Table of Contents

$ arch

Beginner

Print machine hardware architecture

arch prints the machine hardware architecture, equivalent to uname -m. It outputs a string like x86_64, aarch64, armv7l, or i686 identifying the CPU architecture.

arch is a simple utility primarily used in scripts to determine which architecture-specific binaries or packages to install. It is le...

Options & Flags

`(no options)`	Print machine architecture

Practical Examples

Example: Show architecture

```
$ arch
x86_64
```

Displays the CPU architecture.

Example: Conditional download

```
$ [[ $(arch) == "x86_64" ]] && wget amd64.tar.gz || wget arm64.tar.gz
```

Downloads architecture-specific binary.

Example: Compare with uname

```
$ echo "arch: $(arch), uname -m: $(uname -m)"
arch: x86_64, uname -m: x86_64
```

Both commands return the same value.

Tips & Best Practices

Note: Same as uname -m: arch is equivalent to uname -m. Use whichever is available - uname is more universally installed.

Pro Tip: Architecture names: x86_64 = 64-bit Intel/AMD (also called amd64). aarch64 = 64-bit ARM (also called arm64). i686 = 32-bit Intel.

$ cal

Beginner

Display a calendar in the terminal

cal displays a calendar in the terminal. Without arguments, it shows the current month. It can display specific months, entire years, and supports various calendar systems.

cal is a quick reference tool for checking dates, day-of-week, and planning. It highlights today date in the current month ...

Options & Flags

`(no args)`	Show current month
`YEAR`	Show entire year
`MONTH YEAR`	Show specific month
`-3`	Show previous, current, and next month
`-y`	Show current year
`-j`	Show Julian days (day of year)

Practical Examples

Example: Current month

```
$ cal
    January 2024\nSu Mo Tu We Th Fr Sa\n    1  2  3  4  5  6\n 7  8  9 10 11 12 13\n14 15 16 17 18 19 20
```

Shows the current month with today highlighted.

Example: Three months

```
$ cal -3
```

Shows previous, current, and next month side by side.

Example: Full year

```
$ cal 2024
```

Shows all 12 months of 2024.

Example: Specific month

```
$ cal 12 2024
```

Shows December 2024.

Example: Julian days

```
$ cal -j
```

Shows day-of-year numbers instead of day-of-month.

Tips & Best Practices

Pro Tip: Quick date lookup: Need to know what day of the week a date falls on? cal MONTH YEAR shows it instantly.

Note: ncal alternative: ncal shows weeks in columns (Monday-Sunday) instead of rows, which some people find more readable.

Warning: Gregorian calendar: cal shows the Gregorian calendar. Historical dates before 1582 may show differently depending on your locale.

$ chroot

Advanced

Run command or shell with a different root directory

chroot changes the root directory for a command or shell session. The process and its children see the specified directory as / and cannot access files outside of it (without special effort).

chroot is used for system recovery (booting into a broken system), building packages in isolated environ...

Options & Flags

`NEWROOT`	New root directory
`COMMAND`	Command to run (default: /bin/sh)
`--userspec`	Run as specific user:group
`--groups`	Set supplementary groups

Practical Examples

Example: System recovery

```
$ sudo mount /dev/sda1 /mnt && sudo chroot /mnt /bin/bash
```

Mounts a broken system and chroots into it for repair.

Example: Fix bootloader

```
$ sudo chroot /mnt /bin/bash -c 'grub-install /dev/sda && update-grub'
```

Repairs GRUB from a live USB by chrooting into the installed system.

Example: Run in isolated env

```
$ sudo chroot /srv/build /bin/bash
```

Opens a shell in a minimal filesystem for package building.

Example: Recovery with proc/sys

```
$ sudo mount --bind /dev /mnt/dev && sudo mount -t proc proc /mnt/proc && sudo mount -t sysfs sys /mnt/sys && sudo chroot /mnt
```

Full recovery chroot with required virtual filesystems.

Example: Run specific command

```
$ sudo chroot /jail /usr/bin/nginx
```

Runs nginx in a chrooted environment.

Tips & Best Practices

Warning: Not a security boundary: chroot is NOT a security container. Root processes can escape chroot. Use proper containers (Docker, LXC) for security isolation.

Pro Tip: Mount virtual filesystems: For full system recovery: mount /dev, /proc, /sys, and /dev/pts before chrooting. Otherwise many tools will not work.

