Absolutely! Let’s break it down into **simple steps** and **plain English** so you understand what you just did with systemctl and systemd.

**🔧 What is systemd and systemctl?**

* systemd is the **system manager** in Ubuntu Linux.
* systemctl is the **command** you use to **control services** (like starting, stopping, checking status, etc.).

**✅ What You Learned – Step by Step**

**1. View All Running Services**

* Run: systemctl status
* Press **space** to scroll, **Q** to quit.

**2. Check a Service (like cron)**

* Run: systemctl status cron → shows you if it's running.
* Run: sudo !! → runs the same command but with **sudo**, so you see **extra log info**.

**3. Restart a Service**

* sudo systemctl restart cron → restarts cron.
* Useful if changes were made or if it's stuck.

**4. Stop and Disable a Service**

* sudo systemctl stop cron → stops the cron service **right now**.
* sudo systemctl disable cron → stops it from starting **at boot**.

**5. Enable and Start a Service**

* sudo systemctl enable --now cron → enables it at boot **and** starts it now.
* sudo systemctl disable --now cron → disables it **and** stops it now.

**6. Mask and Unmask a Service**

* sudo systemctl mask --now cron → makes the service **impossible** to start until unmasked.
* sudo systemctl unmask cron → allows it to be started again.
* Good for **blocking a service temporarily**.

**7. View a Service’s Configuration**

* systemctl cat cron → shows the service file.
* It’s usually in /lib/systemd/system/.

**8. Change Service Description (Override File)**

* sudo systemctl edit cron → makes a small **custom override** file.
* Add:
* [Unit]
* Description=Schedule regular task
* Save with **Ctrl+S**, exit with **Ctrl+X**.
* Reload: sudo systemctl daemon-reload
* Check: sudo systemctl status cron

**9. Remove Custom Config**

* sudo rm -rf /etc/systemd/system/cron.service.d/ → deletes your override file.

**10. Full Edit (Copy Full File)**

* sudo systemctl edit cron --full → makes a **full copy** of the original config file to edit.
* After editing, reload again: sudo systemctl daemon-reload.

**💡 Summary**

* systemctl status <service> → see if it's running.
* start, stop, restart → control when it runs.
* enable, disable → control whether it runs on **boot**.
* mask / unmask → **block or unblock** it completely.
* edit → change how the service behaves (either partially or fully).
* Always use sudo when changing or editing services.

Would you like a visual cheat sheet or flowchart of this too?

Great! Let’s now break this second part — **Managing systemd Targets** — into **simple steps** and plain explanations, just like before.

**🧠 What Are "Targets" in systemd?**

* **Targets = Groups of services** that systemd starts together.
* They replaced **old "runlevels"** in older Linux systems.
* Think of targets as **checkpoints** during boot, each loading certain services.
* Examples:
  + basic.target → low-level setup
  + multi-user.target → networked, multi-user text environment
  + graphical.target → same as multi-user, but with a GUI (desktop)

**✅ What You Learned — Step by Step**

**1. Check Boot Time**

* systemd-analyze → shows how long your system took to boot.

**2. Find Slow Services**

* systemd-analyze blame → shows which services took the most time to start.
* **Top = slowest service**.

**3. List All Loaded Services**

* systemctl list-units --type=service → see active services.
* systemctl list-units --type=target → see active targets (like runlevels).

**4. See What Target Cron Belongs To**

* systemctl cat cron | grep WantedBy
  + This shows which **target** cron wants to be part of (usually multi-user.target).

**5. View Target Directory**

* tree /etc/systemd/system/multi-user.target.wants/
  + This lists services linked to the multi-user.target.
  + You’ll see a file like cron.service here if it’s enabled.

**6. Disable/Enable a Service and See the Link Disappear/Reappear**

* sudo systemctl disable cron → removes the symlink from the target folder.
* tree /etc/systemd/system/multi-user.target.wants/ → you’ll notice cron.service is gone.
* sudo systemctl enable cron → adds it back.

**7. Default Boot Target**

* systemctl get-default → shows current **default target**.
  + Often it's graphical.target on desktops.
* sudo systemctl set-default multi-user.target
  + Changes the **default boot mode** to text-based (no GUI).
  + **Note:** It doesn't change the current session — only applies at next boot.

