BUC3129 · TOPIC 3 · LINUX

Who is allowed to do what?

Users, groups and permissions on your own server.

BY THE END OF TODAY

Three things you will be able to do

Make an account

Create a user, give them a password, and put them in a group.

Give the right power

Let someone do one job — not everything.

Take access away

Handle a leaver properly, without losing their work.

All of it on your own server. Nothing here is theory you cannot try today.

ONE MINUTE, THEN WE START

Four rules, because some of you have real servers

Snapshot first	One click before every exercise. It turns a disaster into a two-minute reset.
Open two windows	Change things in one. Keep the other logged in. It is your way back.
Never type usermod -G	Without the -a it deletes every other group the person is in. We will look at this properly.
Disable, do not delete	Nobody deletes a user today. Not one person.

If a machine matters to somebody, practise on a copy of it. That is what snapshots are for.

DO IT NOW

3 MIN

Who does your server think you are?

Open a terminal. Type this one line. Then read it back to me.

```
id
```

You will see something like this:

```
uid=1001(amina) gid=1001(amina) groups=1001(amina),27(sudo)
```

Nobody has 0. If yours says 0, you are logged in as root — and that is a habit we will fix today.

THE ONE IDEA BEHIND EVERYTHING TODAY

Your name is decoration. The number is you.

UID

Your user number.
Yours is probably 1000 or 1001.

GID

Your group number.
Usually the same at first.

0

Root. Total power.
Not a person — a capability.

Files are owned by numbers, not names. Change your username and you keep your files. Change your number and you lose them.

That single fact explains most permission problems you will ever meet.

WHERE YOUR USERS ACTUALLY LIVE

Three files. That is the whole system.

/etc/passwd	Contains information about local accounts. Everyone's name, number, home folder and shell. Anyone can read it — there are no passwords in it.
/etc/shadow	The password hashes. Root only. This is the one that matters.
/etc/group	Which groups exist, and who is in them.

```
# have a look - this is safe, it only reads
tail -3 /etc/passwd
```

No database. No console. Three text files you can read with your own eyes.

DO IT NOW

8 MIN

Make your first user

Five lines. Use a real name — a friend, a classmate, anyone.

```
sudo useradd -m joseph      # -m gives them a home folder
sudo passwd joseph                    # set their password
id joseph                             # check it worked
ls -l /home                           # their folder is there
su - joseph                           # become them. exit to come back
```

What UID your new user was given. We will compare numbers.

Everybody's new user gets a number one higher than the last one created. Yours will not all match — and that is a problem we solve later today.

THE EASY IDEA PEOPLE OVERCOMPLICATE

A group is a keyring

You do not give fifty people a key each. You give the keyring to a group, and put people in it.

Without groups

Fifty people. Fifty separate permissions to set, and to remove later.

With groups

One group called finance. Set permission once. Add and remove people.

When they leave

Take them out of the group. Done. Nothing else to hunt for.

Rule for life: never give a permission to a person. Give it to a group, and put the person in the group.

DO IT NOW

8 MIN

Groups — and the famous mistake

```
sudo groupadd finance
sudo usermod -aG finance joseph          # -aG means ADD to the group
id joseph                                # you should see finance in the list
```

Now break it on purpose. Everybody together:

```
sudo usermod -G finance joseph          # no -a this time. Watch what happens.
id joseph                                # every other group is gone
```

One missing letter removed every other group that user had. On a real server, that letter takes away sudo.

Now fix it: sudo usermod -aG finance,sudo joseph

A TRUE STORY, IN EVERY ORGANISATION

Somebody left. Nobody told IT.

Six months later	Their account still works. Their key still opens the server. Nobody noticed.
So do not delete	Delete the account and their files lose their owner. Somebody will need those files.
Lock it instead	Locked, labelled with the date, and their key removed. Reversible, and honest.

```
sudo usermod -L joseph                          # lock the password
sudo usermod -c "Left 01 Sep, locked by NK" joseph
sudo mv /home/joseph/.ssh/authorized_keys /home/joseph/.ssh/keys.revoked
```

Locking the password does NOT stop an SSH key. Miss that third line and they can still get in tomorrow.

Linux Users, Groups & Permissions

Achieving Active Directory style file sharing and access control natively on Linux

Commands covered: `gpasswd` · `mkdir` · `chown` · `chmod` · `setfacl` · `getfacl`

Putting this into action

To put this into action, you will use the Linux terminal. Here is a step-by-step example of how to execute these commands, assuming:

your group	your main user	your outside user
`accounting`	`john`	`sarah`

1 Step 1: Open Your Terminal

You need administrative privileges to manage users and permissions. You will do this by adding sudo before your commands, which will prompt you for your admin password.