Note: debootstrap for minimal environments: Use debootstrap to create a minimal Debian/Ubuntu filesystem for chroot: debootstrap focal /path/to/chroot

$ date

Beginner

Display or set the system date and time

date displays or sets the system date and time. It supports extensive formatting options for creating timestamps, date calculations, and converting between date formats.

date is essential for log timestamps, backup naming, cron job reporting, and any script that needs to work with dates. It supp...

Options & Flags

`+FORMAT`	Custom output format
`-d`	Display specified date (not current)
`-u`	Display UTC time
`-s`	Set the system date/time
`-R`	RFC 2822 format (for email)
`-I`	ISO 8601 format
`+%s`	Unix timestamp (seconds since epoch)

Practical Examples

Example: Current date and time

```
$ date
Mon Jan 15 14:30:00 UTC 2024
```

Shows the current date and time.

Example: Custom format

```
$ date "+%Y-%m-%d %H:%M:%S"
2024-01-15 14:30:00
```

Formats date as YYYY-MM-DD HH:MM:SS.

Example: Unix timestamp

```
$ date +%s
1705327800
```

Shows seconds since January 1, 1970 (epoch).

Example: Yesterday date

```
$ date -d "yesterday" +%Y-%m-%d
2024-01-14
```

Calculates and formats yesterday date.

Example: Date arithmetic

```
$ date -d "+3 days" +%Y-%m-%d
2024-01-18
```

Calculates date 3 days in the future.

Tips & Best Practices

Pro Tip: Common format codes: %Y=year, %m=month, %d=day, %H=hour, %M=minute, %S=second, %s=epoch, %A=weekday name, %B=month name.

Note: Date math with -d: date -d supports: yesterday, tomorrow, last friday, 2 weeks ago, +3 months, next year, etc.

Warning: Setting time: On systemd systems, use timedatectl set-time instead of date -s. Ensure NTP is disabled first.

$ dmesg

Intermediate

Print kernel ring buffer messages

dmesg displays the kernel ring buffer messages - the log of kernel events since boot. It contains hardware detection, driver loading, errors, warnings, and system events from the kernel.

dmesg is essential for troubleshooting hardware issues, driver problems, disk errors, network interface probl...

Options & Flags

`-H`	Human-readable with pager
`-T`	Show human-readable timestamps
`-l`	Filter by log level
`-w`	Follow - wait for new messages
`-c`	Clear the ring buffer after printing
`--color`	Colorize output
`-f`	Filter by facility

Practical Examples

Example: View kernel messages

```
$ dmesg -T | tail -20
```

Shows the last 20 kernel messages with human-readable timestamps.

Example: Show errors and warnings

```
$ dmesg -T -l err,warn
```

Filters to show only error and warning level messages.

Example: Follow new messages

```
$ sudo dmesg -w
```

Continuously shows new kernel messages as they occur.

Example: USB device events

```
$ dmesg -T | grep -i usb | tail
```

Shows recent USB device events (plug/unplug).

Example: Disk errors

```
$ dmesg -T | grep -iE 'error|fail|bad|i/o' | tail
```

Searches for disk error messages in kernel log.

Tips & Best Practices

Pro Tip: Use -T for timestamps: dmesg without -T shows seconds since boot. -T converts to human-readable date/time.

Note: Permission on newer kernels: Some kernels restrict dmesg to root. Use sudo dmesg or check dmesg_restrict: sysctl kernel.dmesg_restrict.

Warning: Ring buffer wraps: The kernel ring buffer has a fixed size. Old messages are overwritten. For persistent logs, use journalctl -k.

$ dmidecode

Beginner

Read hardware information from BIOS/UEFI DMI table

The dmidecode command reads the DMI (Desktop Management Interface) / SMBIOS (System Management BIOS) data from the system BIOS or UEFI firmware. This provides detailed hardware information that is not available through other Linux tools - including motherboard model, BIOS version, RAM slot detail...

Options & Flags

`(no options)`	Dump all DMI data
`-t TYPE`	Show only specific DMI type
`-t bios`	Show BIOS information (type 0)
`-t system`	Show system manufacturer, model, serial (type 1)
`-t baseboard`	Show motherboard info (type 2)
`-t processor`	Show CPU socket info (type 4)
`-t memory`	Show RAM slots and installed DIMMs (types 16,17)
`-s KEYWORD`	Show only a specific string value
`-t chassis`	Show chassis/enclosure info

Practical Examples

Example: Check server model and serial

```
$ sudo dmidecode -t system | grep -E "Manufacturer|Product|Serial"
  Manufacturer: Dell Inc.\n  Product Name: PowerEdge R740\n  Serial Number: ABC1234
```

Get the server manufacturer, model name, and serial number - essential for support tickets and inventory.