**8. View Load Order of Targets**

Each target loads **after** another:

* systemctl cat graphical.target → loads **after multi-user.target**
* systemctl cat multi-user.target → loads **after basic.target**
* systemctl cat basic.target → loads \*\*after sysinit.target` (which you could explore later)

**📦 Summary (In Simple Words)**

| **Command** | **What It Does** |
| --- | --- |
| systemd-analyze | Shows boot time |
| systemd-analyze blame | Shows what slowed down boot |
| systemctl list-units --type=service | Shows all running services |
| systemctl list-units --type=target | Shows system targets (like boot levels) |
| systemctl cat cron | Shows cron's config |
| grep WantedBy | Shows which target cron belongs to |
| tree /etc/systemd/system/multi-user.target.wants/ | Shows enabled services for that target |
| systemctl disable/enable cron | Remove/add service from boot target |
| systemctl get-default | Shows current default boot target |
| systemctl set-default TARGET | Sets what target to boot into by default |
| systemctl cat TARGET | Shows what each target depends on and loads after |

**🧩 What Did You Learn?**

* Targets = boot stages (replacing runlevels)
* Services are linked to targets.
* You can **view, enable, disable, or reassign** them.
* You learned the **boot process** structure: basic → multi-user → graphical.

Absolutely! Here's a **simple explanation** of what Andrew Mallet is saying in that introduction:

👋 **Welcome Message (Who is Speaking?)**

* Hi! I'm **Andrew Mallet**.
* I run a company in the UK called **The Urban Penguin**.
* I **teach and consult** on Linux — helping people learn how to use and manage it.
* I've been working with Linux for **many years** as a **system administrator** (manages systems) and a bit as a **developer** (writes code, especially in **Python**).
* I also work with tools like **Ansible**, **Puppet**, and **Salt**, which help manage lots of computers automatically.

💻 **What Is This Course About?**

* It's called **"Ubuntu Linux Administration: User and Group Management."**
* It's for learning how to **manage users and groups** on **Ubuntu Linux**.
* Ubuntu is a very **popular version of Linux**, used at **home, work**, and even in **certification exams** like LFCS.

📘 **What Will You Learn?**  
In this course, you will learn how to:

1. ✅ **Create and manage users** (like setting up new accounts).
2. 👥 **Create and manage groups** (grouping users together for easier control).
3. 🔒 **Manage passwords and login security**.
4. 🧾 **Set up login scripts** (things that run automatically when a user logs in).
5. 🛡️ Use **PAM (Pluggable Authentication Modules)** — a way to control how users log in and authenticate.

🧠 **What Should You Know Already?**

* You should already be **comfortable using the Linux command line**.
* This is **not** for total beginners — it builds on basic Linux knowledge.

🎯 **By the End of the Course:**

* You’ll be able to **fully manage users and groups** on an Ubuntu Linux system.
* This is a **key skill for system admins**, especially in jobs or certification exams.

Would you like a simple list of commands related to user and group management to get started?

Absolutely! Here's a **simple breakdown** of what Andrew Mallet is saying in that video, using easy-to-understand language:

**👋 Welcome to the Course!**

Hi! This course is about **Ubuntu Linux Administration**, and the focus is on **managing users and groups**. Your instructor is **Andrew Mallet** — you can find him online as **@theurbanpenguin**.

**🧑‍💻 What Will You Learn?**

In this course, you'll learn things like:

* How to **create user accounts**
* How to **set and manage passwords**
* How to work with **system login scripts**
* How to use something called **PAM (Pluggable Authentication Modules)** — this controls how users log in
* How to use tools like sudo, su, and pkexec to **run commands as the root (admin) user**

**🔐 Why Is This Important?**

If you ever manage a Linux system (like a server), you'll need to:

* Add and manage users
* Control who can do what
* Understand how the system keeps track of user information

**📂 Where Are User Accounts Stored?**

User info is stored in the file:

/etc/passwd

Despite the name, this file **doesn't store passwords** anymore. It stores things like:

* Username
* User ID (UID)
* Home directory
* Shell (e.g., bash)

To see this info, you can use commands like:

cat /etc/passwd

or

getent passwd

**🧑 Types of Users**

Linux has 3 main types of users:

1. **Regular users** – People like you and me.
2. **System users** – Used by services (like a web server).
3. **Root user** – The superuser. Has all the power. This user has UID 0.

