Table of Contents

Part 2 of 2 - Explore all 232+ Linux commands at dargslan.com/learn/linux-commands
Each command includes syntax, options, practical examples with output, and pro tips.

$ parted

Advanced

GPT and MBR partition management tool

The parted command is a disk partitioning tool that supports both GPT (GUID Partition Table) and MBR (Master Boot Record) partition schemes. Unlike fdisk which historically only supported MBR, parted handles GPT partitions natively, making it essential for modern systems with drives larger than 2...

Options & Flags

`print`	Display partition table of a device
`mklabel gpt`	Create a new GPT partition table
`mkpart`	Create a new partition
`resizepart NUM END`	Resize a partition to new end position
`rm NUM`	Delete a partition by number
`set NUM FLAG on/off`	Set a flag on a partition (boot, lvm, raid)
`-l`	List partition tables on all block devices
`-s`	Script mode - never prompt for user input
`align-check optimal NUM`	Check if partition is optimally aligned

Practical Examples

Example: View partition table

```
$ sudo parted /dev/sda print
Model: ATA VBOX HARDDISK (scsi)\nDisk /dev/sda: 107GB\nPartition Table: gpt\n\nNumber  Start   End     Size    File system  Name
```

Display the partition table with sizes, filesystem types, and flags.

Example: Create GPT partition table on new disk

```
$ sudo parted -s /dev/sdb mklabel gpt
```

Initialize a new disk with GPT partition table. Required before creating partitions. WARNING: destroys all existing data.

Example: Create a single partition using full disk

```
$ sudo parted -s /dev/sdb mkpart primary ext4 0% 100%
```

Create a single partition spanning the entire disk with optimal alignment using percentages.

Example: Create multiple partitions

```
$ sudo parted -s /dev/sdb mkpart primary ext4 0% 50% && sudo parted -s /dev/sdb mkpart primary ext4 50% 100%
```

Split a disk into two equal partitions.

Example: Resize a partition

```
$ sudo parted /dev/sda resizepart 2 100%
```

Extend partition 2 to use all available space. Follow with resize2fs /dev/sda2 for ext4 or xfs_growfs for XFS.

Tips & Best Practices

Warning: parted changes are immediate: Unlike fdisk which requires writing changes, parted applies changes IMMEDIATELY. There is no undo. Double-check the device path before any destructive operation.

Pro Tip: Use percentages for alignment: Use 0% and 100% instead of exact byte values for optimal partition alignment. This ensures partitions align to physical sector boundaries.

Note: GPT vs MBR: Use GPT for disks larger than 2TB, UEFI systems, or when you need more than 4 primary partitions. MBR is only needed for legacy BIOS systems.

Pro Tip: Resize requires filesystem resize too: parted resizepart only changes the partition boundary. You also need to resize the filesystem: resize2fs (ext4), xfs_growfs (XFS), or btrfs filesystem resize (Btrfs).

$ smartctl

Intermediate

Monitor disk health with SMART data

The smartctl command reads SMART (Self-Monitoring, Analysis, and Reporting Technology) data from hard drives and SSDs. SMART is a monitoring system built into most modern storage devices that tracks drive health indicators like temperature, read errors, reallocated sectors, power-on hours, and we...

Options & Flags

`-i`	Show drive identity and basic info
`-H`	Check overall health status (PASSED/FAILED)
`-A`	Show all SMART attributes
`-a`	Show all SMART information (combines -i, -H, -A)
`-t short`	Run a short self-test (~2 minutes)
`-t long`	Run a long/extended self-test (hours)
`-l selftest`	Show self-test log/results
`-l error`	Show error log
`-d TYPE`	Specify device type (sat, nvme, scsi, megaraid)
`--scan`	Scan for devices and their types

Practical Examples

Example: Quick health check

```
$ sudo smartctl -H /dev/sda
SMART overall-health self-assessment test result: PASSED
```

Check overall health status. Returns PASSED or FAILED.

Example: View all SMART attributes

```
$ sudo smartctl -A /dev/sda
```

Display all SMART attributes including temperature, power-on hours, reallocated sectors, and error rates.

Example: Full drive information

```
$ sudo smartctl -a /dev/sda
```

Complete SMART report: identity, health, attributes, error log, and self-test history.

Example: Check NVMe SSD

```
$ sudo smartctl -a /dev/nvme0
```

NVMe drives report different attributes: temperature, available spare, percentage used, data read/written.

Example: Run short self-test

```
$ sudo smartctl -t short /dev/sda && sleep 120 && sudo smartctl -l selftest /dev/sda
```

Start a short self-test (~2 min), wait, then check results.

