parted Cheatsheet

By Dejan Panovski • Updated on Aug 25, 2026 •

Quick reference for parted commands: inspect disks, create GPT and MBR tables, add aligned partitions, manage flags, resize boundaries, and remove partitions safely

The `parted` command creates and changes disk partition tables from the terminal. This cheatsheet covers disk inspection, GPT and MBR labels, aligned partitions, flags, boundary resizing, removal, and the safety checks to run before every write operation.

Basic Usage

Core command forms and help options.

`parted [OPTIONS] DEVICE [COMMAND [ARGUMENTS]]`	General syntax
`sudo parted /dev/sdX`	Open a disk in interactive mode
`sudo parted /dev/sdX -- print`	Print one disk's partition table
`sudo parted --list`	List partition tables on all detected disks
`parted --help`	Show command-line options
`parted --version`	Show the installed version

Always pass the whole-disk path explicitly. Most `parted` changes take effect immediately, even in interactive mode.

Identify the Target Disk

Confirm the device name, size, model, and current use before changing it.

`lsblk -d -o NAME,SIZE,MODEL,TRAN`	List whole disks with identifying details
`lsblk -o NAME,SIZE,TYPE,FSTYPE,MOUNTPOINTS,MODEL`	Show disks, partitions, filesystems, and mounts
`lsblk -f`	Show filesystem labels and UUIDs
`findmnt -S /dev/sdX1`	Check whether a partition is mounted
`sudo parted /dev/sdX -- print`	Review the current partition table
`sudo parted /dev/sdX -- unit MiB print free`	Show partitions and unallocated space

Use `/dev/sdX` for the whole disk and `/dev/sdX1` for its first partition. NVMe and eMMC disks insert `p` before the number, so partition 1 of `/dev/nvme0n1` is `/dev/nvme0n1p1`. The `MOUNTPOINTS` column needs util-linux 2.37 or later; use `MOUNTPOINT` on older releases. Replace every placeholder with a verified device path.

Command-Line Options

Options must appear before the device and session command.

`-l, --list`	List partition layouts on all block devices
`-s, --script`	Never prompt for input
`-m, --machine`	Print colon-separated, machine-readable output
`-j, --json`	Print JSON output on supported versions
`-f, --fix`	Automatically choose `fix` for repairable exceptions in script mode
`-a optimal, --align optimal`	Use optimal alignment for new partitions
`--`	End option parsing before session commands and negative positions

Script mode is not a dry run. Inspect the disk separately before using `--script` or `--fix`.

Interactive Commands

Enter these commands at the `(parted)` prompt.

`help`	List available session commands
`help mkpart`	Show help for one command
`print`	Print the current partition table
`print free`	Include unallocated space
`unit MiB`	Set the default display and input unit
`select /dev/sdY`	Switch to another disk
`quit`	Exit Parted

The `quit` command does not undo earlier changes because Parted writes most changes as each command runs.

Create a Partition Table

Create a disk label on a new or intentionally cleared disk.

`sudo parted --script /dev/sdX -- mklabel gpt`	Create a GPT partition table
`sudo parted --script /dev/sdX -- mklabel msdos`	Create an MBR partition table
`sudo parted /dev/sdX -- print`	Confirm the new table type

`mklabel` replaces the existing partition table and makes its partitions inaccessible. Use GPT for current systems; use MBR only when legacy compatibility requires it.

Create GPT Partitions

Give each GPT partition a name, filesystem type hint, start, and end.

`sudo parted --script /dev/sdX -- mkpart data ext4 1MiB 100%`	Create one partition using the available disk
`sudo parted --script /dev/sdX -- mkpart data ext4 1MiB 50GiB`	Create a partition ending at 50 GiB
`sudo parted --script /dev/sdX -- mkpart backup ext4 50GiB 100%`	Use the remaining space for a second partition
`sudo parted /dev/sdX -- print`	Review the finished layout
`sudo parted /dev/sdX -- align-check optimal 1`	Check partition 1 alignment

The filesystem type is a partition-table hint. `mkpart` does not create a filesystem.

Create MBR Partitions

An MBR table takes a partition type where GPT takes a name.

`sudo parted --script /dev/sdX -- mkpart primary ext4 1MiB 100%`	Create one primary partition using the available disk
`sudo parted --script /dev/sdX -- mkpart primary ext4 1MiB 50GiB`	Create a primary partition ending at 50 GiB
`sudo parted --script /dev/sdX -- mkpart primary ext4 50GiB 100%`	Add a second primary partition
`sudo parted --script /dev/sdX -- mkpart extended 50GiB 100%`	Create an extended partition to hold logical partitions
`sudo parted --script /dev/sdX -- mkpart logical ext4 51GiB 100%`	Create a logical partition inside the extended partition
`sudo parted /dev/sdX -- print`	Review the finished layout

An MBR table holds four primary partitions, or three primary partitions plus one extended partition that contains the logical ones. Start the first logical partition after the extended partition's start so Parted can write the extended boot record. The `name` command does not work on MBR partitions.