**💡 Extra Tips**

* **Usernames and passwords are case-sensitive** in Linux (so John is different from john).
* You can learn more about how files like /etc/passwd work by using:

man 5 passwd

**🧪 Practice Is Key!**

Don't just watch the videos — you should **set up a practice lab** using tools like **VirtualBox** and **Vagrant**, so you can try things out for yourself.

The exam for the **Linux Foundation Certified System Administrator (LFCS)** is hands-on, not multiple choice. You need to actually manage a Linux system.

**✅ Summary**

This module is helping you:

* Understand the different types of Linux users
* Learn where user info is stored
* Learn how to use tools to run admin tasks
* Prepare for hands-on work by setting up your own lab

Let me know if you'd like me to explain any **specific part** (like sudo, /etc/passwd, or system users) in even more detail!

Absolutely! Let's go through that whole explanation in **very simple words**, step by step:

**🎉 Welcome to Your First Linux Lab**

In this lab, you're going to **practice real commands** on a Linux system — specifically Ubuntu 20.04 — to explore how **users** are stored and managed.

**📂 Where Are User Accounts Stored?**

Linux keeps user account info in a file called:

/etc/passwd

It's pronounced "pass-wd", not "password".

You can see the contents by running:

cat /etc/passwd

This file has information about **all users** on the system (not their passwords though!).

**🔍 What's Inside /etc/passwd?**

Each line in this file is one user account.

The info is separated by **colons (:)** and includes 7 fields:

1. **Username** (e.g., vagrant)
2. **x** (placeholder for password – real password is stored elsewhere)
3. **User ID (UID)** – a number that identifies the user (e.g., 1000)
4. **Group ID (GID)**
5. **Comment** – often the full name of the user
6. **Home directory** (e.g., /home/vagrant)
7. **Shell** (e.g., /bin/bash)

To learn more about the file, you can run:

man 5 passwd

(Use q to quit the manual.)

**📚 What's getent?**

The getent command can also show user account info:

getent passwd

This is better than cat /etc/passwd because it works even when the users come from **external sources** like LDAP or Active Directory (used in bigger company networks).

You can use it for a specific user like this:

getent passwd vagrant

**🔧 What’s nsswitch.conf?**

This file:

/etc/nsswitch.conf

tells Linux **where to look for users** — in:

* Local files like /etc/passwd
* Or in network directories like LDAP

You can check it by running:

grep '^passwd' /etc/nsswitch.conf

**🧠 What Is AWK?**

AWK is a **text filtering tool**. It can break up each line into parts and work with them.

In /etc/passwd, the **colon (:)** separates each field. AWK can help us **filter users** based on their **User ID** (UID).

**✅ Standard Users vs System Users**

* **Standard users** have **UIDs 1000 or higher** (e.g., your login account).
* **System users** have **UIDs below 1000** (used by services like mail, web server).
* **Root user** has **UID 0**

**🔍 Using AWK to Filter Users**

**Example 1: Show all UIDs**

awk -F: '{ print $3 }' /etc/passwd

* -F: tells AWK that the colon : separates the fields.
* $3 means "print the 3rd field" (which is the UID).

**Example 2: Show only standard users (UID ≥ 1000)**

awk -F: '{ if ($3 >= 1000) print }' /etc/passwd

**Example 3: Show only system users (UID < 1000, but not root)**

awk -F: '{ if ($3 > 0 && $3 < 1000) print }' /etc/passwd

* && means AND
* So this says: show users with UID **greater than 0** (not root) **and less than 1000**

**🧪 Why This Matters**

Using tools like grep, awk, and getent, you can:

* Search for users
* See how users are stored
* Understand the system better
* Be ready for real-world Linux administration and certifications

Would you like a **cheat sheet** of these commands or an explanation of how to try this in a test environment (lab)?

Sure! Here's a **simple and clear explanation** of what was just said in the video:

**🧑‍💼 Who's the Boss in Linux?**

* In Linux, the **main boss** (the top admin) is a special user called **root**.
* The root user has **User ID (UID) 0**, which means it can do **anything** on the system.

**🔐 Logging in as Root (or not)**

* On **Ubuntu** (and other Debian systems), the **root account is locked by default** — it has **no password**.
* That means you **can’t log in as root directly** unless you set a password.

**🚀 How to "Become" Root (Privileged Access)**

You don’t need to log in as root directly. Instead, Linux gives you **tools to act like root temporarily**, which is safer.

