

Which is the command used in linux to check all the running processes

In Linux, you can use the **ps** command or **top** command to check all running processes. Here are the details:

1. ps Command:

The **ps** command displays information about the processes running on the system. To see all running processes, use the following options:

```
Copy code
ps aux
  ○ a: Shows processes for all users.
  ○ c: Displays user-oriented format.
  ○ e: Includes processes without a controlling terminal.
  Example output:

    ugid
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 18544 1580 ?        Ss   09:15   0:01 init
```

2. top Command:

The **top** command provides a real-time, dynamic view of system processes. Simply type:

```
Copy code
top

It shows process information along with CPU and memory usage, sorted dynamically by resource usage.
3. htop Command (Optional, if installed):
A more user-friendly alternative to top that allows navigation using arrow keys:
```

```
Copy code
htop

4. ps -ef:
Another format for viewing processes with detailed information:
```

```
Copy code
ps -ef
  ○ -e: Select all processes.
  ○ -f: Full-format listing.
```

How to kill a process

In Linux, you can kill a process using the **kill** command, **killall** command, or **htop** utility. Here's how:

1. Using the kill Command

- Find the **Process ID (PID)** of the process you want to kill using:

```
perl
Copy code
ps aux | grep <process_name>

Or use top or htop to locate the PID.
• Kill the process by specifying its PID:
```

```
bash
kill <PID>

Example:
```

```
bash
Copy code
kill 1234

• To forcefully terminate the process, use the -9 signal:
```

```
bash
Copy code
kill -9 <PID>
```

2. Using the killall Command

- Use **killall** to terminate all processes with a specific name:

```
php
Copy code
killall <process_name>

Example:
```

```
Copy code
killall firefox

• To forcefully terminate:
```

```
php
Copy code
killall -9 <process_name>
```

3. Using htop Utility (if installed)

- Open **htop**.
- Use the groupadd command to create a new group:
- Navigate to the process you want to kill using the arrow keys.
- Press **F9** to kill the process.
- Confirm the kill signal (e.g., **SIGTERM** or **SIGKILL**).

Can you tell me the complete command to add a user and group in linux?

1. Add a Group

Use the **groupadd** command to create a new group:

```
bash
Copy code
sudo groupadd <group_name>
Example:
```

```
bash
Copy code
sudo groupadd developers
```

2. Add a User

Use the **useradd** command to create a new user:

```
bash
Copy code
sudo useradd -m -s /bin/bash -g <group_name> <username>
Explanation:
  • -m: Creates a home directory for the user.
  • -s /bin/bash: Sets the default shell to bash.
  • -g <group_name>: Assigns the primary group to the user.
Example:
```

```
bash
Copy code
sudo useradd -m -s /bin/bash -g developers john
```

3. Add a User to Additional Groups

If you want to add the user to more groups:

```
bash
Copy code
sudo usermod -aG <group_name> <username>
Example:

bash
Copy code
sudo usermod -aG docker john
```

4. Set a Password for the User

After creating the user, set a password:

```
bash
Copy code
sudo passwd <username>
Example:
```

```
bash
Copy code
sudo passwd john
```

5. Verify the User and Group

- Check user details:

```
bash
Copy code
id <username>

Example:
```

How will you create 3 directories in one command Eg: a/b/c ?

To create a nested directory structure like **a/b/c** in one command, use the **mkdir** command with the **-p** option:

Command:

```
bash
Copy code
mkdir -p a/b/c

Explanation:
  • mkdir: Command to create directories.
  • -p: Ensures that parent directories (a and b in this case) are created automatically if they do not exist.

Result:
  • Directory a is created if it doesn't exist.
  • Inside a, directory b is created.
  • Inside b, directory c is created.

Example:
```

```
bash
Copy code
mkdir -p project/src/logs

This command creates the project directory, with src inside it, and logs inside src.
```

What is the use of mv command?

The **mv** command in Linux is used for **moving** or **renaming** files and directories. Here's how it works:

Primary Uses:

- 1. Move Files or Directories**
- Moves a file or directory from one location to another.
- Syntax:

```
bash
Copy code
mv <source> <destination>

• Example:
```

```
bash
Copy code
mv file.txt /home/user/documents/

This moves file.txt to the /home/user/documents/ directory.
```

2. Rename Files or Directories

- Renames a file or directory by "moving" it to the same location with a new name.
- Syntax:

```
php
Copy code
mv <old_name> <new_name>

• Example:
```

```
bash
Copy code
mv oldfile.txt newfile.txt

This renames oldfile.txt to newfile.txt.
```

3. Move Multiple Files to a Directory

- Move multiple files into a directory at once.
- Syntax:

```
php
Copy code
mv <file1> <file2> ... <destination_directory>

• Example:
```

```
bash
Copy code
mv file1.txt file2.txt /home/user/documents/
```

How to start any service in Linux?

