

User and Group Management

LinuXpert Systems

No.5, Dr. Ambedkar St, Rathinapuram,

Nanganallur, Chennai – 600 061.

Phone: 98841 65649, 90808 01382

E-mail: baskar@linuxpert.in

TABLE OF CONTENTS

1. useradd 1

2. usermod 4

3. userdel 7

4. passwd 9

5. groupadd 12

6. groupmod 15

7. groupdel 18

8. id 21

9. whoami 26

10. groups 31

1. useradd

`useradd` is a Linux command used to create a new user account. It is a low-level utility available on most Linux distributions. By default, it creates the account, but many user settings (such as the home directory) depend on the options you provide and the system configuration.

Basic syntax

```
useradd [options] username
```

Common examples

1. Create a new user

```
sudo useradd alice
```

Creates a user named `alice`.

---

2. Create a user with a home directory

```
sudo useradd -m alice
```

- `-m` creates the home directory (for example, `/home/alice`).

---

3. Create a user with a specific shell

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

- `-s` specifies the login shell.

---

4. Set a custom home directory

```
sudo useradd -m -d /home/developers/alice alice
```

- `-d` specifies the home directory path.

5. Add the user to a group

```
sudo useradd -m -G sudo,docker alice
```

- `-G` adds the user to supplementary groups.

Common options

Option	Description
`-m`	Create the user's home directory.
`-d DIR`	Specify the home directory.
`-s SHELL`	Set the login shell.
`-u UID`	Assign a specific user ID.
`-g GROUP`	Set the primary group.
`-G GROUPS`	Add to supplementary groups.
`-c COMMENT`	Add user information (full name or description).
`-e DATE`	Set an account expiration date.

Setting a password

`useradd` does **not** set a password. After creating the user, use:

```
sudo passwd alice
```

You'll be prompted to enter and confirm the new password.

What happens internally?

When you run:

```
sudo useradd -m alice
```

Linux typically:

- Creates a new entry for `alice` in `/etc/passwd`.
- Creates a group for the user (depending on system configuration).
- Creates the home directory `/home/alice`.
- Copies default files from `/etc/skel` into the home directory.
- Assigns a unique User ID (UID).

`useradd` vs. `adduser`

On many Linux distributions:

- **`useradd`**: A lower-level, less interactive command that requires you to specify options explicitly.
- **`adduser`**: A higher-level, user-friendly script (common on Debian and Ubuntu) that interactively prompts for a password, full name, and other details while configuring the account.

For example:

```
sudo adduser alice
```

This guides you through creating the user and setting a password, making it more convenient for interactive use.

In summary, `useradd` is best when you want precise control or are writing scripts, while `adduser` is generally more convenient for creating users manually on systems where it is available.

2. usermod

`usermod` is a Linux command used to **modify the properties of an existing user account**. It is commonly used by system administrators to change a user's login name, home directory, shell, group memberships, account expiry, and more.

Basic syntax

```
usermod [options] username
```

Here, `username` is the existing user account you want to modify.

Common options

Option	Description	Example
`-aG`	Add the user to supplementary groups (without removing existing ones)	`sudo usermod -aG sudo alice`
`-G`	Set the user's supplementary groups (replaces existing groups unless used with `-a`)	`sudo usermod -G developers alice`
`-l`	Change the login name	`sudo usermod -l newname oldname`
`-d`	Change the home directory	`sudo usermod -d /home/newhome alice`
`-m`	Move the contents of the old home directory to the new one (used with `-d`)	`sudo usermod -d /home/alice_new -m alice`
`-s`	Change the login shell	`sudo usermod -s /bin/bash alice`
`-L`	Lock the user account	`sudo usermod -L alice`
`-U`	Unlock the user account	`sudo usermod -U alice`
`-e`	Set the account expiration date	`sudo usermod -e 2026-12-31 alice`

Examples

1. Add a user to the `sudo` group

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

- `-a` = append
- `-G` = supplementary group
- Keeps existing groups and adds `sudo`.

---

2. Change the user's shell

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

The user's default shell becomes `/bin/zsh`.

---

3. Rename a user

```
sudo usermod -l bob alice
```

Changes the username from `alice` to `bob`.

---

4. Move the home directory

```
sudo usermod -d /home/bob -m bob
```

Changes the home directory and moves the existing files.