Here are the 3 main tools:

| **Command** | **What it does** | **Do you need root password?** |
| --- | --- | --- |
| su | Switches to another user (default: root) | ✅ Yes, you need it |
| sudo | Runs one command as root | ❌ No (uses your own password) |
| pkexec | Similar to sudo, for GUI or CLI | ❌ No (uses your own password) |

**📌 The su Command (Substitute User)**

* su lets you **switch to another user**.
* By default, su switches you to **root**.
* You'll be asked for the **password of the user you're switching to**.

Example:

su

You’ll need to type the root password.

**🌍 What's the Difference: su vs su -?**

| **Command** | **What it does** |
| --- | --- |
| su | Switches user, but **keeps old environment** |
| su - | Switches user and **loads full environment** (like a full login) |

Think of su - as a **clean switch** where everything is set up like you're that user.

**🧪 Check Who You Are**

You can always check who you're logged in as:

id # shows user ID and groups

echo $USER # shows username

Sometimes, after using su, $USER still shows your **old username** — unless you use su -.

**🔒 Setting a Root Password (not always recommended)**

If you really want to use su, you can set a root password like this:

sudo passwd

Then enter a new root password.

⚠️ **Warning**: This makes your system **less secure**, because anyone with the root password can fully control your system.

**🧑‍🔧 Testing su with Another User**

Once you're root, you can switch to another user (like bob) **without a password**:

su - bob

Why? Because root is the boss — root doesn’t need permission from anyone else.

**🧹 Don’t Forget to Exit!**

Each time you use su, you create a new "layer" or **shell**.

Use:

exit

or press **Ctrl + D** to go back to your previous user.

If you forget to exit, you'll still be acting as another user (maybe even root), which can be dangerous.

**✅ Summary**

* Root (UID 0) is the **admin boss**.
* On Ubuntu, root login is **disabled** by default.
* You can **become root** with su, sudo, or pkexec.
* su needs a password (unless you're already root).
* su - gives a **clean login** with that user’s full environment.
* Be **careful** when acting as root — and always **exit** when you're done.

Let me know if you'd like me to show you a **step-by-step example** of these commands in a practice setup!

**🔑 What Is sudo?**

* sudo stands for **"superuser do"**.
* It lets a regular user **run admin (root) commands** without logging in as the root user.
* **You don’t need to know the root password** — just your **own** password.

**🆚 sudo vs su**

| **Feature** | **su** | **sudo** |
| --- | --- | --- |
| Uses root password? | ✅ Yes | ❌ No (uses your password) |
| Full root access? | ✅ Yes | ✅ Or limited (you choose) |
| Can control what users can do? | ❌ No | ✅ Yes — choose specific commands |

**✅ Why Use sudo?**

* **Safer** than sharing the root password.
* You can **choose which users can run which commands**.
* You can also **limit what commands** they’re allowed to use — very helpful for security!
* **🛠️ Using Polkit and pkexec to Recover from Broken sudo**
* **🚨 Problem:**
* If you accidentally misconfigure /etc/sudoers, **sudo will break for all users**, and you won’t be able to run admin commands — even as yourself.
* **Fixing it** can be difficult — especially if root login is disabled (which it is by default on Ubuntu and other Debian-based systems).
* **✅ Solution: Use pkexec (PolicyKit)**
* **Polkit** is a Linux component that manages privileges. It works well in desktop environments, but you can use it from the **command line** too, using:
* **pkexec → Runs a command as root**
* **pkttyagent → Waits to handle authentication (in a second terminal)**

Absolutely! Here's a **simple summary** of everything you just read — explained in clear, beginner-friendly language:

**🎉 Great Job! Let’s Recap What You’ve Learned So Far**

You've covered **a lot** about users and how to manage **admin access** on a Linux system.

**👥 1. Understanding Users**

* Every user on a Linux system is listed in the file:
* /etc/passwd
* You can see this list using:
* cat /etc/passwd
* Another way is using:
* getent passwd

This checks users from multiple sources — not just the local file. It can include users from Active Directory or LDAP if those are set up.

