Homework 2: Linux User and Group Management

Môn học: Hệ điều hành Linux và ứng dụng

CS11117 - 22MMT

Sinh viên: Nguyễn Hồ Đăng Duy - 22127085

Table of Contents

Step 1: Create Project Groups and Users

1. Create two groups: `development` and `testing`

```
sudo groupadd development
sudo groupadd testing
```

Explanation:

- `groupadd` creates a new group on the system. These groups represent the two teams.

Screenshot:

2. Create users with home directories and initial group assignment

```
sudo useradd -m -G development dev_lead
sudo useradd -m -G development dev_junior
sudo useradd -m -G testing tester_lead
sudo useradd -m -G testing tester_junior
sudo useradd -m guest_user
```

Explanation:

- `-m` creates a home directory for the user.
- `-G` assigns the user to specified secondary groups (initial group).
- `guest_user` is created without extra group assignment (will belong to default user group).

Screenshot:

3. Set temporary passwords for all users

```
echo "dev_lead:password123" | sudo chpasswd
echo "dev_junior:password123" | sudo chpasswd
echo "tester_lead:password123" | sudo chpasswd
echo "tester_junior:password123" | sudo chpasswd
echo "guest_user:password123" | sudo chpasswd
```

Explanation:

- `chpasswd` updates the password for each user with a simple temporary password.

Screenshot:

```
┌──(denver㉿kali)-[~]
└─$ echo "dev_lead:password123" | sudo chpasswd

┌──(denver㉿kali)-[~]
└─$ echo "dev_junior:password123" | sudo chpasswd

┌──(denver㉿kali)-[~]
└─$ echo "tester_lead:password123" | sudo chpasswd

┌──(denver㉿kali)-[~]
└─$ echo "tester_junior:password123" | sudo chpasswd

┌──(denver㉿kali)-[~]
└─$ echo "guest_user:password123" | sudo chpasswd

┌──(denver㉿kali)-[~]
└─$
```

4. Verify the users and groups

```
getent passwd dev_lead
getent group development
getent group testing
getent passwd guest_user
```

Explanation:

- `getent` checks system databases for users and groups to confirm creation.
- `dev_lead` user exists with UID 1001, primary GID 1003, home `/home/dev_lead`, shell `/bin/sh`.
- Group `development` (GID 1001) includes members `dev_lead` and `dev_junior`.
- Group `testing` (GID 1002) includes members `tester_lead` and `tester_junior`.
- `guest_user` exists with UID 1005, GID 1007, home `/home/guest_user`, shell `/bin/sh`.

Screenshot:

```
┌──(denver㉿kali)-[~]
└─$ getent passwd dev_lead
dev_lead:x:1001:1003::/home/dev_lead:/bin/sh

┌──(denver㉿kali)-[~]
└─$ getent group development

development:x:1001:dev_lead,dev_junior

┌──(denver㉿kali)-[~]
└─$ getent group testing

testing:x:1002:tester_lead,tester_junior

┌──(denver㉿kali)-[~]
└─$ getent passwd guest_user
guest_user:x:1005:1007::/home/guest_user:/bin/sh

┌──(denver㉿kali)-[~]
└─$
```

Checkpoint Question

> **What group is assigned to new users by default if no group is specified?**

By default, each new user is assigned a private group with the same name as their username.

Step 2: Assign Users to Groups

1. Add `dev_lead` and `dev_junior` to the `development` group

```
sudo usermod -aG development dev_lead
sudo usermod -aG development dev_junior
```

Explanation:

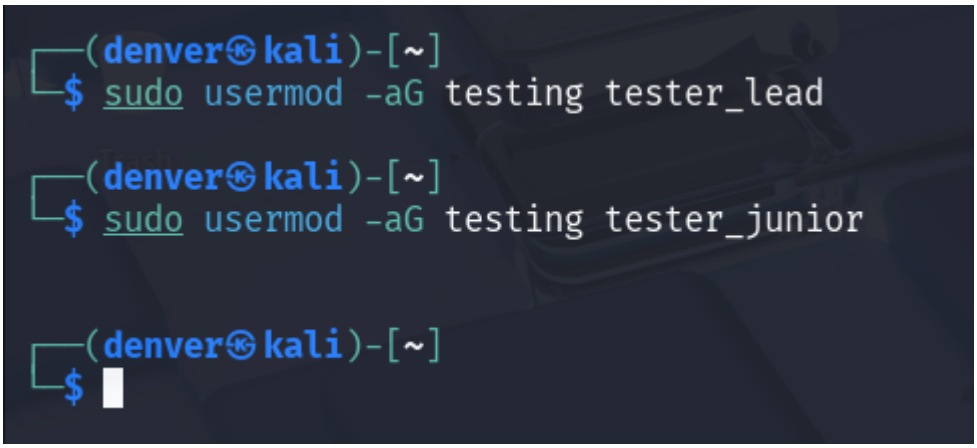
- `usermod -aG` adds groups to user without removing existing groups (`-a` means append).

