

Users, Groups & Sudo (Core)

Linux Commands Course · Section 9

Goal

Learn how to **inspect and manage users and groups**, and understand **sudo** – the gateway to administrative privileges.

This section is foundational for system administration and security.

Users and Groups in Linux

Every user on Linux has:

- A **username** (like `student` or `root`)
- A **UID** (user ID number)
- A **primary group**
- Optional secondary groups
- A **home directory** and **default shell**

Groups organize users for shared permissions and access control.

Inspecting User Information – `id`

Show your user and group identity:

```
id
```

Example output:

```
uid=1000(student) gid=1000(student) groups=1000(student),27(sudo)
```

- `uid` → your user ID
 - `gid` → your main group
 - `groups` → all groups you belong to
-

Who Am I? – whoami

Prints your current effective username:

```
whoami
```

Useful in scripts or when using `sudo` to confirm who you are.

Listing Group Memberships – groups

Show which groups you belong to:

```
groups
```

Example:

```
student : student sudo docker
```

Active Users – who and w

`who` shows users currently logged in:

```
who
```

`w` gives more detail – what each user is doing:

```
w
```

Example:

```
student pts/0 2025-10-22 10:31 bash
```

Login History – last

Displays recent logins and reboots.

```
last
```

Output example:

```
student pts/0 192.168.1.15 Wed Oct 22 10:00 still logged in
reboot  system boot Wed Oct 22 09:55
```

This information is stored in `/var/log/wtmp`.

Understanding `sudo`

`sudo` lets authorized users run commands as another user – typically `root`.

Example:

```
sudo apt update
```

You'll be prompted for your **own password**, not `root`'s.

Why use `sudo` instead of logging in as `root`?

- Safer (tracks every action)
 - Temporarily elevates privileges
 - Logs activity to `/var/log/auth.log`
-

How `sudo` Works

`sudo` checks its configuration file `/etc/sudoers` to see who can run what.

You can view effective privileges with:

```
sudo -l
```

If allowed, your command runs as if root executed it.

Example:

```
sudo whoami  
# Output: root
```

Editing sudo Rules – visudo

You **must** use `visudo` to safely edit sudo privileges.

```
sudo visudo
```

Why?

- `visudo` checks syntax before saving, preventing broken access.
- Editing `/etc/sudoers` manually can lock out admin access!

Example rule in the file:

```
alice ALL=(ALL:ALL) ALL
```

Meaning:

- **alice** → username
- **ALL** → any host
- **(ALL:ALL)** → can act as any user and group
- **ALL** → may run any command

You can restrict to specific commands:

```
bob ALL=(ALL) /usr/bin/systemctl restart nginx
```

Now Bob can only restart nginx, not anything else.

Granting Group Access via `sudo`

Instead of editing user-by-user, use groups.

Example line in `/etc/sudoers`:

```
%sudo    ALL=(ALL:ALL) ALL
```

Meaning: anyone in the `sudo` group has full admin rights.

Add user to that group:

```
sudo usermod -aG sudo alice
```

On RHEL/Fedora, the equivalent group is `wheel`.

Security Tips for Sudo

- Never edit `/etc/sudoers` directly – always use `visudo`.
 - Limit commands users can run if full access isn't needed.
 - Use `sudo -l` to verify your privileges.
 - Avoid `sudo su` (defeats auditing and accountability).
-

Creating a New User – useradd

Add a new user to the system (requires root).

```
sudo useradd -m alice
```

Options:

- `-m` → create home directory
- `-s /bin/bash` → set shell
- `-G` → add to extra groups

Example:

```
sudo useradd -m -s /bin/bash -G sudo alice
```

Set password:

```
sudo passwd alice
```

Modifying a User – usermod

Change user settings such as shell, group, or name.

Add a user to a group:

```
sudo usermod -aG docker alice
```

Change shell:

```
sudo usermod -s /bin/zsh alice
```

Rename user:

```
sudo usermod -l newname oldname
```

Deleting a User – userdel

Remove a user account.

```
sudo userdel alice
```

Delete the user's home directory as well:

```
sudo userdel -r alice
```

Always ensure data is backed up before deletion.

Changing Passwords – `passwd`

Change your password:

```
passwd
```

Change another user's password (admin only):

```
sudo passwd bob
```

Force password change on next login:

```
sudo passwd -e alice
```

Password Aging and Expiry – chage

View password aging info:

```
sudo chage -l alice
```

Set password expiration (e.g., 90 days):

```
sudo chage -M 90 alice
```

Set account expiry date:

```
sudo chage -E 2025-12-31 alice
```

This helps enforce password rotation and account control.

Managing Groups – groupadd

Create a new group:

```
sudo groupadd developers
```

Add an existing user to it:

```
sudo usermod -aG developers alice
```

Group Passwords and Administration – gpasswd

`gpasswd` manages group membership and optional passwords.

Add or remove users from a group:

```
sudo gpasswd -a bob developers  
sudo gpasswd -d bob developers
```

Set a group administrator (can add/remove users):

```
sudo gpasswd -A alice developers
```

Set a group password (rarely used today):

```
sudo gpasswd developers
```

Recap

- Inspect users: `id`, `whoami`, `groups`, `who`, `w`, `last`
- Manage users: `useradd`, `usermod`, `userdel`, `passwd`, `chage`
- Manage groups: `groupadd`, `gpasswd`
- Privilege control: `sudo`, `visudo`

`sudo` is the bridge between normal users and root privileges – use it carefully.

Practice

1. Create a new user `labuser` with a home directory and bash shell.
 2. Set a password and add them to the `sudo` group.
 3. Create a group called `developers` and add `labuser` to it.
 4. Check the user's groups with `id`.
 5. Edit sudo rules with `visudo` to allow only `/usr/bin/apt` commands.
 6. Log in as `labuser` and test `sudo whoami`.
-

Next Up

Processes, Services & Logs (Core) – managing programs, monitoring, and troubleshooting.