* The file /etc/nsswitch.conf controls where Linux looks for users (local files, LDAP, etc.)
* You learned how to use the **awk** command to filter and sort users:
  + Users with **ID 1000 or more** = normal users
  + Users with **ID below 1000** = system or service users

**🔐 2. Elevating Privileges (Becoming Admin)**

**🔄 su – Switch User**

* su username = switch to another user (you need *that* user's password)
* su - or su -l = same as above, but loads the full environment (like you're fully logging in)

**⚡ sudo – Run Commands As Admin**

* sudo is more flexible and **safer** than su
* You only need **your own password**, not the root user’s
* You can allow users to run only specific commands (e.g., only reset passwords)
* You control this using the **sudoers** file

**✍️ visudo – Safely Edit sudo Rules**

* Always use visudo to edit sudo settings, not a regular editor
* Why? Because it **checks for errors**
* If you break the file, sudo stops working 😬
* Use:
* sudo visudo -f /etc/sudoers.d/filename

This keeps your custom rules separate and safe from system updates

**🛟 3. If You Break sudo, Use pkexec to Fix It**

**🔑 What is pkexec?**

* A tool from **PolicyKit (Polkit)** that lets you run admin commands securely
* It's especially useful if you mess up sudo and get locked out

**🪟 How to Use It (Requires 2 Terminal Windows)**

1. In **window 1**, find your shell's process ID:
2. echo $$
3. In **window 2**, run:
4. pkttyagent --process [PID from step 1]

This waits for an authentication request.

1. Back in **window 1**, run:
2. pkexec visudo

Now you'll be prompted (in window 2) to enter your password and confirm the action.

1. This opens the visudo editor with admin access — so you can fix what you broke 👍

**🧠 Final Thoughts**

Here’s what you learned:

| **Topic** | **What You Learned** |
| --- | --- |
| /etc/passwd | Stores user info |
| awk, getent, id | Useful commands to view/filter users |
| su | Switch users (need their password) |
| sudo | Run admin commands (need *your* password) |
| visudo | Safely edit sudo rules with syntax checks |
| pkexec + pkttyagent | Recover from broken sudo configuration |

**🎓 Pro Tip:**

Practice all of this! Break things (on purpose in a VM), and fix them again. The more hands-on you get, the more confident you'll feel — even in stressful situations or exams.

Whenever you're ready, the next module will be about **creating and managing user accounts**. Want a preview or cheat sheet for that too?

Here’s a **simple and clear explanation** of what you just read — broken down so it’s easy to understand:

**👋 Welcome to the User Management Module!**

You’re going to learn how to:

1. ✅ **Create users**
2. ✏️ **Edit user accounts**
3. ❌ **Delete users**
4. 📁 **Work with home directories**
5. 👥 **Manage user groups**

Let’s go step by step. 🚶‍♂️

**1. 🔨 Creating Users with useradd**

The most common way to create a user is with:

sudo useradd username

But just typing that will:

* **Not** create a home directory (on Ubuntu)
* Use default settings (shell, groups, etc.)

**2. 🔍 Where Are User Accounts Stored?**

Local users are listed in:

/etc/passwd

If your system connects to things like Active Directory or LDAP, use:

getent passwd username

It looks both locally *and* in network-based directories.

**3. ⚙️ Default User Settings**

Defaults used by useradd come from two files:

* /etc/default/useradd
* /etc/login.defs

You can change the default shell, home dir location, etc. by editing these.

**4. 🏠 Home Directories**

On Ubuntu:

sudo useradd user1

...will not create a home directory.

To create it automatically:

sudo useradd -m user2

The files inside the new home directory are copied from a template folder called the **skeleton directory**:

/etc/skel

You can add files to /etc/skel, and they’ll be included in every new user’s home directory.

**5. 📝 Setting User Shell and Groups**

You can customize even more:

sudo useradd -m -s /bin/bash -G sudo user2

* -m → create home directory
* -s → set shell
* -G → add to extra groups (like sudo for admin access)

**6. 🧾 Viewing User Info**

* View a user's ID, primary group, and extra groups:
* id user1
* See their info in the passwd file:
* cat /etc/passwd | grep user1
* Learn the file format:
* man 5 passwd

**7. 🔄 Modifying Users**

Change user settings with usermod. Example:

sudo usermod -s /bin/zsh user1

Changes the login shell to Zsh.

**8. 🗑️ Deleting Users**

To delete a user:

sudo userdel user1

To delete the user **and their home directory**:

sudo userdel -r user1

However, this won’t delete files they created in other folders (like shared folders).

