

System Information

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. uname 1

2. hostname 3

3. uptime 6

4. date 9

5. cal 13

6. who 16

7. w 20

8. last 24

9. lscpu 28

10. lsmem 34

11. lsblk 39

12. lspci 46

13. lsusb 52

1. uname

`uname` is a **Unix/Linux command** used to display information about the **system kernel and operating system**.

Basic usage

```
uname
```

Output example

```
Linux
```

This means the system is running the **Linux kernel**.

Breaking it down

- **`uname`** = **Unix NAME**
    - It prints the name and details of the operating system/kernel.

Common options

Command	Description	Example output
`uname -s`	Shows kernel name	`Linux`
`uname -n`	Shows network hostname	`my-computer`
`uname -r`	Shows kernel release version	`6.8.0-31-generic`
`uname -v`	Shows kernel version details	`#31-Ubuntu SMP...`
`uname -m`	Shows machine hardware architecture	`x86_64`
`uname -p`	Shows processor type (may not be supported everywhere)	`x86_64`
`uname -a`	Shows all available information	`Linux my-computer 6.8.0... x86_64 GNU/Linux`

Most commonly used form

```
uname -a
```

Example:

```
Linux ubuntu-pc 6.8.0-31-generic #31-Ubuntu SMP x86_64 GNU/Linux
```

Meaning:

- `Linux` → kernel name
- `ubuntu-pc` → hostname
- `6.8.0-31-generic` → kernel release
- `#31-Ubuntu SMP` → kernel build information
- `x86_64` → CPU architecture
- `GNU/Linux` → operating system family

When you use `uname`

It is useful for:

- checking what OS/kernel you are running
- troubleshooting compatibility issues
- finding the correct software package for your system
- checking whether a system is 32-bit or 64-bit

For example, before installing software, you might run:

```
uname -m
```

If it returns:

```
x86_64
```

you know your machine uses a 64-bit Intel/AMD architecture.

2. hostname

`hostname` is a **Linux/Unix command** used to display or manage the **name of a computer on a network**.

Basic usage

```
hostname
```

Example output

```
ubuntu-server
```

This means the computer's hostname is `ubuntu-server`.

Breaking it down

- **`host`** → a computer or device connected to a network
- **`name`** → the identifier assigned to that computer

So, **`hostname`** means "show the name of this computer."

Common options

Command	Description	Example
`hostname`	Displays the current hostname	`my-pc`
`hostname -s`	Shows the short hostname (without domain)	`server1`
`hostname -f`	Shows the fully qualified domain name (FQDN)	`server1.example.com`
`hostname -i`	Shows the IP address associated with the hostname	`192.168.1.10`
`hostname -I`	Shows all assigned IP addresses	`192.168.1.10 10.0.0.5`

Changing the hostname

You can temporarily change the hostname using:

```
sudo hostname new-name
```

Example:

```
sudo hostname web-server
```

After this command, the hostname becomes:

```
web-server
```

(Note: this change may not persist after reboot on many systems.)

Permanent hostname change (Linux)

On systems using `systemd`, use:

```
sudo hostnamectl set-hostname new-name
```

Example:

```
sudo hostnamectl set-hostname database-server
```

Check the change:

```
hostname
```

Output:

```
database-server
```

When you use `hostname`

It is useful for:

- identifying a machine on a network
- configuring servers
- troubleshooting network problems
- writing scripts that depend on the computer name
- managing multiple servers

Difference between `hostname` and `uname`

Command	Shows
`uname`	Information about the operating system/kernel
`hostname`	The computer's network name

Example:

```
uname -s
```

Output:

```
Linux
```

```
hostname
```

Output:

```
web-server
```

Together, they tell you:

"This is a Linux system, and its network name is web-server."

3. uptime

`uptime` is a **Linux/Unix command** used to show **how long a system has been running**, along with information about system load and logged-in users.

Basic usage

```
uptime
```

Example output

```
10:35:42 up 5 days, 3:20, 2 users, load average: 0.15, 0.10, 0.05
```

Breaking down the output

1. Current time

```
10:35:42
```

- The current system time.

---

2. System uptime

```
up 5 days, 3:20
```

- The computer has been running for:
  - **5 days**
  - **3 hours and 20 minutes**
- It shows the time since the last reboot.

---

3. Logged-in users

```
2 users
```

- The number of users currently logged into the system.

4. Load average

```
load average: 0.15, 0.10, 0.05
```

These are the system load averages for:

Value	Time period
0.15	Last 1 minute
0.10	Last 5 minutes
0.05	Last 15 minutes

What is load average?

