Table of Contents

$ lshw

Intermediate

List detailed hardware configuration

lshw (list hardware) displays detailed information about the complete hardware configuration of the machine. It reports on memory, CPU, disks, network interfaces, graphics, and all other hardware components.

lshw reads from /proc, /sys, DMI tables, and other sources to build a comprehensive hard...

Options & Flags

`-short`	Show brief hardware summary
`-class`	Show specific hardware class
`-html`	Output as HTML
`-json`	Output as JSON
`-sanitize`	Remove sensitive info (serial numbers)
`-businfo`	Show bus information

Practical Examples

Example: Brief summary

```
$ sudo lshw -short
H/W path     Class      Description
/0/0         memory     16GiB System Memory
/0/1         processor  Intel Core i7
/0/100/1f.2  storage    SATA Controller
```

Shows a concise hardware inventory.

Example: Memory details

```
$ sudo lshw -class memory
```

Shows detailed memory configuration including slots and modules.

Example: Network hardware

```
$ sudo lshw -class network
```

Shows network interface details including driver, speed, and MAC address.

Example: Disk information

```
$ sudo lshw -class disk
```

Shows disk hardware details including model, serial, and capacity.

Example: HTML report

```
$ sudo lshw -html > /tmp/hardware.html
```

Creates a comprehensive HTML hardware report.

Tips & Best Practices

Pro Tip: Use -short for quick overview: sudo lshw -short gives a one-line-per-device summary. Much easier to scan than the full output.

Warning: Requires root: Run with sudo for complete information. Without root, many details are unavailable or shown as Unknown.

Note: Alternative tools: lscpu for CPU, lsblk for disks, lspci for PCI devices, lsusb for USB. lshw shows everything but specialized tools have more detail.

$ lsmod

Beginner

List loaded kernel modules

The lsmod command displays the status of currently loaded Linux kernel modules. Kernel modules are pieces of code that can be loaded into the kernel on demand, extending its functionality without rebooting. They handle hardware drivers, filesystems, network protocols, and various kernel features....

Options & Flags

`(no options)`	lsmod takes no arguments - lists all loaded modules
`| grep MODULE`	Filter output for a specific module
`| head -20`	Show first 20 loaded modules
`| wc -l`	Count total loaded modules
`| sort -k2 -n -r`	Sort by size (largest first)

Practical Examples

Example: List all loaded modules

```
$ lsmod
Module                  Size  Used by\nnvidia               2265088  48\nnvidia_modeset        1142784  1\next4                  8
```

Shows all currently loaded kernel modules with size and dependency information.

Example: Check if WireGuard is loaded

```
$ lsmod | grep wireguard
wireguard  90112  0
```

Verify the WireGuard kernel module is loaded. Empty output means it is not loaded.

Example: Find network driver modules

```
$ lsmod | grep -E "e1000|igb|ixgbe|mlx|r8169|virtio_net"
```

Check which network driver module is in use. Helps identify network card driver issues.

Example: Count loaded modules

```
$ echo "Loaded modules: $(lsmod | tail -n +2 | wc -l)"
Loaded modules: 142
```

Count the total number of loaded kernel modules (excluding header line).

Example: Show largest modules

```
$ lsmod | sort -k2 -n -r | head -10
```

List the 10 largest kernel modules by memory usage.

Tips & Best Practices

Note: lsmod reads /proc/modules: lsmod simply formats the output of /proc/modules. You can also read this file directly: cat /proc/modules

Pro Tip: Use modinfo for module details: For detailed info about a module: modinfo MODULE_NAME - shows description, author, license, parameters, dependencies, and file path.

Note: Used by column: The "Used by" count shows how many other modules depend on it. A module with "Used by 0" can be safely removed with modprobe -r.

$ lspci

Intermediate

List all PCI devices

lspci lists all PCI devices in the system including graphics cards, network adapters, storage controllers, USB controllers, and sound cards. PCI is the main bus connecting hardware components.

lspci is essential for identifying hardware, finding correct drivers, and troubleshooting device detect...

Options & Flags

-v	Verbose - show detailed info
-vv	Very verbose
-nn	Show vendor/device IDs
-k	Show kernel driver in use
-s	Show specific device by slot
-d	Show devices by vendor:device ID

Practical Examples

Example: List all PCI devices

```
$ lspci
00:00.0 Host bridge: Intel Corporation
00:02.0 VGA compatible controller: NVIDIA
00:1f.2 SATA controller: Intel
```

Shows all PCI devices with one line each.

Example: Show with drivers

```
$ lspci -k
```

Shows which kernel driver is loaded for each device.

Example: Show with IDs

```
$ lspci -nn
00:02.0 VGA [0300]: NVIDIA [10de:2504]
```

Shows vendor:device IDs in brackets - useful for finding drivers online.