You can find leftover files by running:

sudo find / -user user1

**9. 👥 What’s the GECOS Field?**

When creating users, there’s a “comment” field that can hold personal info (like full name, phone, etc). This is called the **GECOS field** (from an old operating system called GE COS).

**10. 🧑‍💻 The adduser Script**

There’s also a **friendly script** for creating users:

sudo adduser username

It asks you questions (like password, full name, etc.) and sets up everything easily.

**11. 👪 Managing Groups**

You can organize users into groups to manage permissions more easily.

Useful commands:

* Create a group:
* sudo groupadd mygroup
* Add user to a group:
* sudo usermod -aG mygroup username
* Remove a user from a group (indirectly):  
  You can manually edit /etc/group or use tools like gpasswd.

**🧠 Summary**

You learned how to:

| **Task** | **Command** |
| --- | --- |
| Create a user | sudo useradd -m user1 |
| View users | cat /etc/passwd or getent passwd |
| Customize defaults | Edit /etc/login.defs |
| Set user shell/group | -s /bin/bash -G group |
| Modify a user | sudo usermod |
| Delete a user | sudo userdel -r user1 |
| Friendly user creation | sudo adduser |
| Manage groups | groupadd, usermod -aG |

**🚀 What’s Next?**

You’ll now get hands-on practice with all this. Try creating users, editing them, deleting them, and organizing them into groups.

Let me know if you’d like me to create a cheat sheet or mini lab for you to practice this on your own!

Absolutely! Let’s dive deeper into the **/etc/skel** directory and how it works in Linux user management.

**What is /etc/skel exactly?**

* The name **skel** stands for **skeleton** — it’s like a blueprint or template.
* It’s a directory located at /etc/skel on your Linux system.
* When you create a **new user** (with a home directory), the system **copies all the files and folders inside /etc/skel into the new user’s home directory**.
* This means **every new user starts off with the same initial files and folders** set by the administrator.

**Why is /etc/skel important?**

* When users first log in, you want them to have some default environment settings and useful files.
* Instead of creating those files manually for every new user, Linux automates this by copying from /etc/skel.
* This ensures consistency and saves time.
* Example: setting up default shell environment files, like .bashrc or .profile, that customize the command prompt and environment for users.

**What does /etc/skel usually contain?**

* Mostly **hidden files** — files that start with a dot (.), called **dotfiles**.
* These dotfiles include:
  + .bashrc — Configurations for the bash shell (aliases, prompt style, environment variables).
  + .profile — Environment variables and startup programs.
  + .bash\_logout — Commands run when the user logs out.
* These files help customize how the user’s shell behaves.

**How does it work when you add content to /etc/skel?**

1. **You put files or directories inside /etc/skel** — for example, a custom .vimrc to configure the Vim editor.
2. When you create a **new user account** with useradd or adduser, Linux **copies everything from /etc/skel into the user’s new home directory**.
3. So the new user starts with those files already set up.

**Important details about copying:**

* The copying happens **only once, at the moment the user’s home directory is created**.
* **If you add or change files inside /etc/skel after users are created, those changes do NOT affect existing users’ home directories.**
* This means /etc/skel is a **template for new users only**.

**How to see what's inside /etc/skel?**

* Since it mainly has hidden files, just running ls /etc/skel won’t show them.
* Use the command to **list all files including hidden ones**:

ls -la /etc/skel

* Output typically looks like:

-rw-r--r-- 1 root root 3771 Apr 12 2023 .bashrc

-rw-r--r-- 1 root root 1481 Apr 12 2023 .profile

-rw-r--r-- 1 root root 675 Apr 12 2023 .bash\_logout

**Adding extra files or symbolic links to /etc/skel**

* You can put any file or directory you want inside /etc/skel.
* For example, if you want all new users to have easy access to system documentation, you could create a **symbolic link** inside /etc/skel pointing to /usr/share/doc:

sudo ln -s /usr/share/doc /etc/skel/doc

* This creates a symbolic link named doc inside /etc/skel.
* Now every new user will get this symbolic link in their home directory, pointing to /usr/share/doc.

**Checking the effect on new users**

* After adding your content to /etc/skel, create a new user:

sudo useradd user5

* Look inside their home directory to see the copied files:

ls -la /home/user5

* You’ll see the .bashrc, .profile, .bash\_logout, and now the doc symbolic link as well.

**What about existing users?**

* **Their home directories won’t get updated automatically.**
* If you want existing users to have new content, you’d need to copy files manually or provide instructions for users to update their configs.