Screenshot:

2. Add `tester_lead` and `tester_junior` to the `testing` group

```
sudo usermod -aG testing tester_lead
sudo usermod -aG testing tester_junior
```

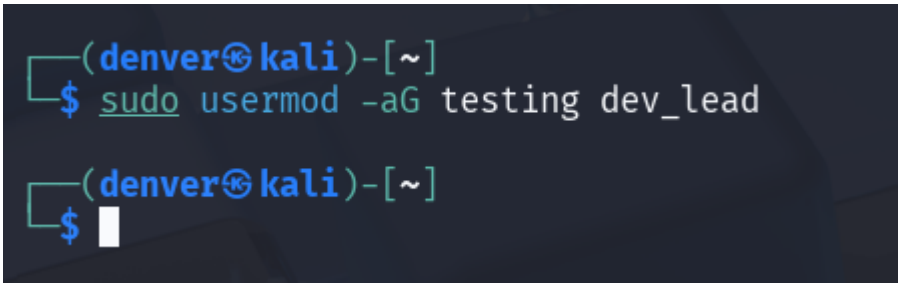
Screenshot:



3. Add `dev_lead` to both `development` and `testing` groups

```
sudo usermod -aG testing dev_lead
```

Screenshot:



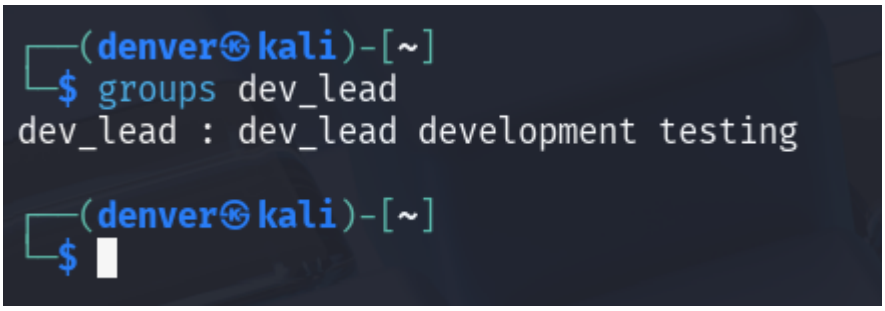
4. Verify `dev_lead`'s group memberships

```
groups dev_lead
```

Explanation:

- `groups` shows all groups the user belongs to.

Screenshot:



Checkpoint Question

> **After adding `dev_lead` to both groups, what groups does `dev_lead` belong to?**

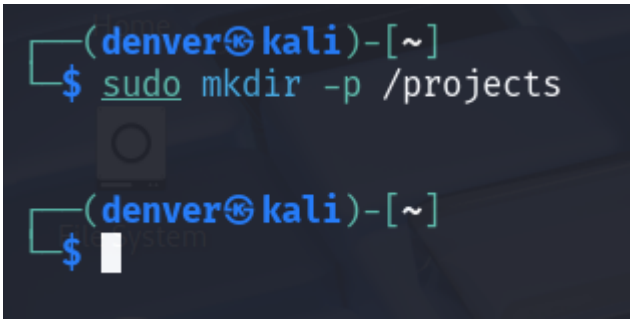
The user belongs to both `development` and `testing` groups.

Step 3: Set Up Project Directories

1. Create `/projects` directory if it does not exist

```
sudo mkdir -p /projects
```

Screenshot:



2. Create directories for Alpha and Beta teams

```
sudo mkdir /projects/alpha
sudo mkdir /projects/beta_reports
```

Screenshot:

3. Set group ownership of each directory

```
sudo chgrp development /projects/alpha
sudo chgrp testing /projects/beta_reports
```

Explanation:

- `chgrp` changes the group owner of a file or directory. **Screenshot:**

4. Verify group ownership

```
ls -ld /projects/alpha /projects/beta_reports
```

Explanation:

- `ls -ld` lists directory details, showing ownership and permissions.