Load average shows how much work the CPU and system are handling.

- On a **single-core CPU**:
  - `1.00` ≈ fully loaded
  - `2.00` ≈ overloaded (twice the available CPU capacity)
- On a **4-core CPU**:
  - `4.00` ≈ fully loaded
  - Values below `4.00` usually mean the system has available capacity

Example:

```
load average: 0.50, 0.40, 0.30
```

means the system is lightly loaded.

Useful options

Command	Description
`uptime -p`	Shows uptime in a human-readable format
`uptime -s`	Shows the exact time when the system started

Example:

```
uptime -p
```

Output:

```
up 5 days, 3 hours, 20 minutes
```

Example:

```
uptime -s
```

Output:

```
2026-07-20 07:15:10
```

When you use `uptime`

It is useful for:

- checking if a server has recently restarted
- monitoring server stability
- quickly checking system load
- troubleshooting performance issues

Difference between `uptime` and `uname`

Command	Purpose
`uname`	Shows system/kernel information
`uptime`	Shows how long the system has been running and its current load

Example:

```
uname -a
```

tells you:

"What kind of system is this?"

```
uptime
```

tells you:

"How long has this system been running, and how busy is it?"

4. date

`date` is a **Linux/Unix command** used to **display or set the system date and time**.

Basic usage

```
date
```

Example output

```
Sat Jul 25 14:30:45 UTC 2026
```

Breaking down the output

Example:

```
Sat Jul 25 14:30:45 UTC 2026
```

Part	Meaning
`Sat`	Day of the week
`Jul`	Month
`25`	Day of the month
`14:30:45`	Current time (hours:minutes:seconds)
`UTC`	Time zone
`2026`	Year

Common options

Command	Description	Example output
`date`	Displays current date and time	`Sat Jul 25 14:30:45 UTC 2026`
`date +%D`	Shows date in MM/DD/YY format	`07/25/26`
`date +%F`	Shows date in ISO format (YYYY-MM-DD)	`2026-07-25`
`date +%T`	Shows time only	`14:30:45`
`date -u`	Shows time in UTC	`Sat Jul 25 14:30:45 UTC 2026`

Custom date formatting

The + symbol allows you to choose the output format.

Example:

```
date "+%Y-%m-%d"
```

Output:

```
2026-07-25
```

Common format codes:

Code	Meaning	Example
%Y	Four-digit year	2026
%m	Month (01-12)	07
%d	Day of month	25
%H	Hour (24-hour format)	14
%M	Minutes	30
%S	Seconds	45
%A	Full weekday name	Saturday

Example:

```
date "+%A, %d %B %Y"
```

Output:

```
Saturday, 25 July 2026
```

Display a past or future date

You can calculate dates using `-d`:

Example:

```
date -d "tomorrow"
```

Output:

```
Sun Jul 26 14:30:45 UTC 2026
```

Example:

```
date -d "7 days ago"
```

Output:

```
Sat Jul 18 14:30:45 UTC 2026
```

Setting the system date (requires administrator privileges)

Example:

```
sudo date -s "2026-07-25 15:00:00"
```

This changes the system clock.

> On modern Linux systems, changing the time is usually managed with tools like `timedatectl` instead of directly using `date`.

When you use `date`

It is useful for:

- checking the current system time
- creating timestamps in scripts
- naming backup files with dates
- logging events
- scheduling and troubleshooting tasks

Difference between `date` and `uptime`

Command	Purpose
`date`	Shows the current date and time
`uptime`	Shows how long the system has been running

Example:

`date`

tells you:

"What time is it now?"

`uptime`

tells you:

"How long has this computer been running?"

5. cal

`cal` is a **Linux/Unix command** used to **display a calendar in the terminal**.

Basic usage

```
Cal
```

Example output

```
     July 2026
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

This shows the calendar for the **current month**.

Breaking down the output

Example:

```
     July 2026
```

- Displays the current month and year.