**Practical examples of what you can put in /etc/skel**

* Custom **welcome message file** (e.g., WELCOME.txt).
* Default **bash aliases** or environment variables in .bashrc.
* Company **policy documents** or **scripts**.
* A **custom .vimrc or .gitconfig** file.
* **Directories** with templates or starter files.

**Summary**

| **What** | **Explanation** |
| --- | --- |
| /etc/skel | Template directory for new user home contents |
| Hidden dotfiles | Shell configs like .bashrc, .profile, .bash\_logout |
| Files copied on user creation | When home directory is made for a new user |
| Symbolic links | Can be added to /etc/skel to provide shortcuts |
| Only new users affected | Existing users’ home dirs remain unchanged |

If you want, I can help you with specific commands or scripts to customize /etc/skel for your needs! Would you like to see how to create a custom .bashrc or add a welcome message?

Absolutely! Let’s dive **deeper and step-by-step** into managing users in Linux — focusing on **modifying users** with usermod and **deleting users** with userdel. I’ll explain each part carefully with examples and what you need to watch out for.

**Part 1: Modifying Users with usermod**

**What is usermod?**

* The usermod command lets you **change properties** of an existing user account.
* You can change things like:
  + Groups the user belongs to
  + User’s login shell
  + Home directory
  + Username, user ID
  + Password expiry info, etc.

**Important option: -G (groups)**

* -G is used to set **secondary groups** (additional groups beyond the primary one).
* Example:
* sudo usermod -G staff user2

This sets the secondary group list of user2 to only the group staff.

**BUT here’s the catch:**

* When you use -G **without** -a (append), **you overwrite** the user’s current secondary groups.
* So if user2 was already a member of sudo and admin, now they belong **only** to staff.

**How to append groups safely**

* To **add** groups without removing the existing ones, use both -a and -G:
* sudo usermod -a -G admin,sudo user2
* This **adds** the admin and sudo groups to user2’s existing secondary groups, preserving old ones.

**Check which groups a user belongs to**

You can see all groups a user belongs to with:

id user2

or

groups user2

**Part 2: Deleting Users with userdel**

**What is userdel?**

* userdel deletes a user account from the system.
* But by default, it **does not** delete the user’s home directory or files.

**Delete user’s home directory and mail files with -r**

* To remove the user and their **home directory + mail spool file + crontab**, use:
* sudo userdel -r user2
* This deletes:
  + /home/user2 (user’s home folder)
  + User’s mail spool (usually in /var/mail or /var/spool/mail)
  + User’s crontab (scheduled jobs)

**What userdel -r does NOT do**

* It **won’t delete** files owned by the user outside their home directory.
* For example, files under /var, /tmp, or shared folders might still exist.

**Find files owned by deleted users (orphaned files)**

* Files with no user owner can cause confusion and clutter.
* You can search for files without a valid user using:
* sudo find /home -nouser -print
* sudo find /var -nouser -print
* This shows files that don’t belong to any user (because the user was deleted).

**Delete orphaned files if confirmed**

* If you are sure these files are not needed, delete them with:
* sudo find /home -nouser -delete
* sudo find /var -nouser -delete

**Part 3: Best practices before deleting a user**

* **Check the user’s files** for important data or work.
* You might want to **archive** or back up the files first.
* Review their **email inbox** if relevant.
* Communicate with your team about who should take over any work.

**Recap with commands example**

**1. Append groups to user2 without removing old groups**

sudo usermod -a -G admin,sudo user2

id user2

**2. Delete user3 and their home directory + mail files**

sudo userdel -r user3

**3. Find orphaned files left by deleted users in /home and /var**

sudo find /home -nouser -print

sudo find /var -nouser -print

**4. Delete those orphaned files if safe**

sudo find /home -nouser -delete

sudo find /var -nouser -delete

**Extra tip: Use !! to repeat last command with extra options**

Say you ran:

sudo find /home -nouser -print

To repeat the same command but add -delete at the end, just run:

!! -delete

If you want, I can also create a **step-by-step script or demo** for you to practice these commands safely on a test machine. Would you like that? Or do you want me to explain any part in even more detail?

Absolutely! Here's a **simple explanation** of what the speaker is saying — broken down in plain English:

**👤 Managing Users in Linux (Simplified)**

This is a hands-on session — just like your **real exam** will be. It’s **not multiple choice**, you need to actually **use commands** to do things.