Units and Positions

Suffix boundary values so Parted does not have to infer the unit.

`s`	Sectors, used for exact sector positions
`MiB, GiB, TiB`	Exact IEC binary positions
`MB, GB, TB`	Decimal positions that may allow a nearby boundary
`1MiB`	Common aligned start for a new partition
`100%`	End at the last usable part of the disk
`-1s`	Last sector of the disk; put `--` before the command
`sudo parted /dev/sdX -- unit s print`	Display all boundaries in sectors

Use `MiB` or `GiB` for repeatable commands and `unit MiB print free` when comparing boundaries.

Partition Names and Flags

Names and available flags depend on the partition table type.

`sudo parted --script /dev/sdX -- name 1 data`	Rename GPT partition 1
`sudo parted --script /dev/sdX -- set 1 esp on`	Mark an EFI System Partition
`sudo parted --script /dev/sdX -- set 1 bios_grub on`	Mark a BIOS GRUB partition on GPT
`sudo parted --script /dev/sdX -- set 1 swap on`	Mark a swap partition
`sudo parted --script /dev/sdX -- set 1 lvm on`	Mark an LVM physical volume where supported
`sudo parted --script /dev/sdX -- set 1 raid on`	Mark a software RAID member where supported
`sudo parted --script /dev/sdX -- set 1 FLAG off`	Disable a flag

Run `help set` interactively to see the flags supported by the current table. Flags record a partition's purpose but do not create an EFI filesystem, swap area, LVM volume, or RAID array.

Resize a Partition Boundary

Confirm that free space is adjacent to the partition before extending it.

`sudo parted /dev/sdX -- unit MiB print free`	Check the partition and adjacent free space
`sudo parted --script /dev/sdX -- resizepart 1 100%`	Move partition 1's end to the end of the disk
`sudo partprobe /dev/sdX`	Ask the kernel to reread the partition table
`sudo resize2fs /dev/sdX1`	Grow an ext2, ext3, or ext4 filesystem afterward
`sudo xfs_growfs /mount/point`	Grow a mounted XFS filesystem afterward
`lsblk -f`	Check the resulting partition and filesystem size

`resizepart` changes only the partition boundary. When shrinking, shrink the filesystem first and use a filesystem-specific procedure.

Remove or Rescue a Partition

Keep inspection and removal as separate commands.

`sudo parted /dev/sdX -- print`	Note the target partition number and boundaries
`sudo parted --script /dev/sdX -- rm 2`	Remove partition 2 immediately
`sudo parted /dev/sdX`	Open interactive mode for recovery
`rescue START END`	Search near the old boundaries for a lost partition
`sudo partprobe /dev/sdX`	Ask the kernel to reread the changed table

After an accidental removal, stop writing to the disk and try `rescue` before creating another partition. Recovery is not guaranteed.

Format and Mount

Create a filesystem only after checking the new partition path.

`lsblk -f /dev/sdX`	Confirm the new partition and existing filesystems
`sudo mkfs.ext4 -L data /dev/sdX1`	Create an ext4 filesystem with label `data`
`sudo mkfs.xfs -L data /dev/sdX1`	Create an XFS filesystem with label `data`
`sudo mkswap -L swap /dev/sdX2`	Create swap space on partition 2
`sudo mkdir -p /mnt/data`	Create a mount point
`sudo mount /dev/sdX1 /mnt/data`	Mount the new filesystem
`lsblk -f`	Confirm filesystem and mount details

Formatting destroys data on the selected partition. A partition name and a filesystem label are separate values.

Troubleshooting

Common Parted problems and the next check to run.

`Partition is not aligned`	Run `align-check optimal NUMBER`; recreate an empty partition with a suitable start
`Kernel still shows the old table`	Unmount filesystems, disable swap, then run `sudo partprobe /dev/sdX` or reboot
`Device or resource is busy`	Check `lsblk`, `findmnt`, swap, LVM, RAID, and encrypted mappings
`New partition does not appear`	Run `sudo partprobe /dev/sdX`, then check `lsblk`; reboot if the disk remains busy
`unrecognised disk label`	The disk has no supported table; verify it is the correct empty disk before using `mklabel`
`mkpart rejects the partition name`	The disk uses an MBR table; pass `primary`, `extended`, or `logical` instead of a name
`Filesystem size did not change`	Run the correct filesystem grow command after `resizepart`

Related Guides

Use these guides and cheatsheets for the complete storage workflow.

parted Command in Linux	Full walkthrough for inspecting and changing partitions
Linux Block Devices, Partitions, Filesystems, and Mount Points	Understand where partitioning fits in the storage stack
fdisk Cheatsheet	Menu-driven partitioning command reference
mount Cheatsheet	Mount options, UUIDs, labels, and `/etc/fstab` entries