In Linux, you can start a service using **init** systems like **systemd**, **systeminit**, or others depending on your distribution. Here's how to start a service based on the most commonly used systems:

1. Using systemd (Most Modern Distros, e.g., Ubuntu 16.04+, CentOS 7+, Fedora)

- Command to Start a Service:**

```
bash
Copy code
sudo systemctl start <service_name>

Example:
```

```
bash
Copy code
sudo systemctl start nginx

• Check Service Status:
```

```
bash
Copy code
sudo systemctl status <service_name>

• Enable Service to Start on Boot:
```

```
bash
Copy code
sudo systemctl enable <service_name>
```

2. Using Service Command (Older Distros)

- Command to Start a Service:**

```
bash
Copy code
sudo service <service_name> start

Example:
```

```
bash
Copy code
sudo service apache2 start

• Check Service Status:
```

```
bash
Copy code
sudo systemctl status <service_name> status
```

3. Using init.d Scripts (Legacy Systems)

- Command to Start a Service:**

```
bash
Copy code
sudo /etc/init.d/<service_name> start

Example:
```

```
bash
Copy code
sudo /etc/init.d/sshd start
```

4. Using upstart (Ubuntu 14.04 and Earlier)

- Command to Start a Service:**

```
bash
Copy code
sudo start <service_name>

Example:
```

```
bash
Copy code
sudo start mysql
```

Verifying Service Startup

After starting the service, you can verify that it is running:

- Check Active Services (with **systemd**):

```
bash
Copy code
systemctl list-units --type=service --state=active

• Process Check (General):

bash
Copy code
ps aux | grep <service_name>
```

Common Services You May Work With

- Web Server: nginx, apache2
- Database: mysql, postgresql

What is bashrc?

The **.bashrc** file is a script that is executed whenever a new interactive shell session is started (**lnash** (Bourne Again Shell)). It is a configuration file for the bash shell that allows you to customize your shell environment.

Key Features and Use of .bashrc:

- Shell Customization**
 - You can define environment variables, shell options, and aliases to enhance your workflow.
- Aliases**
 - You can create shortcuts for commands.
- Environment Variables**
 - Set environment variables that apply to your shell session.

- Shell Behavior**
 - Configure shell behavior with options like **shopt** or settings like **HISTCONTROL**.

```
bash
Copy code
shopt -s histappend
export HISTCONTROL=ignore-dups:erasedups
```

- Startup Scripts**
 - Include custom scripts or commands that should run every time a new shell session is initiated.

- Aliases for Custom Scripts**
 - Link frequently used scripts or commands to simple aliases.

Location of .bashrc

- The **.bashrc** file is usually located in the user's home directory:

```
bash
Copy code
~/.bashrc
```

How to Apply Changes in .bashrc

After editing **.bashrc**, apply the changes by running:

```
bash
Copy code
source ~/.bashrc
or simply:

bash
Copy code
. ~/.bashrc
```

What is profile in Linux?

In Linux, **/etc/profile** and **~/.profile** are configuration files used to set environment variables and startup commands for shell sessions. These files are executed when a user logs in, making them key components for configuring a login shell.

Types of Profile Files:

- Global Profile (/etc/profile):**
 - Located at **/etc/profile**.
 - Applies to all users on the system.
 - It sets system-wide environment variables and startup programs.
 - Often calls additional scripts in **/etc/profile.d/** for modular configurations.
- User Profile (~/.profile or ~/.bashrc):**
 - Located in the user's home directory (**~** or **\$HOME**).
 - Contains user-specific environment settings.
 - Only applies to the individual user who owns the file.

When is profile Loaded?

- Login Shells:** The profile files are executed when a user logs in via:
 - Text-based consoles (e.g., **TTY**).
 - SSH sessions**.
- Desktop environments**, if configured to start a login shell.
- For non-login shells, files like **~/.bashrc** are used instead.

Difference Between profile and bashrc:

Feature	~/.profile	~/.bashrc
Execution Context	Login shells	Non-login interactive shells
Purpose	Environment variables, startup commands	Aliases, functions, shell-specific commands
Trigger Method	Logging in (e.g., SSH)	Opening a terminal

Editing .profile:

- Open the file with a text editor:

```
bash
Copy code
nano ~/.profile
```
- Add your configurations, such as environment variables or scripts.
- Save the file and apply changes by logging out and logging back in, or by running:

```
bash
Copy code
source ~/.profile
```

How to change the ownership of a file/directories in linux?