**🔍 Working with Existing Users**

When we list users' **home directories**, we can **see which users exist** on the system.

Let’s focus on:

1. **Modifying users**
2. **Deleting users**
3. **Finding leftover files** from deleted users

**🧑‍💻 Example: user2**

We had a user named user2, and we added them to extra groups like sudo and admin.

* **Primary group** = the default group for a user
  + Files you create will be **owned by this group**
* **Secondary groups** = extra groups you belong to
  + Give you access to **shared resources**, like sudo access

If we look at user2 using the command:

id user2

We can see all the groups they belong to.

**⚠️ Modifying Group Membership Mistake**

We tried to add user2 to a new group (staff) using this:

sudo usermod -G staff user2

But **oops!** This **overwrites** their group list — removing sudo and admin.

So now, user2 only belongs to staff.

✅ **Fix**: Use -aG to **append** instead of overwrite:

sudo usermod -aG admin,sudo,staff user2

Now they belong to all the correct groups again!

**❌ Deleting Users**

We can delete users with:

sudo userdel user2

But this only removes the **account**, **not the files**. Their **home directory still exists**, but we can’t tell who it belongs to — the username is gone.

If we want to delete the **account and files**, use:

sudo userdel -r user3

* The -r flag = **remove home folder and mail**

**🔎 Finding Leftover Files**

Sometimes, files still exist that **belonged to deleted users**.

We can search for them using:

sudo find /home /var -nouser

This shows files that don’t belong to **any existing user**.

Once you're sure the files can be deleted, use:

sudo find /home /var -nouser -delete

Or use a shortcut:

sudo !!

!! repeats the last command — and we can just **add -delete** at the end.

**✅ Final Result**

Now when we list /home, only the remaining user folders are there — like user5. The old users’ home directories are gone.

**💡 Takeaway:**

* Use usermod -aG to **add groups** without removing old ones
* Use userdel -r to **delete users and their files**
* Use find -nouser to find **leftover files**
* Use !! to **repeat commands** quickly

Let me know if you want examples or a quick cheat sheet!

**✅ Summary (Quick Commands):**

| **Action** | **Command** |
| --- | --- |
| View groups | cat /etc/group or getent group |
| Create group | sudo groupadd sales |
| Rename group | sudo groupmod -n grp-sales sales |
| Delete group | sudo groupdel grp-sales |
| View group info | getent group sudo |

Of course! Here's a **simple summary** of what that final part is saying — in easy-to-understand words:

**🌅 End of the Module: Managing Users and Groups**

You're now **one step closer** to becoming a Linux guru! Let's review what you learned:

**👤 User Accounts**

* **Stored in**: /etc/passwd
* To view or search users:
* getent passwd

You can:

* **Add a user**: useradd
* **Change a user**: usermod
* **Delete a user**: userdel

**👥 Group Accounts**

* **Stored in**: /etc/group
* To view or search groups:
* getent group

You can:

* **Add a group**: groupadd
* **Rename a group**: groupmod -n newname oldname
* **Delete a group**: groupdel

**🗑️ Deleting Users – Be Careful!**

* If you run userdel **without -r**, the **user is deleted**, but their **files stay** (like their home folder).
* You can find these leftover files with:
* find /home /var -nouser
* And delete them with:
* find /home /var -nouser -delete

**📇 User Info (GECOS / Finger Data)**

This is extra info stored for each user:

* Full name
* Office number
* Work phone
* Home phone

To set it:

* Use chfn (change finger info)
* **Standard users** can’t set their full name
* **Admins (root)** can set full name using:
* sudo chfn -f "Full Name" username

**👀 To Read This Info Easily**

Install and use:

sudo apt install finger

finger username

It shows user info in a **nice and clean format**.

**✅ Summary of Commands:**

| **Purpose** | **Command** |
| --- | --- |
| View users/groups | getent passwd / getent group |
| Add user/group | useradd / groupadd |
| Modify user/group | usermod / groupmod |
| Delete user/group | userdel / groupdel |
| Set extra user info | chfn |
| View user info nicely | finger username |
| Find leftover files | find / -nouser -delete |

When you're comfortable with this, the **next topic** will be:  
🔐 **Managing Passwords and Authentication in Linux**

Let me know if you'd like a cheat sheet or practice questions!