```
Su Mo Tu We Th Fr Sa
```

- Shows the days of the week:
    - Su → Sunday
    - Mo → Monday
    - Tu → Tuesday
    - We → Wednesday
    - Th → Thursday
    - Fr → Friday
    - Sa → Saturday

The numbers represent the days of the month.Common options

Command	Description
`cal`	Displays the current month's calendar
`cal -3`	Displays previous, current, and next month
`cal -y`	Displays the entire calendar for the current year
`cal 2025`	Displays the calendar for a specific year
`cal 7 2026`	Displays July 2026 calendar

Examples

Show the whole year

```
cal -y
```

Output:

```
                                2026

       January               February                March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
...
```

Show a specific month and year

```
cal 12 2026
```

Output:

```
    December 2026
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
...
```

Show nearby months

```
cal -3
```

Displays:

- previous month
- current month
- next month

When you use `cal`

It is useful for:

- quickly checking dates
- finding the day of the week for a date
- planning tasks from the command line
- working on servers without a graphical calendar

Difference between `date` and `cal`

Command	Purpose
`Date`	Shows the exact current date and time
`cal`	Shows a monthly/yearly calendar

Example:

```
date
```

tells you:

> "What is the current date and time?"

```
cal
```

tells you:

> "What does this month's calendar look like?"

6. who

`who` is a **Linux/Unix command** used to display information about **users currently logged into the system**.

Basic usage

```
who
```

Example output

```
john     pts/0        2026-07-25 10:15 (192.168.1.50)
alice    tty1         2026-07-25 09:30
```

Breaking down the output

Example:

```
john     pts/0        2026-07-25 10:15 (192.168.1.50)
```

Field	Meaning
`john`	Username of the logged-in user
`pts/0`	Terminal session being used
`2026-07-25 10:15`	Login date and time
`(192.168.1.50)`	Remote machine IP address (if connected remotely)

Understanding terminal types

`tty`

Example:

```
john tty1
```

- A physical terminal session.
- Usually means the user is logged in directly on the machine.

`pts`

Example:

```
john pts/0
```

- A pseudo-terminal session.
- Commonly created when using:
    - SSH
    - Terminal applications
    - Remote connections

Common options

Command	Description
`who`	Shows logged-in users
`who -H`	Shows column headings
`who -b`	Shows the last system boot time
`who -q`	Shows usernames and total number of users
`who -r`	Shows the current run level

Examples

Show headings

```
who -H
```

Output:

```
NAME     LINE     TIME
john     pts/0    2026-07-25 10:15
```

Show system boot time

```
who -b
```

Output:

```
system boot  2026-07-20 07:15
```

This tells you when the computer was last started.

Show number of users

```
who -q
```

Output:

```
john alice
# users=2
```

Difference between `who`, `w`, and `users`

Command	Purpose
`who`	Shows who is logged in and login details
`users`	Shows only usernames currently logged in
`w`	Shows users plus what they are doing and system load

Example:

```
who
```

answers:

"Who is logged into this system?"

```
w
```

answers:

"Who is logged in, and what are they doing?"

When you use `who`

It is useful for:

- checking active users on a server
- monitoring remote logins
- finding login times
- checking who is using a shared system
- basic system administration tasks

In a multi-user Linux server environment, `who` helps administrators know **which users are currently connected and from where**.

7. w

`w` is a **Linux/Unix command** used to display **who is logged into the system and what they are doing**. It provides more detailed information than the `who` command by showing user activity and system load.

Basic usage

```
w
```

Example output

```
 14:35:20 up 5 days, 3:20, 2 users, load average: 0.15, 0.10, 0.05
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
john     pts/0    192.168.1.50     10:15    5:20   0.10s  0.05s bash
alice    tty1                      09:30    2:00   0.05s  0.01s vim file.txt
```

Breaking down the output

First line (system information)

```
14:35:20 up 5 days, 3:20, 2 users, load average: 0.15, 0.10, 0.05
```

Part	Meaning
`14:35:20`	Current system time
`up 5 days, 3:20`	How long the system has been running
`2 users`	Number of logged-in users
`load average`	System workload for the last 1, 5, and 15 minutes

User information columns

Example:

```
john pts/0 192.168.1.50 10:15 5:20 0.10s 0.05s bash
```

Column	Meaning
USER	Username of the logged-in user
TTY	Terminal session being used
FROM	Remote host or IP address
LOGIN@	Login time
IDLE	Time since the user last interacted with the terminal
JCPU	CPU time used by all processes attached to that terminal
PCPU	CPU time used by the current process
WHAT	Current command/process being run

Understanding important fields

TTY

Shows the terminal type:

Example:

```
pts/0
```

- A pseudo-terminal, usually from SSH or a terminal window.

Example:

```
tty1
```

- A physical console terminal.

IDLE

Shows how long the user has been inactive.

Example:

```
IDLE
5:20
```

Means the user has not typed anything for 5 minutes and 20 seconds.

WHAT

Shows what the user is currently doing.

Examples:

`bash`

- User is running a shell.

`vim file.txt`

- User is editing a file.

`ssh server1`

- User is connected to another machine.

Common options

Command	Description
`w`	Shows logged-in users and activity
`w -h`	Hides the header line
`w -s`	Shows a shorter format
`w username`	Shows information for a specific user

Example: Check one user

```
w john
```

Output:

```
john pts/0 192.168.1.50 10:15 bash
```

Difference between `who`, `w`, and `users`

Command	Information shown
`users`	Only usernames
`who`	Users, login time, and terminal
`w`	Users, login details, system load, and current activity

Example:

```
who
```

Answers:

"Who is logged in?"