Screenshot:

```
┌──(denver㉿kali)-[~]
└─$ ls -ld /projects/alpha /projects/beta_reports

drwxr-xr-x 2 root development 4096 May 30 06:26 /projects/alpha
drwxr-xr-x 2 root testing     4096 May 30 06:26 /projects/beta_reports

┌──(denver㉿kali)-[~]
└─$
```

Checkpoint Question

> **What is the default owner and group of a directory created by the root user?**

By default, both owner and group are `root`.

Step 4: Set Up Directory Permissions

1. Set permissions for `/projects/alpha`

```
sudo chmod 770 /projects/alpha
```

- Owner and group: full permissions (read/write/execute)
- Others: no access

Screenshot:

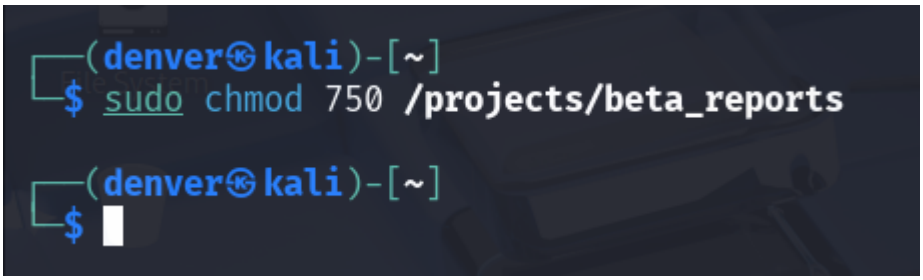
```
┌──(denver㉿kali)-[~]
└─$ sudo chmod 770 /projects/alpha

┌──(denver㉿kali)-[~]
└─$
```

2. Set permissions for `/projects/beta_reports`

```
sudo chmod 750 /projects/beta_reports
```

- Owner: full permissions (rwx)
- Group: read and execute (r-x)
- Others: no permissions

Screenshot:



Checkpoint Question

> **Translate these permissions to octal notation:**

- For `/projects/alpha` → 770 :
  - Owner: read (4) + write (2) + execute (1) = 7
  - Group: read (4) + write (2) + execute (1) = 7
  - Others: no permissions = 0
- For `/projects/beta_reports` → 750:
  - Owner: read (4) + write (2) + execute (1) = 7
  - Group: read (4) + execute (1) = 5
  - Others: no permissions = 0

Step 5: Test Access to Alpha Directory

1. Login as `dev_junior`

```
su - dev_junior
cd /projects/alpha
touch dev_file.txt
ls -l
```

Explanation:

- `su - dev_junior` fully switches shell session to `dev_junior`, with their environment, permissions, and access.
- `dev_junior` can navigate, create, and list files inside `/projects/alpha`.

Screenshot:

```
┌──(denver㉿kali)-[~]
└─$ su - dev_junior

Password:
$ whoami
dev_junior
$ pwd
/home/dev_junior
$ cd /projects/alpha
$ touch dev_file.txt
$ ls -l
total 0
-rw-rw-r-- 1 dev_junior dev_junior 0 May 30 06:40 dev_file.txt
$
```

2. Login as `guest_user`

```
su - guest_user
cd /projects/alpha
```

Explanation:

- `guest_user` cannot enter `/projects/alpha` (Permission denied).

Screenshot:

```
┌──(denver㉿kali)-[~]
└─$ su - guest_user
Password:
$ whoami
guest_user
$ pwd
/home/guest_user
$ cd /projects/alpha
-sh: 3: cd: can't cd to /projects/alpha
$
```

Checkpoint Question

> **Why can `dev_junior` write to the directory but `guest_user` is blocked?**

Because `dev_junior` is in the `development` group with write permission, whereas `guest_user` is neither owner nor group member, and others have no permissions.

Step 6: Test Access to Beta Reports Directory

1. Login as `tester_junior`

```
su - tester_junior
cd /projects/beta_reports
touch testfile.txt
```

Explanation:

- `tester_junior` can enter `/projects/beta_reports` and read files but cannot create new files because the group has read and execute permissions only (no write).