Tips & Best Practices

Warning: Replace on Reallocated Sectors: If Reallocated_Sector_Ct (ID 5) is non-zero and increasing, the drive is actively failing. Plan replacement immediately and backup all data.

Pro Tip: Automate with smartd: Enable the smartd daemon for continuous monitoring. Configure /etc/smartd.conf to run periodic tests and email alerts on attribute changes.

Note: Power-On Hours: Enterprise HDDs are rated for ~50,000 power-on hours (~5.7 years 24/7). Consumer drives for ~20,000-30,000 hours. Check attribute 9 (Power_On_Hours) for drive age.

Pro Tip: Install smartmontools: Install with: apt install smartmontools (Debian/Ubuntu), dnf install smartmontools (Fedora/RHEL). Includes both smartctl and the smartd daemon.

$ swapon

Intermediate

Enable and manage swap space

The swapon command enables swap space on a device or file, making it available for the kernel's virtual memory system. Swap provides overflow space when physical RAM is full - the kernel moves less-used memory pages to swap, freeing RAM for active processes.

Swap is critical for system stability...

Options & Flags

`-s`	Display swap usage summary
`--show`	Show swap devices in columns format
`-a`	Enable all swaps from /etc/fstab
`-p PRIORITY`	Set swap priority (higher = used first)
`DEVICE`	Enable swap on a specific device or file
`--verbose`	Verbose output showing details

Practical Examples

Example: Show active swap

```
$ swapon --show
NAME      TYPE SIZE  USED PRIO\n/swapfile file   4G 256M   -2
```

Display all active swap devices/files with size, type, and usage.

Example: Create and enable a swap file

```
$ sudo fallocate -l 4G /swapfile && sudo chmod 600 /swapfile && sudo mkswap /swapfile && sudo swapon /swapfile
```

Create a 4GB swap file: allocate space, set secure permissions, format as swap, and activate.

Example: Make swap permanent

```
$ echo "/swapfile none swap sw 0 0" | sudo tee -a /etc/fstab
```

Add swap file to /etc/fstab so it activates on boot.

Example: Disable swap temporarily

```
$ sudo swapoff /swapfile
```

Disable a swap file. All data in swap is moved back to RAM - ensure enough free RAM first.

Example: Resize swap file

```
$ sudo swapoff /swapfile && sudo fallocate -l 8G /swapfile && sudo mkswap /swapfile && sudo swapon /swapfile
```

Resize swap from 4GB to 8GB. Must disable first, recreate, then re-enable.

Tips & Best Practices

Pro Tip: Recommended swap size: Servers: equal to RAM for hibernation, or 1-2GB minimum for stability. If RAM >16GB and no hibernation, 2-4GB swap is usually sufficient.

Warning: swapoff needs free RAM: swapoff moves ALL swap data back to RAM. If there is not enough free RAM, swapoff will hang or fail. Check free -h before running swapoff.

Note: Swappiness: Control how aggressively Linux uses swap: sysctl vm.swappiness=10 (10 = prefer RAM, 60 = default, 100 = swap aggressively). For databases, use 10.

Pro Tip: Swap file vs partition: Swap files are easier to resize and manage than swap partitions. Performance is identical on ext4/XFS. Use swap files unless you have a specific reason for partitions.

$ umount

Intermediate

Unmount a filesystem

umount detaches a mounted filesystem from the directory tree. It flushes any cached writes to the device before disconnecting, ensuring data integrity.

umount is essential before removing USB drives, external disks, or any removable storage. Removing a device without unmounting can cause data co...

Options & Flags

`-l`	Lazy unmount (detach immediately, cleanup later)
`-f`	Force unmount (for unreachable NFS)
`-a`	Unmount all filesystems
`-R`	Recursively unmount

Practical Examples

Example: Unmount filesystem

```
$ sudo umount /mnt/usb
```

Safely unmounts the USB drive.

Example: Unmount by device

```
$ sudo umount /dev/sdb1
```

Unmounts using the device name instead of mount point.

Example: Lazy unmount

```
$ sudo umount -l /mnt/busy
```

Detaches immediately. Cleanup happens when processes stop using it.

Example: Force NFS unmount

```
$ sudo umount -f /mnt/nfs
```

Forces unmount of an unreachable NFS share.

Example: Find blocking processes

```
$ fuser -vm /mnt/data
```

Shows which processes are preventing unmount.

Tips & Best Practices

Warning: Always unmount before removing: Never unplug a USB drive or external disk without unmounting first. Data in write cache will be lost.

Pro Tip: Target busy mounts: If 'device is busy': use fuser -vm /mnt to find blocking processes, or lsof /mnt. Kill them, then retry.

Note: Lazy unmount as last resort: umount -l detaches immediately but processes can still access cached data. It is cleaner than force unmount.