```
w
```

Answers:

"Who is logged in, how long have they been idle, and what are they doing?"

When you use `w`

It is useful for:

- monitoring users on a server
- checking active sessions
- finding inactive terminals
- troubleshooting high system load
- seeing what processes users are running

System administrators often use `w` as a quick overview of **current users and system activity**.

9. last

`last` is a **Linux/Unix command** used to display a **history of user logins and system reboots**. It reads login records from system log files and shows who logged in, when they logged in, and how long they stayed connected.

Basic usage

```
last
```

Example output

```
john     pts/0        192.168.1.50     Sat Jul 25 10:15   still logged in
alice    tty1                          Fri Jul 24 09:30 - 17:45  (08:15)
reboot   system boot  6.8.0-31         Fri Jul 24 07:20   still running
```

Breaking down the output

Example:

```
john pts/0 192.168.1.50 Sat Jul 25 10:15 still logged in
```

Field	Meaning
`john`	Username of the person who logged in
`pts/0`	Terminal session used
`192.168.1.50`	Remote IP address (if login was remote)
`Sat Jul 25 10:15`	Login date and time
`still logged in`	User is currently connected

Understanding special entries

`reboot`

Example:

```
reboot   system boot  6.8.0-31  Fri Jul 24 07:20
```

- Shows when the system was started.
- Helps determine system uptime and reboot history.

`shutdown`

Example:

```
shutdown system down Fri Jul 24 18:00
```

- Shows when the system was shut down.

Common options

Command	Description
`last`	Shows all login history
`last -n 5`	Shows the last 5 login records
`last username`	Shows login history for a specific user
`last reboot`	Shows system reboot history
`last shutdown`	Shows shutdown history
`last -a`	Shows hostname/IP at the end of each entry

Examples

Show only recent logins

```
last -n 5
```

Example output:

```
john   pts/0   192.168.1.50   Sat Jul 25 10:15
alice  tty1                  Fri Jul 24 09:30
```

Check a specific user's login history

```
last john
```

Output:

```
john pts/0 192.168.1.50 Sat Jul 25 10:15 - 12:30
john pts/1 192.168.1.20 Thu Jul 23 14:00 - 16:00
```

This shows when the user `john` logged in and logged out.

Check reboot history

```
last reboot
```

Example:

```
reboot system boot 6.8.0-31 Fri Jul 24 07:20
reboot system boot 6.8.0-31 Mon Jul 20 08:10
```

Useful for checking how often a machine has restarted.

Where does `last` get its information?

`last` reads from:

```
/var/log/wtmp
```

This file stores historical login/logout records.

Related commands:

Command	Log file	Purpose
`Last`	`/var/log/wtmp`	Previous login history
`lastb`	`/var/log/btmp`	Failed login attempts
`who`	`/var/run/utmp`	Currently logged-in users

Difference between `last`, `who`, and `w`

Command	Shows
`who`	Users currently logged in
`w`	Current users + what they are doing
`last`	Previous login history

Example:

`who`

answers:

"Who is logged in right now?"

`w`

answers:

"Who is logged in and what are they doing?"

`last`

answers:

"Who logged in previously and when?"

When you use `last`

It is useful for:

- auditing user logins
- checking server access history
- investigating unexpected logins
- finding reboot times
- monitoring shared systems or servers

System administrators commonly use `last` to review **who accessed a system and when**.

9. lscpu

`lscpu` is a **Linux command** used to display detailed information about the **CPU (Central Processing Unit)** architecture and configuration of a system.

It is part of the **util-linux** package and is commonly used by system administrators to inspect processor details.

Basic usage

```
lscpu
```

Example output

```
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              8
On-line CPU(s) list: 0-7
Thread(s) per core:  2
Core(s) per socket:  4
Socket(s):           1
Model name:          Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
CPU MHz:             1800.000
Virtualization:      VT-x
L1d cache:           128 KiB
L2 cache:            1 MiB
L3 cache:            6 MiB
```

Breaking down the output

1. CPU Architecture