Screenshot:

2. As `root`, create a sample report file

```
sudo bash -c 'echo "Sample report content" >
/projects/beta_reports/sample_report.txt'
```

Screenshot:

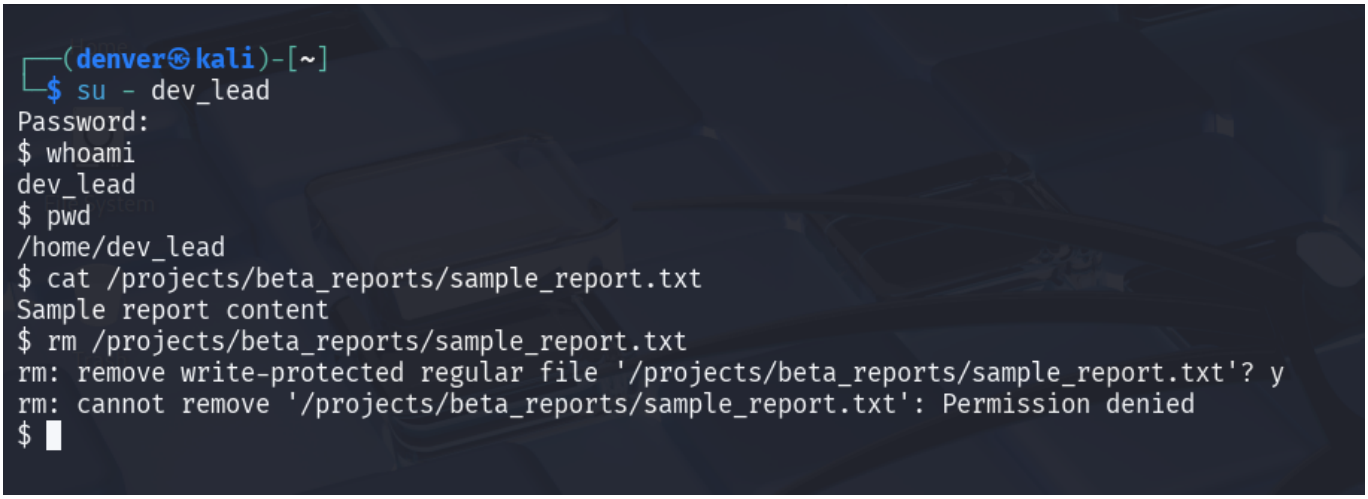
3. Login as `dev_lead`

```
su - dev_lead
cat /projects/beta_reports/sample_report.txt
rm /projects/beta_reports/sample_report.txt
```

Explanation:

- `dev_lead`, who belongs to both `development` and `testing` groups, can read files in `/projects/beta_reports` but cannot delete them due to lack of write permission on the directory.

Screenshot:



Checkpoint Questions

> **Was `tester_junior` able to write to the directory? Why or why not?**

No, because the group has read and execute, but no write permission.

> **Could `dev_lead` read or delete the file? Explain.**

Can read (member of testing group with read access) but cannot delete because deleting requires write permission on the directory.

Final Reflection

> **1. What do directory execute (`x`) permissions allow users to do?**

Allow users to enter (`cd`) the directory and access files and subdirectories inside it if they have the required read permissions.

> **2. What are the consequences of setting group permissions without configuring group ownership?**

If the group ownership is not set correctly, users in the intended group may not get the expected permissions, as permissions apply to the owner and group assigned.

> **3. What command would you use to safely assign a user to multiple groups without removing existing memberships?**

Use `sudo usermod -aG group1,group2 username`, where `-a` means append groups.

Extension Challenge

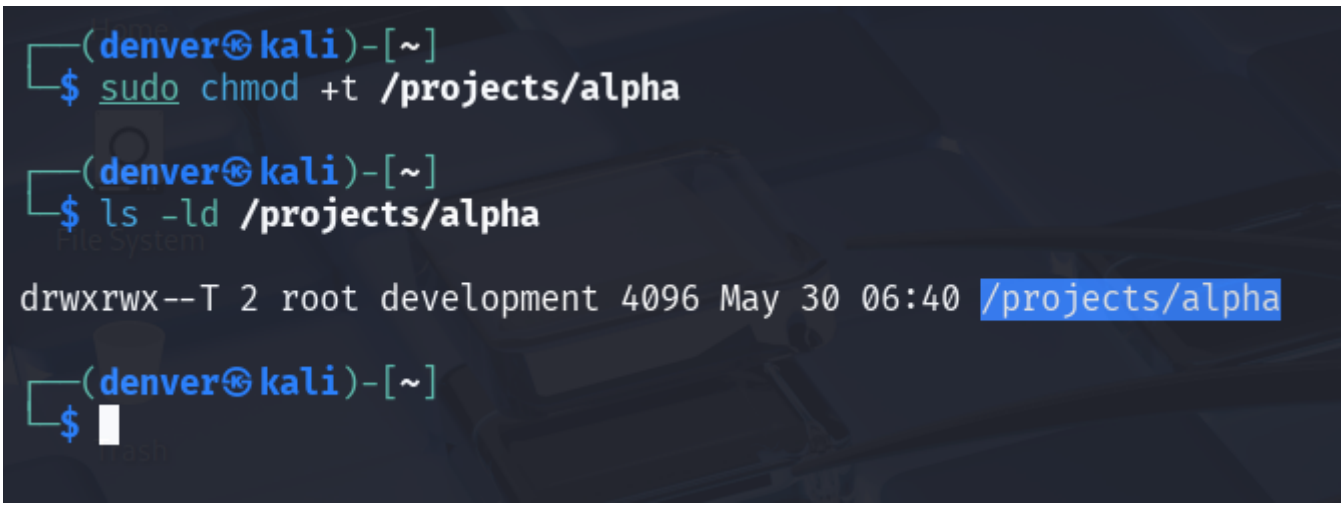
Prevent team members from deleting each other's files in `/projects/alpha`

```
sudo chmod +t /projects/alpha
```