Example: Check BIOS version

```
$ sudo dmidecode -t bios
```

View BIOS vendor, version, and release date. Important for firmware update planning.

Example: Check RAM configuration

```
$ sudo dmidecode -t memory | grep -E "Size|Type|Speed|Locator"
```

Show installed RAM modules: size, type (DDR4/DDR5), speed, and which physical slot they occupy.

Example: Find maximum RAM capacity

```
$ sudo dmidecode -t 16 | grep "Maximum Capacity"
  Maximum Capacity: 512 GB
```

Check the maximum RAM the system supports - crucial for memory upgrade planning.

Example: Count available RAM slots

```
$ sudo dmidecode -t 17 | grep -c "Size:"
16
```

Count total DIMM slots (including empty ones).

Tips & Best Practices

Note: Requires root access: dmidecode reads from /dev/mem or /sys/firmware/dmi and requires root privileges. Always run with sudo.

Pro Tip: Virtual machines: In VMs, dmidecode shows the virtualization platform (VMware, KVM, VirtualBox, Hyper-V) as the manufacturer. Useful for detecting if you are on physical or virtual hardware.

Warning: Data accuracy: DMI data is written by the BIOS/UEFI manufacturer. Some fields may be inaccurate or empty on cheap hardware. Cross-reference with lscpu, lspci, and free for verification.

Pro Tip: Useful -s keywords: Quick single values: dmidecode -s system-product-name, -s system-serial-number, -s bios-version, -s baseboard-product-name, -s processor-version

$ free

Beginner

Display amount of free and used memory

free displays the amount of free and used memory (RAM) and swap in the system. It shows total, used, free, shared, buffer/cache, and available memory.

The "available" column is the most useful - it shows how much memory is actually available for new applications, accounting for reclaimable cache...

Options & Flags

`-h`	Human-readable (KB, MB, GB)
`-m`	Show in megabytes
`-g`	Show in gigabytes
`-s N`	Update every N seconds
`-t`	Show total row
`-w`	Wide output (separate buffers and cache)

Practical Examples

Example: Human-readable memory

```
$ free -h
               total        used        free      shared  buff/cache   available
Mem:           15Gi       4.2Gi       1.3Gi       256Mi        10Gi        10Gi
Swap:         2.0Gi          0B       2.0Gi
```

Shows memory usage in human-readable format.

Example: Monitor memory

```
$ free -h -s 5
```

Updates memory display every 5 seconds.

Example: Available memory

```
$ free -h | awk '/Mem:/{print $7}'
10Gi
```

Extracts just the available memory value.

Example: Memory percentage

```
$ free | awk '/Mem:/{printf "%.1f%%\n", $3/$2*100}'
28.0%
```

Calculates memory usage percentage.

Example: Wide output

```
$ free -hw
```

Separates buffers and cache into distinct columns.

Tips & Best Practices

Warning: Free is not available: \"Free\" memory looks low because Linux uses RAM for disk cache. Look at the \"available\" column - it shows actual available memory for applications.

Pro Tip: Quick memory check: free -h | grep Mem gives you the essential memory line: total, used, free, and available in one glance.

Note: Swap usage: Consistent swap usage is OK. Active swapping (swap growing over time) indicates insufficient RAM. Monitor with vmstat.

$ hostname

Beginner

Show or set the system hostname

hostname displays or sets the system hostname. The hostname identifies the machine on a network and is used in shell prompts, log files, and network communications.

hostname can show the short hostname, the fully qualified domain name (FQDN), and the IP addresses associated with the hostname. Se...

Options & Flags

`-f`	Show fully qualified domain name (FQDN)
`-I`	Show all IP addresses
`-i`	Show IP address for the hostname
`-s`	Show short hostname
`-d`	Show domain name
`NAME`	Set hostname temporarily

Practical Examples

Example: Show hostname

```
$ hostname
webserver01
```

Displays the current system hostname.

Example: Show FQDN

```
$ hostname -f
webserver01.example.com
```

Shows the fully qualified domain name.