Important note about `-aG`

A common mistake is:

```
sudo usermod -G docker alice
```

This **replaces all** of Alice's supplementary groups with only `docker`.

To **add** the user to a group while keeping existing memberships, use:

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

Requirements

- Typically requires **root** or **sudo** privileges.
- The user being modified should ideally not be actively logged in for changes like renaming or moving the home directory.

In summary, `usermod` is the standard Linux utility for changing the configuration of an existing user account without deleting and recreating it.

3. userdel

`userdel` is a Linux/Unix command used to **delete a user account** from the system. It removes the user's account information, and optionally their home directory and mail files.

Syntax

```
userdel [options] username
```

Common options

Option	Description
`-r`	Remove the user's home directory and mail spool along with the account.
`-f`	Force deletion, even if the user is still logged in or other issues exist (use with caution).
`-Z`	Remove any SELinux user mapping (on SELinux-enabled systems).

Examples

Delete a user account only

```
sudo userdel alice
```

This removes the user account but leaves the user's home directory (for example, `/home/alice`) and files intact.

Delete a user account and its home directory

```
sudo userdel -r alice
```

This removes:

- The user account
- The user's home directory
- The user's mail spool (if present)

How it works

When you run `userdel`, it typically:

1. Removes the user's entry from `/etc/passwd`.
2. Removes the user's password entry from `/etc/shadow`.
3. Removes the user's group entry from `/etc/group` if appropriate.
4. Deletes the home directory and mail files only if `-r` is specified.

Permissions

You generally need **root privileges** to delete users:

```
sudo userdel username
```

Important notes

- You usually cannot delete a user who is currently logged in without using force (and even then, it may not be advisable).
- Using `-r` permanently deletes the user's home directory and its contents, so ensure any important data has been backed up.
- Files owned by the deleted user outside their home directory are **not** automatically removed. They remain on the system with the deleted user's numeric user ID (UID) until ownership is changed or the files are deleted.

Summary

`userdel` is the standard Linux command for removing user accounts:

- `userdel username` → Deletes the account only.
- `userdel -r username` → Deletes the account, home directory, and mail files.
- Typically requires `sudo` or root access.

4. passwd

`passwd` is a Linux/Unix command used to **set or change a user's password**. It can also be used by system administrators to manage password-related settings, such as locking or expiring passwords.

Syntax

```
passwd [options] [username]
```

- If **no username** is specified, the command changes the password of the currently logged-in user.
- If a **username** is specified, you generally need root (or `sudo`) privileges to change that user's password.

Common options

Option	Description
`-l`	Lock a user's password (disable password login).
`-u`	Unlock a previously locked password.
`-d`	Delete a user's password (creates a passwordless account if other authentication isn't enforced).
`-e`	Expire the password immediately, forcing the user to change it at the next login.
`-S`	Display password status information for a user.

Examples

Change your own password

```
passwd
```

You'll be prompted to:

1. Enter your current password.
2. Enter a new password.
3. Confirm the new password.

Change another user's password

```
sudo passwd alice
```

The administrator sets a new password for the user `alice`.

Lock a user account's password

```
sudo passwd -l alice
```

This prevents password-based logins for `alice` until the password is unlocked.

Unlock a user's password

```
sudo passwd -u alice
```

Force a password change at next login

```
sudo passwd -e alice
```

How it works

When you change a password:

1. `passwd` verifies your identity (if required).
2. The new password is checked against the system's password policy (such as minimum length or complexity).
3. The password is securely hashed.
4. The hashed password is stored in `/etc/shadow` (not in plain text).

Permissions

- **Regular users** can change only their own passwords:
  `passwd`
- **Administrators** can change any user's password:
  `sudo passwd username`

Important notes

- Passwords are **not stored in plain text**; only secure hashes are stored.
- A strong password should be long, unique, and difficult to guess.
- Locking a password with `passwd -l` disables password authentication but does not necessarily disable other login methods, such as SSH keys.

Summary

`passwd` is the standard Linux command for managing user passwords:

- `passwd` → Change your own password.
- `sudo passwd username` → Change another user's password.
- `sudo passwd -l username` → Lock password-based login.
- `sudo passwd -u username` → Unlock password-based login.
- `sudo passwd -e username` → Force a password change at the next login.

