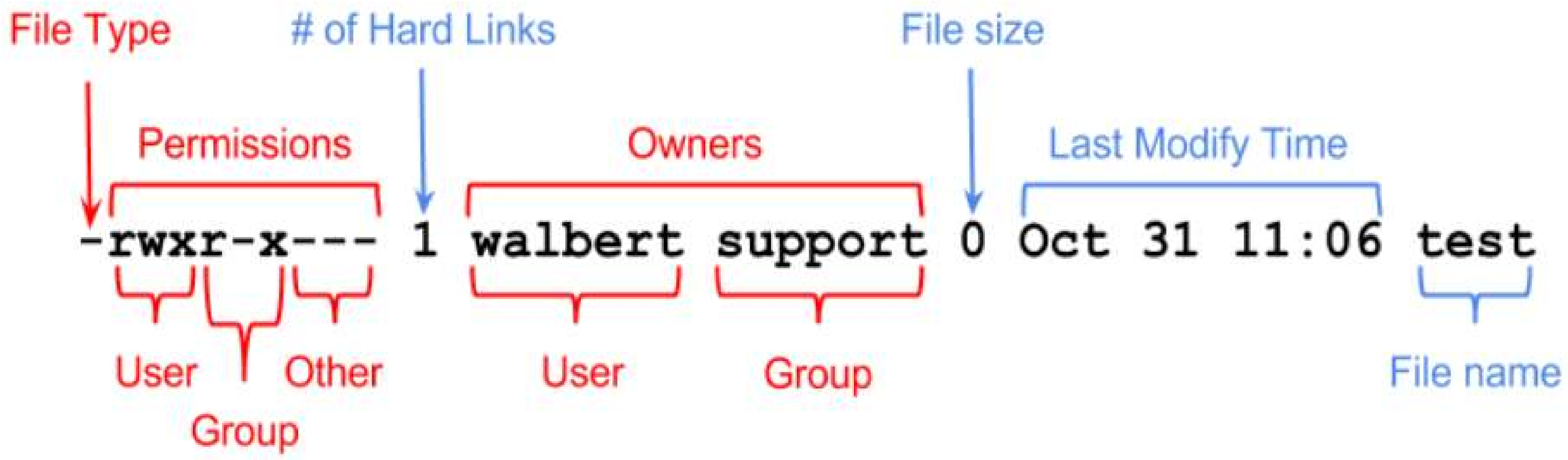
File Permissions



File Permissions



File and Directory Access Permissions



```
-rwxr-x--- 1 walbert support 0 Oct 31 11:06 test
```

chmod – changing file permissions

- The chmod command is used to change the permissions of a file or directory. To use it, you specify the desired permission settings and the file or files that you wish to modify. There are two ways to specify the permissions. In this lesson we will focus on one of these, called the ***octal notation* method**.
- Here's how it works:

```
rwx rwx rwx = 111 111 111
rw- rw- rw- = 110 110 110
rwx --- --- = 111 000 000
```

- and so on... rwx = 111 in binary = 7 rw- = 110 in binary = 6 r-x = 101 in binary = 5 r-- = 100 in binary = 4
- Now, if you represent each of the three sets of permissions (owner, group, and other) as a single digit, you have a pretty convenient way of expressing the possible permissions settings. For example, if we wanted to set myfile to have read and write permission for the owner, but wanted to keep the file private from others, we would:

```
sudo chmod 600 myfile
```

Files common settings

Value	Meaning
777	(rwxrwxrwx) No restrictions on permissions. Anybody may do anything. Generally not a desirable setting.
755	(rwxr-xr-x) The file's owner may read, write, and execute the file. All others may read and execute the file. This setting is common for programs that are used by all users.
700	(rwx------) The file's owner may read, write, and execute the file. Nobody else has any rights. This setting is useful for programs that only the owner may use and must be kept private from others.
666	(rw-rw-rw-) All users may read and write the file.
644	(rw-r--r--) The owner may read and write a file, while all others may only read the file. A common setting for data files that everybody may read, but only the owner may change.
600	(rw-------) The owner may read and write a file. All others have no rights. A common setting for data files that the owner wants to keep private.

Directory Permissions

- The **chmod** command can also be used to control the access permissions for directories. Again, we can use the octal notation to set permissions, but the meaning of the **r, w,** and **x** attributes is different:
  - **r** - Allows the contents of the directory to be listed if the x attribute is also set.
  - **w** - Allows files within the directory to be created, deleted, or renamed if the x attribute is also set.
  - **x** - Allows a directory to be entered (i.e. **cd dir**).