Example: Show IP addresses

```
$ hostname -I
192.168.1.100 10.0.0.5
```

Shows all network IP addresses.

Example: Set hostname permanently

```
$ sudo hostnamectl set-hostname newname
```

Permanently changes the hostname (systemd systems).

Example: Set hostname temporarily

```
$ sudo hostname tempname
```

Changes hostname until next reboot.

Tips & Best Practices

Pro Tip: Use hostnamectl for permanent changes: hostname changes are temporary. Use sudo hostnamectl set-hostname name for permanent changes on systemd systems.

Note: hostname -I for scripts: hostname -I returns IP addresses without DNS resolution - faster and more reliable for scripts than hostname -i.

Warning: Update /etc/hosts too: After changing hostname, update /etc/hosts to map the new hostname to 127.0.1.1 to avoid DNS resolution issues.

$ lsblk

Beginner

List information about block devices (disks)

lsblk lists information about all available block devices (disks, partitions, LVM volumes, etc.). It shows device names, sizes, types, mount points, and filesystem types in a tree format.

lsblk is the primary tool for understanding disk layout - which disks are present, how they are partitioned,...

Options & Flags

`-f`	Show filesystem type, label, UUID, and mount point
`-l`	List format (no tree)
`-o`	Specify output columns
`-d`	Show only disk devices (no partitions)
`-p`	Print full device paths
`-J`	JSON output
`-b`	Print sizes in bytes

Practical Examples

Example: List all block devices

```
$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0   500G  0 disk
??sda1   8:1    0   512M  0 part /boot/efi
??sda2   8:2    0   499G  0 part /
```

Shows disks and partitions in tree format.

Example: Show filesystems

```
$ lsblk -f
```

Shows filesystem type, label, UUID, and mount point.

Example: Custom columns

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

Shows only specific columns.

Example: Disks only

```
$ lsblk -d -o NAME,SIZE,MODEL
NAME SIZE MODEL
sda  500G Samsung SSD 870
sdb  1T   WD Blue
```

Shows only disk devices with model names.

Example: JSON output

```
$ lsblk -J
```

Outputs device information in JSON format for scripting.

Tips & Best Practices

Pro Tip: Use -f for UUIDs: lsblk -f shows filesystem UUIDs needed for /etc/fstab entries. More reliable than device names which can change.

Note: lsblk vs fdisk: lsblk shows device info without root. fdisk -l shows partition tables but requires root. Use lsblk for viewing, fdisk for modifying.

Warning: Not all devices shown: lsblk shows block devices only. Network mounts (NFS), tmpfs, and other non-block filesystems are not shown. Use df for mount points.

$ lscpu

Beginner

Display CPU architecture information

lscpu displays detailed information about the CPU architecture. It shows the number of CPUs, cores, threads, sockets, cache sizes, CPU model, architecture, and virtualization capabilities.

lscpu reads from /proc/cpuinfo and sysfs, presenting the information in an organized, readable format. It i...

Options & Flags

`(no flags)`	Show CPU information
`-e`	Extended readable format
`-p`	Parseable format
`-J`	JSON output

Practical Examples

Example: CPU information

```
$ lscpu
Architecture:          x86_64
CPU(s):                8
Thread(s) per core:    2
Core(s) per socket:    4
```

Shows complete CPU architecture information.

Example: JSON output

```
$ lscpu -J
```

Shows CPU info in JSON format for programmatic use.

Example: Extended format

```
$ lscpu -e
CPU NODE SOCKET CORE ONLINE\n0   0    0      0    yes
```

Shows per-CPU information in table format.

Example: Quick core count

```
$ nproc
8
```

Shows just the number of available processing units.

Example: Check virtualization

```
$ lscpu | grep 'Virtualization\|Hypervisor'
Hypervisor vendor: KVM\nVirtualization type: full
```

Checks if system is virtualized and what technology is available.

Tips & Best Practices

Pro Tip: nproc for core count: If you just need the CPU count for make -j or parallelization, use nproc instead of parsing lscpu output.

Note: Threads vs cores: Threads per core > 1 means hyperthreading is enabled. 4 cores with 2 threads = 8 logical CPUs, but only 4 physical cores.

Warning: Container CPUs: In containers, lscpu shows host CPU info. Check cgroup limits: cat /sys/fs/cgroup/cpu/cpu.cfs_quota_us for actual available CPU.