5. groupadd

`groupadd` is a Linux/Unix command used to **create a new group** on the system. Groups are used to organize users and manage permissions for files, directories, and other resources.

Syntax

```
groupadd [options] groupname
```

- `groupname` is the name of the new group you want to create.

Common options

Option	Description
`-g GID`	Specify a Group ID (GID) for the new group.
`-f`	Exit successfully if the group already exists; if used with `-g`, chooses another GID if needed.
`-r`	Create a system group (typically with a lower GID, depending on the system configuration).
`-K KEY=VALUE`	Override default settings (such as GID ranges) for this command.

Examples

Create a new group

```
sudo groupadd developers
```

This creates a group named `developers`.

Create a group with a specific GID

```
sudo groupadd -g 2000 developers
```

This creates the `developers` group with Group ID (GID) `2000`.

Create a system group

```
sudo groupadd -r nginx
```

This creates a system group named `nginx`, commonly used for system services.

How it works

When you run `groupadd`:

1. The system checks whether the group name already exists.
2. A unique Group ID (GID) is assigned automatically unless you specify one.
3. The group information is added to the system's group database (typically `/etc/group`).
4. If group passwords are used (rare on modern systems), related entries may also be updated in `/etc/gshadow`.

Permissions

Creating groups requires administrator privileges:

```
sudo groupadd groupname
```

Verifying the group

To check that the group was created:

```
getent group developers
```

Or search the group file:

```
grep developers /etc/group
```

Related commands

- `groupdel` – Delete a group.
- `groupmod` – Modify a group's name or GID.
- `usermod -aG group username` – Add an existing user to a supplementary group.
- `groups username` – Display the groups a user belongs to.

Important notes

- Group names must be unique.
- Creating a group does **not** automatically add users to it.
- To add a user to a group, use:

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

  Here, `alice` becomes a member of the `developers` group.

Summary

`groupadd` is the standard Linux command for creating groups:

- `sudo groupadd developers` → Create a new group.
- `sudo groupadd -g 2000 developers` → Create a group with a specific GID.
- `sudo groupadd -r nginx` → Create a system group.

Groups make it easier to manage permissions by allowing multiple users to share access to files and resources.

6. groupmod

`groupmod` is a Linux/Unix command used to **modify an existing group**. You can use it to change a group's **name** or **Group ID (GID)**.

Syntax

```
groupmod [options] groupname
```

- `groupname` is the name of the existing group you want to modify.

Common options

Option	Description
`-n NEW_GROUP_NAME`	Change the group's name.
`-g GID`	Change the group's Group ID (GID).
`-o`	Allow a non-unique GID (used with `-g`). Use with caution.

Examples

Rename a group

```
sudo groupmod -n developers programmers
```

This renames the group `programmers` to `developers`.

Change a group's GID

```
sudo groupmod -g 2000 developers
```

This changes the GID of the `developers` group to `2000`.

Allow a duplicate GID

```
sudo groupmod -o -g 2000 developers
```

This assigns GID `2000` even if another group already uses it. This is rarely needed and can make permission management more confusing.

How it works

When you run `groupmod`, the system:

1. Verifies that the specified group exists.
2. Applies the requested changes (such as a new name or GID).
3. Updates the system's group database (typically `/etc/group` and `/etc/gshadow`).

Permissions

You need administrator privileges to modify groups:

```
sudo groupmod [options] groupname
```

Verifying the changes

To check the updated group information:

```
getent group developers
```

Or search the group file:

```
grep developers /etc/group
```

Important notes

- Renaming a group does **not** remove users from it. Members remain in the group after the name change.
- Changing a group's GID can affect file ownership. Files associated with the old GID may need to have their group ownership updated manually. For example:

  ```
  sudo find / -group OLD_GID -exec chgrp developers {} \;
  ```

- Avoid using duplicate GIDs unless you have a specific administrative reason.

Related commands

- `groupadd` – Create a new group.
- `groupdel` – Delete an existing group.
- `usermod -aG group username` – Add a user to a supplementary group.
- `groups username` – Display the groups a user belongs to.

Summary

`groupmod` is the standard Linux command for modifying groups:

- `sudo groupmod -n developers programmers` → Rename a group.
- `sudo groupmod -g 2000 developers` → Change a group's GID.
- `sudo groupmod -o -g 2000 developers` → Assign a duplicate GID (not generally recommended).