In Linux, you can change the ownership of files or directories using the **chown** command. Here's how it works:

1. Basic Syntax

```
bash
Copy code
sudo chown [OPTIONS] <new_owner> <file/directory>
```

2. Change Ownership of a File

- Example:

```
bash
Copy code
sudo chown john file.txt
```

This changes the ownership of **file.txt** to the user **john**.

3. Change Ownership of a Directory

- Example:

```
bash
Copy code
sudo chown john /path/to/directory
```

4. Change Both Owner and Group

- Syntax:

```
bash
Copy code
sudo chown <new_owner>:<new_group> <file/directory>
```

- Example:

```
bash
Copy code
sudo chown john:developers file.txt
```

Feature	Initramps (Initial RAM Filesystem)	init (Initial Process)
What is it?	A temporary root filesystem loaded into RAM during early boot	The first user-space process started by the kernel
Purpose	Helps kernel load drivers, mount root filesystem	Starts and manages system services and daemons
Role in boot process	Comes before init in boot order	Comes after initramps
Located where?	Bundle with the kernel in /boot/initramfs.img	Located at /sbin/init, /lib/systemd/systemd, etc.
Used as a fallback?	Invoked only if other than kernel is loaded	After some filesystems are mounted

Which is the command used in linux to change the file permissions?

In Linux, the **chmod** command is used to change the permissions of a file or directory. You can modify who can read, write, or execute a file using this command.

1. Basic Syntax:

```
bash
Copy code
chmod [OPTIONS] <permissions> <file/directory>
```

2. File Permissions Format

Permissions are represented by a combination of **read (r)**, **write (w)**, and **execute (x)** permissions for the **user (u)**, **group (g)**, and **others (o)**.

- r (read):** Permission to read the file or directory.
- w (write):** Permission to modify the file or directory.
- x (execute):** Permission to run the file (if it is a script or binary), or traverse the directory.

3. Numeric Mode (Octal Representation)

Permissions can be specified using a 3-digit octal number:

- 4 = Read (r)**
 - 2 = Write (w)**
 - 1 = Execute (x)**
 - 0 = No permission**
- Each digit represents the permissions for **user**, **group**, and **others** respectively. Add the values to assign permissions.
- Examples:**
- 777 = rwxrwxrwx** (Read, Write, Execute for user, group, and others).
 - 755 = rwx-r-x** (Read, Write, Execute for user; Read, Execute for group and others).

4. Changing Permissions Using Numeric Mode

- Example:

```
bash
Copy code
chmod 755 file.txt
```

This command gives:

- User:** Read, Write, Execute (rwx)
- Group:** Read, Execute (r-x)
- Others:** Read, Execute (r-x)

- Another Example:**

```
bash
Copy code
chmod 644 file.txt
```

This command gives:

- User:** Read, Write (rw)
- Group:** Read (r-)
- Others:** Read (r-)

5. Changing Permissions Using Symbolic Mode

Permissions can also be modified symbolically using letters and operators (+, -, =).

- Common Operators:**
- +**: Add a permission.
 - : Remove a permission.
 - =**: Set permission exactly, removing others.

Examples:

- Add Execute Permission to User:**

```
bash
Copy code
chmod u+x file.txt
```

This adds execute permission to the user.

- Remove Write Permission from Group:**

```
bash
Copy code
chmod g-w file.txt
```

This removes write permission from the group.

- Set Permissions for User Only:**

```
bash
Copy code
chmod u=rwx file.txt
```

5. Verify the User and Group

- Check user details:

```
bash
Copy code
id <username>
```

Example:

```
bash
Copy code
id john
```

- List all groups:







```
bash
Copy code
cat /etc/group
```

- Process Check (Generic):

```
bash
Copy code
ps aux | grep <service_name>
```

Common Services You May Work With

- Web Server: nginx, apache2
- Database: mysql, postgresql
- SSH: sshd

It?	during early boot	by the kernel
 Purpose	Helps kernel load drivers , mount real root filesystem	Starts and manages system services and runlevels
 Role in boot process	Comes before init in boot order	Comes after <i>initramfs</i>
 Located where?	Bundled with the kernel in <i>/boot/initramfs.img</i>	Located at <i>/sbin/init</i> , <i>/lib/systemd/systemd</i> , etc.
 Used when?	Immediately after the kernel is loaded	After root filesystem is mounted
 Used for?	Hardware detection, disk decryption, RAID, drivers	Starting services, running scripts, system startup
 PID?	No PID (runs inside early user space)	Always PID 1 (first user-space process)