Example: Find GPU

```
$ lspci | grep -i vga
00:02.0 VGA compatible controller: NVIDIA GeForce RTX 3080
```

Shows graphics card(s) in the system.

Example: Find network card

```
$ lspci | grep -i net
```

Shows network interface hardware.

Tips & Best Practices

Pro Tip: Find driver for a device: lspci -nn shows vendor:device IDs in brackets. Search these online to find the correct Linux driver.

Note: -k shows loaded driver: lspci -k shows "Kernel driver in use:" for each device. If missing, the driver is not loaded.

Warning: Update PCI database: If devices show as "Unknown", update the PCI ID database: sudo update-pciids.

$ lsusb

Intermediate

List USB devices

lsusb lists all USB devices connected to the system. It shows the bus number, device number, vendor ID, product ID, and device description for each USB device.

lsusb is useful for identifying connected USB devices, troubleshooting USB connections, finding vendor:product IDs for driver configurat...

Options & Flags

`-v`	Verbose - show detailed device info
`-t`	Show device tree hierarchy
`-s`	Show specific bus:device
`-d`	Show devices by vendor:product ID

Practical Examples

Example: List USB devices

```
$ lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 046d:c52b Logitech, Inc. Wireless Mouse
```

Shows all connected USB devices.

Example: Device tree

```
$ lsusb -t
/:  Bus 01.Port 1: Dev 1, Class=root_hub, Driver=xhci_hcd, 480M
```

Shows USB device hierarchy with speed information.

Example: Verbose device info

```
$ lsusb -v -s 001:003
```

Shows detailed information about a specific USB device.

Example: Find by vendor

```
$ lsusb -d 046d:
```

Shows all Logitech devices (vendor ID 046d).

Example: Check USB storage

```
$ lsusb | grep -i storage
```

Finds USB mass storage devices (flash drives, external disks).

Tips & Best Practices

Pro Tip: Check dmesg after plugging in: After connecting a USB device, run dmesg -T | tail to see kernel detection messages and any errors.

Note: Vendor:product IDs: The hex IDs (e.g., 046d:c52b) are useful for configuring udev rules and finding drivers online.

Warning: USB database updates: If devices show as "Unknown", update the USB ID database: sudo update-usbids.

$ modprobe

Intermediate

Load and remove kernel modules with dependency handling

The modprobe command intelligently loads and removes Linux kernel modules along with their dependencies. Unlike insmod which loads a single module without dependency resolution, modprobe reads module dependency information from /lib/modules/$(uname -r)/modules.dep and automatically loads all requ...

Options & Flags

`MODULE`	Load a kernel module and its dependencies
`-r MODULE`	Remove (unload) a module and unused dependencies
`-n`	Dry run - show what would be done without doing it
`-v`	Verbose - show each step being performed
`--show-depends`	Show module dependencies without loading
`-c`	Show current modprobe configuration
`MODULE param=value`	Load module with specific parameters
`-b`	Apply blacklist directives (used by udev)

Practical Examples

Example: Load a kernel module

```
$ sudo modprobe wireguard && lsmod | grep wireguard
wireguard  90112  0
```

Load the WireGuard module and verify it is loaded. modprobe automatically resolves and loads dependencies.

Example: Remove a kernel module

```
$ sudo modprobe -r wireguard
```

Unload WireGuard and any dependencies no longer in use. Fails if the module is currently in use.

Example: Load module at boot

```
$ echo "wireguard" | sudo tee /etc/modules-load.d/wireguard.conf
```

Configure a module to load automatically at every boot by creating a .conf file in modules-load.d.

Example: Blacklist a module

```
$ echo "blacklist nouveau" | sudo tee /etc/modprobe.d/blacklist-nouveau.conf && sudo update-initramfs -u
```

Prevent a module from loading automatically. Common for replacing open-source GPU drivers with proprietary ones.

Example: Load bonding with parameters

```
$ sudo modprobe bonding mode=802.3ad miimon=100 lacp_rate=fast
```

Load the bonding module with LACP (802.3ad) mode for link aggregation with 100ms monitoring.

Tips & Best Practices

Warning: Rebuild initramfs after blacklist: After blacklisting a module, rebuild initramfs: update-initramfs -u (Debian/Ubuntu) or dracut -f (RHEL/Fedora). Otherwise the module may still load from initramfs.

Pro Tip: Use modinfo for parameters: Before loading with parameters, check available options: modinfo -p MODULE_NAME lists all module parameters with descriptions.

Note: modprobe vs insmod: Always use modprobe instead of insmod. modprobe handles dependencies automatically and respects blacklists. insmod is a low-level tool that loads a single .ko file.