Explanation:

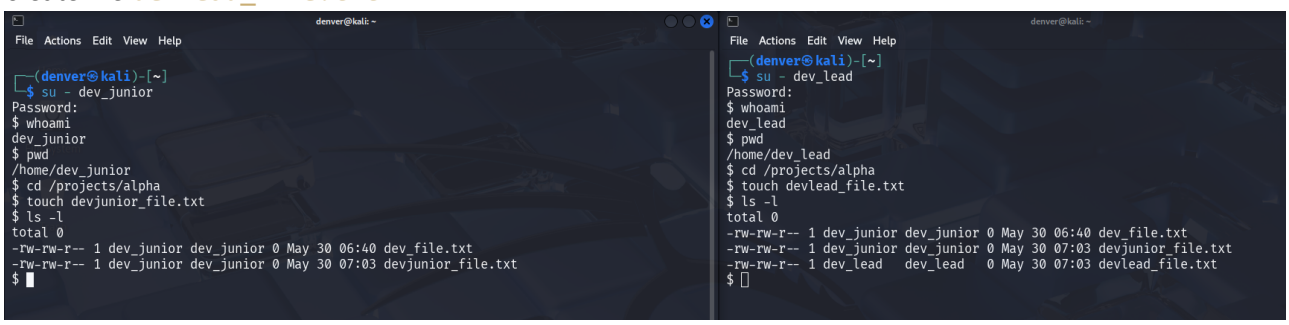
- The sticky bit (`t`) on a directory means:
  - Only the owner of a file inside the directory or the root user can delete or rename that file.
  - Other users, even if they have write permission on the directory, cannot delete or rename files owned by others.
- This prevents team members from accidentally or maliciously deleting each other's files.

Screenshot:



Example:

- Login as `dev_junior` and create file `devjunior_file.txt`. In other terminal, login as `dev_lead` and create file `devlead_file.txt`



- They can't delete each other's file but can delete their own files

```
┌──(denver㉿kali)-[~]
└─$ su - dev_junior
Password:
$ whoami
dev_junior
$ pwd
/home/dev_junior
$ cd /projects/alpha
$ touch devjunior_file.txt
$ ls -l
total 0
-rw-rw-r-- 1 dev_junior dev_junior 0 May 30 06:40 dev_file.txt
-rw-rw-r-- 1 dev_junior dev_junior 0 May 30 07:03 devjunior_file.txt
$ rm devlead_file.txt
rm: remove write-protected regular empty file 'devlead_file.txt'? y
rm: cannot remove 'devlead_file.txt': Operation not permitted
$ rm devjunior_file.txt
$ ls -l
total 0
-rw-rw-r-- 1 dev_junior dev_junior 0 May 30 06:40 dev_file.txt
-rw-rw-r-- 1 dev_lead   dev_lead   0 May 30 07:03 devlead_file.txt
$
```

```
┌──(denver㉿kali)-[~]
└─$ su - dev_lead
Password:
$ whoami
dev_lead
$ pwd
/home/dev_lead
$ cd /projects/alpha
$ touch devlead_file.txt
$ ls -l
total 0
-rw-rw-r-- 1 dev_junior dev_junior 0 May 30 06:40 dev_file.txt
-rw-rw-r-- 1 dev_junior dev_junior 0 May 30 07:03 devjunior_file.txt
-rw-rw-r-- 1 dev_lead   dev_lead   0 May 30 07:03 devlead_file.txt
$ rm devjunior_file.txt
rm: remove write-protected regular empty file 'devjunior_file.txt'? y
rm: cannot remove 'devjunior_file.txt': Operation not permitted
$ rm devlead_file.txt
$ ls -l
total 0
-rw-rw-r-- 1 dev_junior dev_junior 0 May 30 06:40 dev_file.txt
$
```