It is primarily used by system administrators when reorganising groups or updating group identifiers.

7. groupdel

`groupdel` is a Linux/Unix command used to **delete an existing group** from the system. It removes the group's entry from the system's group database.

Syntax

```
groupdel groupname
```

- `groupname` is the name of the group you want to delete.

Example

Delete a group

```
sudo groupdel developers
```

This removes the `developers` group from the system.

How it works

When you run `groupdel`:

1. The system checks whether the group exists.
2. It verifies that the group is not the **primary group** of any user.
3. If the checks pass, the group is removed from the system's group database (typically `/etc/group` and `/etc/gshadow`).

Permissions

Deleting a group requires administrator privileges:

```
sudo groupdel groupname
```

Verify the deletion

To confirm the group has been deleted:

```
getent group developers
```

If no output is displayed, the group no longer exists.

You can also check:

```
grep developers /etc/group
```

Common error

If the group is the primary group of a user, you may see an error similar to:

```
groupdel: cannot remove the primary group of user 'alice'
```

To resolve this:

1. Change the user's primary group:

   ```
   sudo usermod -g users alice
   ```

2. Then delete the old group:

   ```
   sudo groupdel developers
   ```

Important notes

- `groupdel` **does not delete users** who belong to the group.
- Users simply lose membership in the deleted group if it was a supplementary group.
- Files owned by the deleted group's **Group ID (GID)** are **not automatically updated**. They may still be associated with the old numeric GID until you change their group ownership.

For example, to change the group ownership of files:

```
sudo chgrp newgroup filename
```

Or recursively for a directory:

```
sudo chgrp -R newgroup /path/to/directory
```

Related commands

- `groupadd` – Create a new group.
- `groupmod` – Modify an existing group.
- `usermod -aG group username` – Add a user to a supplementary group.
- `groups username` – Display the groups a user belongs to.

Summary

`groupdel` is the standard Linux command for deleting groups:

- `sudo groupdel developers` → Deletes the `developers` group.
- Requires administrator (`sudo`) privileges.
- Cannot delete a group if it is the primary group of any user.
- Does not delete users or automatically update file group ownership.

8. id

`id` is a Linux/Unix command used to **display information about a user**, including the user's **UID (User ID), GID (Group ID), and group memberships**.

It is commonly used by system administrators to check user identity and permissions.

Syntax

```
id [options] [username]
```

- If no username is provided, `id` displays information about the **currently logged-in user**.
- If a username is provided, it displays information about that user.

Basic Example

```
id
```

Example output:

```
uid=1000(alice) gid=1000(alice) groups=1000(alice),27(sudo),1001(developers)
```

Understanding the output

Field	Meaning
`uid=1000(alice)`	User ID and username
`gid=1000(alice)`	Primary Group ID and group name
`groups=...`	All groups the user belongs to

Examples

Display information about the current user

```
id
```

Example:

```
uid=1001(bob) gid=1001(bob) groups=1001(bob),10(wheel)
```

This tells you:

- User name: `bob`
- UID: `1001`
- Primary group: `bob`
- Additional group: `wheel`

Check another user's information

```
id alice
```

Example output:

```
uid=1002(alice) gid=1002(alice) groups=1002(alice),1003(developers)
```

Common options

Option	Description
`-u`	Display only the user's UID
`-g`	Display only the primary GID
`-G`	Display all group IDs
`-n`	Display names instead of numbers (used with `-u`, `-g`, or `-G`)

Examples with options

Show only the UID

```
id -u alice
```

Output:

```
1002
```

Show only the primary GID

```
id -g alice
```

Output:

```
1002
```

Show all group IDs

```
id -G alice
```

Output:

```
1002 1003 27
```

Show group names instead of IDs

```
id -Gn alice
```

Output:

```
alice developers sudo
```

How it works

The `id` command gets user and group information from system databases such as:

- `/etc/passwd` → User account information
- `/etc/group` → Group information
- `/etc/nsswitch.conf` → Defines where user/group information is retrieved from (local files, LDAP, etc.)

Difference between UID and GID

UID (User ID)

- A unique number assigned to each user.
- Used by Linux to identify file ownership and permissions.

Example:

```
uid=1000(alice)
```

GID (Group ID)

- A unique number assigned to each group.
- Used to manage shared access to files.