Pro Tip: Debug module loading: If a module fails to load, check dmesg for kernel messages: sudo modprobe module && dmesg | tail -20

$ nproc

Beginner

Print the number of processing units available

nproc prints the number of available processing units (CPU cores/threads). It is the simplest way to determine how many parallel tasks the system can handle.

nproc is commonly used with make -j for parallel compilation and with xargs -P for parallel command execution. It respects cgroup limits i...

Options & Flags

`(no args)`	Show available processors
`--all`	Show all installed processors

Practical Examples

Example: Get CPU count

```
$ nproc
8
```

Shows the number of available processing units.

Example: Parallel make

```
$ make -j$(nproc)
```

Compiles using all available CPU cores.

Example: Parallel xargs

```
$ find . -name '*.png' | xargs -P $(nproc) -I {} convert {} -resize 50% {}
```

Processes images in parallel using all cores.

Example: Leave one core free

```
$ make -j$(($(nproc) - 1))
```

Compiles using all cores except one, keeping system responsive.

Example: All vs available

```
$ echo "Available: $(nproc), Total: $(nproc --all)"
Available: 4, Total: 8
```

Shows available vs total processors (may differ in containers).

Tips & Best Practices

Pro Tip: Use with make -j: make -j$(nproc) is the standard way to parallelize compilation. It uses all available cores automatically.

Note: Container awareness: nproc respects cgroup CPU limits in containers. If a container is limited to 2 cores, nproc returns 2.

Warning: Leave cores for the system: For long-running background tasks, use nproc - 1: make -j$(($(nproc) - 1)) to keep the system responsive.

$ uname

Beginner

Print system information (kernel, OS, architecture)

uname displays system information including the kernel name, version, architecture, hostname, and operating system. It is the standard way to identify the running system in scripts and documentation.

uname -a shows all available information in one line. Individual flags extract specific pieces: ...

Options & Flags

`-a`	Print all system information
`-s`	Kernel name
`-r`	Kernel release version
`-m`	Machine hardware architecture
`-n`	Network hostname
`-o`	Operating system
`-v`	Kernel version (build info)

Practical Examples

Example: Show all info

```
$ uname -a
Linux hostname 5.15.0-91-generic #101-Ubuntu SMP x86_64 GNU/Linux
```

Displays complete system identification.

Example: Check architecture

```
$ uname -m
x86_64
```

Shows hardware architecture (x86_64, aarch64, armv7l, etc.).

Example: Check kernel version

```
$ uname -r
5.15.0-91-generic
```

Shows the running kernel version.

Example: Check OS

```
$ uname -o
GNU/Linux
```

Shows the operating system name.

Example: Conditional in script

```
$ [[ $(uname -m) == "x86_64" ]] && echo "64-bit" || echo "Other"
64-bit
```

Architecture check for conditional installation.

Tips & Best Practices

Pro Tip: Check architecture for downloads: Use uname -m before downloading binaries. x86_64 = 64-bit Intel/AMD, aarch64 = 64-bit ARM, armv7l = 32-bit ARM.

Note: More detailed info: For OS distribution details, use cat /etc/os-release or lsb_release -a instead of uname.

Warning: Container kernel: In containers, uname shows the HOST kernel, not the container OS. Use /etc/os-release for container OS info.

$ uptime

Beginner

Show how long the system has been running

uptime shows how long the system has been running, the current time, number of logged-in users, and system load averages for the last 1, 5, and 15 minutes.

uptime provides a quick health check of any system. High load averages relative to CPU count indicate the system is overloaded. Load average...

Options & Flags

`-p`	Show uptime in human-readable format
`-s`	Show system boot time

Practical Examples

Example: Standard uptime

```
$ uptime
 14:30:00 up 45 days, 3:22, 2 users, load average: 0.15, 0.10, 0.05
```

Shows time, uptime, users, and load averages.

Example: Pretty format

```
$ uptime -p
up 45 days, 3 hours, 22 minutes
```

Shows uptime in human-readable format.

Example: Boot time

```
$ uptime -s
2024-11-15 11:08:00
```

Shows when the system was started.

Example: Check load averages

```
$ uptime | awk -F"load average:" '{print $2}'
 0.15, 0.10, 0.05
```

Extracts just the load averages.

Example: Script monitoring

```
$ echo "$(hostname): $(uptime -p), Load: $(cut -d" " -f1 /proc/loadavg)"
```

Creates a monitoring summary line.

Tips & Best Practices

Pro Tip: Load average interpretation: Load average should be compared to CPU count. On a 4-core system: <4 is healthy, 4 is fully loaded, >4 is overloaded. Check cores: nproc.

Note: Three averages: 1-minute average shows current activity. 5-minute shows recent trend. 15-minute shows longer trend. Rising 1-min with low 15-min means a new spike.

Warning: Load includes I/O wait: Load average includes processes waiting for I/O (disk), not just CPU. High load with low CPU usage means I/O bottleneck.