```
Architecture: x86_64
```

- Shows the CPU architecture.
- Common values:
    - `x86_64` → 64-bit Intel/AMD processors
    - `aarch64` → 64-bit ARM processors

2. CPU operating modes

```
CPU op-mode(s): 32-bit, 64-bit
```

Shows the modes supported by the processor.

- **32-bit** → Can run 32-bit applications
- **64-bit** → Can run modern 64-bit operating systems and applications

---

3. Number of CPUs

```
CPU(s): 8
```

Shows the total number of logical CPUs available.

Example:

- 4 physical cores
- 2 threads per core

Calculation:

```
4 cores × 2 threads = 8 logical CPUs
```

---

4. Threads per core

```
Thread(s) per core: 2
```

Shows whether the CPU supports simultaneous multithreading (SMT), such as:

- Intel Hyper-Threading
- AMD SMT

Example:

```
1 core → 2 threads
```

5. Cores per socket

```
Core(s) per socket: 4
```

Shows the number of physical CPU cores in each processor package.

---

6. Sockets

```
Socket(s): 1
```

Shows the number of physical CPU packages installed.

Example:

- Desktop/laptop: usually `1`
- Servers may have multiple sockets.

---

7. CPU model name

```
Model name: Intel(R) Core(TM) i5-8250U
```

Shows the exact processor model.

---

8. CPU frequency

```
CPU MHz: 1800.000
```

Shows the current CPU operating speed.

- Measured in MHz (megahertz)
- `1000 MHz = 1 GHz`

9. Virtualization support

```
Virtualization: VT-x
```

Shows hardware virtualization support.

Examples:

- Intel → VT-x
- AMD → AMD-V

Used for running virtual machines.

---

10. Cache information

Example:

```
L1d cache: 128 KiB
L2 cache: 1 MiB
L3 cache: 6 MiB
```

CPU cache is high-speed memory close to the processor.

Levels:

- **L1** → Fastest, smallest
- **L2** → Larger, slightly slower
- **L3** → Shared cache, larger capacity

Useful options

Command	Description
`lscpu`	Displays all CPU information
`lscpu -e`	Shows CPU information in a table format
`lscpu -p`	Shows CPU information in a parseable format

Example: Display CPU list

```
lscpu -e
```

Output:

```
CPU NODE SOCKET CORE MHz
0   0    0      0    1800
1   0    0      0    1800
2   0    0      1    1800
3   0    0      1    1800
```

Difference between `lscpu` and `uname`

Command	Shows
`uname -m`	CPU architecture only
`lscpu`	Complete CPU details

Example:

```
uname -m
```

Output:

```
x86_64
```

Only tells:

"This system uses a 64-bit architecture."

```
lscpu
```

Shows:

"What CPU is installed, how many cores it has, its speed, cache, and features."

When you use `lscpu`

It is useful for:

- checking CPU specifications
- troubleshooting performance issues
- verifying hardware compatibility
- configuring virtual machines
- system monitoring and administration

A quick summary:

`lscpu` = "Show me detailed information about my processor."

10. lsmem

`lsmem` is a Linux command used to display information about the system's memory (RAM) layout. It shows details about available memory blocks, their size, state, and whether they are online or offline.

It is part of the **util-linux** package and is mainly used by system administrators.

Basic usage

```
lsmem
```

Example output

```
RANGE                                  SIZE  STATE REMOVABLE BLOCK
0x0000000000000000-0x000000007fffffff  2G    online       yes      0
0x0000000080000000-0x00000000ffffffff  2G    online       yes      1

Memory block size:       128M
Total online memory:       4G
Total offline memory:      0B
```

Breaking down the output

1. RANGE

```
RANGE
0x0000000000000000-0x000000007fffffff
```

- Shows the physical memory address range.
- Represents where a memory block exists in the system's physical memory map.

2. SIZE

```
SIZE
2G
```

- Shows the size of that memory block.
- Common units:
  - `K` → Kilobytes
  - `M` → Megabytes
  - `G` → Gigabytes

---

3. STATE

```
STATE
online
```

Shows whether the memory block is currently available.

Possible values:

State	Meaning
`online`	Memory is active and usable
`offline`	Memory is disabled and not usable

---

4. REMOVABLE

```
REMOVABLE
yes
```

Shows whether the memory block can be removed or hot-unplugged.

- `yes` → Can potentially be removed while the system is running
- `no` → Cannot be removed

This is mainly relevant on servers with hot-pluggable memory.

5. BLOCK

```
BLOCK
0
```

Shows the memory block number.

Linux divides physical memory into smaller units called **memory blocks** for management purposes.

Summary information

At the bottom, `lsmem` may show:

```
Memory block size: 128M
```

- The size of each memory management block.