2 Step 2: Remove a Person from a Group

If john leaves the accounting department, you remove him from the accounting group instantly by running:

```
$ sudo gpasswd -d john accounting
```

What happens

John can no longer access any files restricted to the accounting group.

To check if it worked

Type groups john to see a list of his current groups.

3 Step 3: Create and Secure the Shared Folder

Let's create a shared folder at /home/shares/accounting_docs and set it up so only group members can access it.

Create the folder:

```
$ sudo mkdir -p /home/shares/accounting_docs
```

Change the group owner to accounting:

```
$ sudo chown -R root:accounting /home/shares/accounting_docs
```

Lock down permissions so only the root owner and the accounting group can read/write to it (others get zero access):

```
$ sudo chmod 770 /home/shares/accounting_docs
```

Apply the SGID rule so any new file created inside this folder automatically stays owned by the accounting group:

```
$ sudo chmod g+s /home/shares/accounting_docs
```

Read(r)-4

Write(w)-2

Execute(x)-1

7

Owner/root

Group

Others

750

147

4 Step 4: Let an Outsider In (Using ACLs)

If sarah is not in the accounting group, but she needs to see (Read-Only) or edit (Read/Write) the files inside that folder, you use an Access Control List (ACL).

GRANT To give Sarah Read and Write access:

```
$ sudo setfacl -m u:sarah:rw- /home/shares/accounting_docs
```

VERIFY To verify the permissions:

```
$ getfacl /home/shares/accounting_docs
```

(You will see a detailed output showing that the accounting group has full access, and user sarah has rw access).

Would you like to test these commands out on your system right now, or do you need help creating the users and groups first to test with?

Did I just build Active Directory on Linux?

Yes, you achieve the same results for file sharing and permissions, but you do it manually without a centralized domain.

Windows Active Directory (AD)

You use a graphical interface to create users, assign them to security groups, and manage folder permissions (NTFS).

Linux

The commands we used (useradd, gpasswd, chmod, and setfacl) accomplish the exact same security goals—restricting or granting access to specific people—but directly through the system's core settings.

Active Directory vs. Native Linux Permissions

Feature	Windows Active Directory	Native Linux (What you just did)
User & Group Control	AD Users and Computers (GUI)	`useradd, groupadd, gpasswd`
Main Group Access	Security Group Permissions	`Standard Group Permissions (chmod 770)`
Special Group Rules	Advanced Inherited Permissions	`The SGID bit (chmod g+s)`
Adding Outside Users	Adding specific users to the folder's ACL	`Linux Access Control Lists (setfacl)`

The Core Difference

What you just built

a localized permission system

It works perfectly for managing users and shared folders on that specific Linux server.

What Active Directory does extra

centralization

AD acts as a single phonebook for your entire network, allowing users to log into any computer or server on the network using the same username and password.

Want to make Linux act EXACTLY like an AD Domain?

If you want a Linux server to act as a true Windows Active Directory Domain Controller—where Windows computers can join it, and you can manage users with a central login—you can install a free tool called Samba.



Would you like to know how to install Samba to share this folder with Windows computers on your network, or do you want to practice managing more users with the local Linux commands first?

Class Lab: Linux User & Group Administration

Scenario: you are the System Administrator for TechCorp

The situation

- TechCorp has 9 employees in two departments: Engineering and HR.
- An external auditor needs temporary access to read Engineering reports.
- One employee quit today and must be removed immediately.

Phase 1: your task

- Create every user below with a default password.
- Create the two departments as groups.
- Add each employee to their department group.

Engineering

`group: engineering`

`alex` `blake` `charlie` `dana` `evan`

evan quit today. He must lose all team privileges.

HR

`group: hr`

`fiona` `george` `hannah` `ian`

External auditor

`no department`

`auditor_victor`

Phase 2: System Management Tasks

Complete each task in the Linux terminal. Write down or save every command you use.

1 Handle the terminated employee

1. Remove evan from the engineering group so he loses all team privileges.
2. Lock his account entirely so he cannot log into the system at all.

2 Create the Engineering workspace

1. Create the directory /shares/eng_workspace.
2. Change its group ownership to engineering.
3. Set permissions so only root and the engineering group have read, write and execute. Everyone else gets Access Denied.
4. Apply the SGID bit so new files inside automatically belong to engineering.

3 Grant external auditor access

1. Use an Access Control List (ACL) to give auditor_victor read only access to /shares/eng_workspace.
2. Do not add him to the engineering group.

4 Verification

1. Check alex's groups to confirm he is properly assigned.
2. Check evan's groups to confirm he was removed.
3. Print the detailed ACL for /shares/eng_workspace to prove the auditor has access and the SGID bit is active.