Example:

```
gid=1000(alice)
```

Related commands

- `whoami` → Shows only the current username.
- `groups` → Shows groups a user belongs to.
- `users` → Shows currently logged-in users.
- `getent passwd` → Displays user account information.

Summary

`id` displays user identity information:

```
id
```

→ Show current user's UID, GID, and groups.

```
id username
```

→ Show information about a specific user.

```
id -u username
```

→ Show only the user's UID.

```
id -Gn username
```

→ Show all group names for a user.

9. whoami

`whoami` is a Linux/Unix command used to **display the username of the current user** who is logged into the system.

It answers the question: **"Who am I logged in as?"**

Syntax

```
whoami
```

No options or arguments are usually required.

Example

Run:

```
whoami
```

Output:

```
alice
```

This means the current session is running as the user `alice`.

How it works

`whoami` displays the effective username of the current user. Internally, it uses the user's **effective User ID (EUID)** and looks up the corresponding username.

For example:

```
uid=1000(alice)
```

The command will display:

```
alice
```

Using with `sudo`

A common use of `whoami` is checking whether you are running commands as another user.

Example:

```
whoami
```

Output:

```
alice
```

Run:

```
sudo whoami
```

Output:

```
root
```

This shows that the command was executed with administrator privileges.

Difference between `whoami` and `id`

`whoami`

Shows only the current username:

```
whoami
```

Output:

```
alice
```

`id`

Shows detailed user information:

```
id
```

Output:

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

`id` provides:

- User ID (UID)
- Primary Group ID (GID)
- Group memberships

Difference between `whoami` and `who`

`whoami`

Shows **your current username**:

```
whoami
```

Output:

```
alice
```

`who`

Shows **all users currently logged into the system**:

```
who
```

Example output:

```
alice   pts/0   10:30
bob     pts/1   10:45
```

Common uses

1. Check the current logged-in user

```
whoami
```

Useful when working on servers with multiple accounts.

---

2. Verify root access

```
sudo whoami
```

Output:

```
root
```

Confirms that `sudo` is working.

---

3. Use in scripts

Example:

```
echo "Running script as $(whoami)"
```

Output:

```
Running script as alice
```

Related commands

Command	Purpose
`id`	Shows UID, GID, and groups
`who`	Shows users currently logged in
`w`	Shows logged-in users and their activities
`users`	Lists logged-in usernames
`su`	Switch to another user

Summary

`whoami` is a simple Linux command that shows the **current effective username**.

Examples:

```
whoami
```

→ Displays your username.

```
sudo whoami
```

→ Displays `root` if the command runs with administrator privileges.

It is mainly used to quickly verify **which user account is active in the current terminal session**.

10. groups

The command **`groups`** can mean different things depending on the context. In most cases, it refers to the **Linux/Unix `groups` command**.

Linux `groups` command

The `groups` command displays the **groups that a user belongs to**.

Syntax

```
groups [username]
```

Examples

1. Show your own groups

```
groups
```

Example output:

```
john adm cdrom sudo docker
```

Meaning:

- `john` → your username
- `adm`, `cdrom`, `sudo`, `docker` → groups your user is a member of

---

2. Show groups for another user

```
groups alice
```

Output:

```
alice : alice sudo developers
```

This means the user `alice` belongs to:

- `alice` (primary group)
- `sudo`
- `developers`

Why are groups important in Linux?

Linux uses **users and groups** to manage permissions.

For example:

```
ls -l file.txt
```

Output:

```
-rw-r----- 1 john developers 2048 Jul 25 file.txt
```

Here:

- Owner: `john`
- Group: `developers`
- Permissions:
    - `john` can read/write
    - Members of `developers` can read
    - Others have no access

If your user is in the `developers` group, you can access files assigned to that group.

Related commands

Show detailed user information

```
id
```

Example:

```
uid=1000(john) gid=1000(john) groups=1000(john),27(sudo),1001(developers)
```

Add a user to a group

```
sudo usermod -aG developers john
```

List all groups on the system

```
cat /etc/group
```

Change file group ownership

```
chgrp developers file.txt
```

So, in short:

`groups` = "Show which Linux groups a user belongs to."

It is mainly used for checking permissions and managing access to files, services, and system resources.