```
Total online memory: 4G
```

- Total RAM currently available to the operating system.

```
Total offline memory: 0B
```

- RAM that exists but is currently disabled.

Common options

Command	Description
`lsmem`	Displays memory block information
`lsmem -a`	Shows all memory ranges, including empty ones
`lsmem -s`	Displays a summary only
`lsmem -o +RANGE`	Adds specific output columns

Example: Show only summary

```
lsmem -s
```

Output:

```
Memory block size:       128M
Total online memory:       8G
Total offline memory:      0B
```

Difference between `lsmem` and `free`

Command	Shows
`lsmem`	Physical memory layout and memory blocks
`free`	Current RAM usage (used, free, cached memory)

Example:

```
free -h
```

Shows:

How much RAM is currently being used.

```
lsmem
```

Shows:

How RAM is arranged and managed by the kernel.

Difference between `lsmem` and `lscpu`

Command	Purpose
`lscpu`	Displays CPU information
`lsmem`	Displays memory information

Example:

`lscpu`

answers:

"What processor does this system have?"

`lsmem`

answers:

"How is the system's RAM organised?"

When you use `lsmem`

It is useful for:

- checking RAM configuration
- troubleshooting memory issues
- managing large servers
- checking memory hot-plug support
- understanding how Linux sees physical memory

In short:

`lsmem` = "Show me how the system's physical RAM is organised."

11. lsblk

`lsblk` is a Linux command used to display information about block devices connected to a system, such as hard drives, SSDs, USB drives, and their partitions.

The name stands for:

> **lsblk = list block devices**

A **block device** is a storage device that reads and writes data in fixed-size blocks.

Basic usage

```
lsblk
```

Example output

```
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda           8:0    0   500G  0 disk
├─sda1        8:1    0   100G  0 part /
├─sda2        8:2    0   300G  0 part /home
└─sda3        8:3    0   100G  0 part [SWAP]
sdb           8:16   1    32G  0 disk
└─sdb1        8:17   1    32G  0 part /media/usb
```

Breaking down the output

1. NAME

```
NAME
sda
```

Shows the device name.

Common names:

Name	Meaning
`sda`	First hard disk/SSD
`sdb`	Second disk or USB drive
`nvme0n1`	NVMe SSD
`sda1`	First partition of `sda`

Example:

```
sda
├── sda1
├── sda2
└── sda3
```

Means:

- `sda` → physical disk
- `sda1`, `sda2`, `sda3` → partitions inside that disk

---

2. MAJ:MIN

```
MAJ:MIN
8:0
```

Shows the Linux device numbers.

- **Major number** → identifies the device driver
- **Minor number** → identifies a specific device or partition

Usually system administrators do not need to interact with these directly.

3. RM (Removable)

```
RM
1
```

Shows whether the device is removable.

Value	Meaning
`0`	Non-removable
`1`	Removable

Example:

```
sdb   1
```

Could be a USB flash drive.

---

4. SIZE

```
SIZE
500G
```

Shows the storage capacity.

Examples:

```
500G
```

→ 500 gigabytes

```
32G
```

→ 32 gigabytes

5. RO (Read Only)

```
RO
0
```

Shows whether the device is read-only.

Value	Meaning
`0`	Read/write
`1`	Read-only

---

6. TYPE

```
TYPE
disk
```

Shows the type of device.

Common types:

Type	Meaning
`disk`	Physical storage device
`part`	Partition
`rom`	Read-only media
`lvm`	Logical volume

Example:

```
sda      disk
├─sda1   part
└─sda2   part
```

Means:

- `sda` is the physical disk
- `sda1` and `sda2` are partitions

7. MOUNTPOINTS

```
MOUNTPOINTS
/
```

Shows where a partition is attached in the Linux filesystem.

Examples:

Mount point	Purpose
`/`	Root filesystem
`/home`	User files
`/boot`	Boot files
`/media/usb`	USB storage

Example:

```
sda1   /
```

Means the first partition contains the main Linux filesystem.

Useful options

1. Show filesystem information

```
lsblk -f
```

Example:

```
NAME   FSTYPE LABEL UUID                                 MOUNTPOINT
sda
├─sda1 ext4         1234-5678                            /
└─sda2 swap         9876-5432                            [SWAP]
```

Shows:

- filesystem type (`ext4`, `xfs`, `ntfs`)
- labels
- UUIDs
- mount points

2. Show detailed information

```
lsblk -a
```

Displays all available block devices, including empty devices.

---

3. Show device sizes in a readable format

```
lsblk -h
```

Displays sizes in a human-friendly format.

---

4. Display only specific columns

```
lsblk -o NAME,SIZE,TYPE,MOUNTPOINT
```

Example:

```
NAME   SIZE TYPE MOUNTPOINT
sda    500G disk
sda1   100G part /
sda2   300G part /home
```

Difference between `lsblk`, `df`, and `du`

Command	Purpose
`lsblk`	Shows disks and partitions
`df`	Shows filesystem space usage
`du`	Shows size of files/directories

Example:

```
lsblk
```

Answers:

"What storage devices and partitions exist?"

```
df -h
```

Answers:

"How much free space is left?"

```
du -sh folder/
```

Answers:

"How much space does this folder use?"

Difference between `lsblk` and `lscpu`

Command	Shows
`lscpu`	CPU information
`lsmem`	RAM information
`lsblk`	Storage device information

When you use `lsblk`

It is useful for:

- identifying hard drives and USB devices
- checking disk partitions
- finding where storage is mounted
- preparing disks for installation
- troubleshooting storage problems

In short:

`lsblk` = "Show me all storage devices and their partitions connected to this Linux system."

12. lspci

`lspci` is a Linux command used to display information about PCI (Peripheral Component Interconnect) devices connected to a system.

The name stands for:

lspci = list PCI devices

PCI devices are hardware components connected to the motherboard through the PCI bus, such as:

- Graphics cards (GPU)
- Network cards (Ethernet/Wi-Fi adapters)
- Audio controllers
- Storage controllers
- USB controllers

Basic usage

```
lspci
```

Example output

```
00:00.0 Host bridge: Intel Corporation 8th Gen Core Processor Host Bridge
00:02.0 VGA compatible controller: Intel Corporation UHD Graphics 620
00:14.0 USB controller: Intel Corporation USB 3.0 Controller
00:1f.3 Audio device: Intel Corporation Audio Controller
01:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111
```

Breaking down the output

Example:

```
01:00.0 Ethernet controller: Realtek Semiconductor Co., Ltd. RTL8111
```

1. PCI Address

```
01:00.0
```

This is the PCI device address.

It is divided into:

```
Bus:Device.Function
```

Example:

```
01:00.0
│  │  │
│  │  └── Function number
│  └───── Device number
└──────── Bus number
```

Linux uses this address to identify each PCI device.

---

2. Device type

```
Ethernet controller
```

Shows what kind of hardware it is.

Common types:

Type	Device
`VGA compatible controller`	Graphics card
`Ethernet controller`	Wired network adapter
`Network controller`	Wi-Fi adapter
`Audio device`	Sound card
`USB controller`	USB hardware
`SATA controller`	Storage controller

3. Manufacturer and model

```
Realtek Semiconductor Co., Ltd. RTL8111
```

Shows:

- hardware manufacturer
- device model

Examples:

- Intel
- AMD
- NVIDIA
- Realtek
- Broadcom

Common options

1. Show detailed information

```
lspci -v
```

Displays more details about each PCI device.

Example:

```
01:00.0 VGA compatible controller: NVIDIA Corporation RTX 3060
        Kernel driver in use: nvidia
        Memory at d0000000
        Capabilities: <access denied>
```

Shows:

- kernel driver
- memory addresses
- device capabilities

2. Show very detailed information

```
lspci -vv
```

Provides even more hardware details.

Useful for troubleshooting.

---

3. Show kernel drivers

```
lspci -k
```

Example:

```
02:00.0 Network controller: Intel Corporation Wi-Fi 6 AX200
        Kernel driver in use: iwlwifi
        Kernel modules: iwlwifi
```

Shows:

- currently loaded driver
- available kernel modules

---

4. Show PCI device IDs

```
lspci -nn
```

Example:

```
03:00.0 VGA compatible controller: NVIDIA Corporation RTX 3060 [10de:2504]
```

The numbers:

```
[10de:2504]
```

are:

- Vendor ID
- Device ID

They help identify hardware precisely.

5. Display devices in tree format

```
lspci -t
```

Example:

```
-[0000:00]-+-00.0
           +-02.0
           +-14.0
           \-1f.3
```

Shows how PCI devices are connected.

Difference between `lspci` and other hardware commands

Command	Purpose
`lscpu`	Shows CPU information
`lsmem`	Shows RAM information
`lsblk`	Shows storage devices
`lspci`	Shows PCI-connected hardware

Example:

```
lscpu
```

answers:

"What processor do I have?"

```
lsblk
```

answers:

"What disks and partitions are connected?"

```
lspci
```

answers:

"What expansion hardware is installed?"

When you use `lspci`

It is useful for:

- identifying installed hardware
- checking graphics card information
- troubleshooting network adapters
- finding missing drivers
- verifying PCI hardware on servers

For example, if your Wi-Fi is not working, you can run:

```
lspci | grep -i network
```

to find the wireless network device.

In short:

`lspci` = "Show me all PCI hardware devices connected to this Linux system."

13. lsusb

`lsusb` is a Linux command used to display information about USB (Universal Serial Bus) devices connected to a system.

The name stands for:

lsusb = list USB devices

It helps you identify connected USB hardware such as:

- USB keyboards and mice
- Flash drives
- Webcams
- Printers
- Wi-Fi adapters
- External hard drives
- Mobile phones

Basic usage

```
lsusb
```

Example output

```
Bus 002 Device 003: ID 046d:c52b Logitech, Inc. USB Receiver
Bus 002 Device 004: ID 0781:5567 SanDisk Corp. Cruzer Blade
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
```

Breaking down the output

Example:

```
Bus 002 Device 003: ID 046d:c52b Logitech, Inc. USB Receiver
```

1. Bus number

```
Bus 002
```

- Identifies the USB bus where the device is connected.
- A system can have multiple USB buses.

---

2. Device number

```
Device 003
```

- The device number assigned by the Linux kernel.
- It identifies the device on that USB bus.

---

3. Vendor ID and Product ID

```
ID 046d:c52b
```

This contains two identifiers:

```
046d : c52b
  │      │
  │      └── Product ID
  └─────────── Vendor ID
```

- **Vendor ID** identifies the manufacturer.
- **Product ID** identifies the specific device model.

Example:

```
046d
```

→ Logitech

```
c52b
```

→ A specific Logitech USB receiver model

4. Device description

```
Logitech, Inc. USB Receiver
```

Shows:

- Manufacturer name
- Device type

Common options

1. Show detailed information

```
lsusb -v
```

Displays detailed information about USB devices.

Example:

```
Device Descriptor:
  idVendor           0x046d Logitech
  idProduct          0xc52b USB Receiver
  bcdUSB             2.00
```

Shows:

- vendor ID
- product ID
- USB version
- device capabilities

---

2. Show device tree

```
lsusb -t
```

Example:

```
/:  Bus 02.Port 1: Dev 1, Class=root_hub
    |__ Port 2: Dev 3, Class=Human Interface Device
    |__ Port 3: Dev 4, Class=Mass Storage
```

Shows how USB devices are connected.

3. Show USB device IDs

```
lsusb -nn
```

Example:

```
ID 046d:c52b Logitech USB Receiver
```

Displays vendor and product IDs in numeric form.

---

4. Filter a specific device

You can combine `lsusb` with `grep`.

Example:

```
lsusb | grep -i camera
```

Finds USB cameras.

Example:

```
lsusb | grep -i wireless
```

Finds USB wireless adapters.

Difference between `lsusb` and `lspci`

Command	Shows
`lsusb`	USB-connected devices
`lspci`	PCI-connected devices

Examples:

```
lsusb
```

Answers:

"What USB devices are connected?"

```
lspci
```

Answers:

"What internal PCI hardware is installed?"

Difference between `lsusb` and `lsblk`

Command	Purpose
`lsusb`	Lists USB hardware devices
`lsblk`	Lists storage devices and partitions

Example:

A USB flash drive may appear in both:

```
lsusb
```

Shows:

The USB controller sees a SanDisk device.

```
lsblk
```

Shows:

The SanDisk device has a partition and storage size.

When you use `lsusb`

It is useful for:

- checking whether a USB device is detected
- troubleshooting USB problems
- finding device IDs for drivers
- identifying unknown USB hardware
- checking connected peripherals

Example troubleshooting:

If a USB Wi-Fi adapter is not working:

```
lsusb
```

Check whether the adapter appears.

If it appears:

```
ID 0bda:c811 Realtek Semiconductor Corp. Wireless Adapter
```

you can use the vendor and product ID to find the correct driver.

In short:

> **`lsusb` = "Show me all USB devices connected to this Linux system."**