Linux Commands Cheatsheet

Essential commands for system administration and daily use

This cheatsheet provides a quick reference to fundamental Linux commands, syntax, and advanced features, ideal for both beginners and experienced administrators.

File Operations
Manage files and directories

System Info
Monitor system resources

Process Management
Control running applications

Network Commands
Diagnose network issues

Text Processing
Manipulate text streams

File & Directory Operations

List Content: `ls`

List directory contents.

```
ls          # List files in current directory
ls -l       # Long format (permissions, owner, size, date)
ls -a       # List all files, including hidden ones (starting with .)
ls -lh      # Long format, human-readable sizes
```

Change Directory: `cd`

Navigate the filesystem.

```
cd ~        # Go to home directory
cd /var/log # Go to an absolute path
cd ..       # Go up one directory
cd -        # Go to the previous directory
```

Remove: `rm`

Delete files and directories.

```
rm myfile.txt       # Delete a file
rm -i myfile.txt    # Interactive delete (prompts for confirmation)
rm -r mydirectory   # Delete a directory and its contents recursively
rm -rf mydirectory  # Force recursive delete (use with caution!)
```

Make Directory: `mkdir`

Create new directories.

```
mkdir newdir        # Create a new directory
mkdir -p parent/child # Create parent directories as needed
```

Copy: `cp`

Copy files and directories.

```
cp file.txt /tmp/   # Copy file to /tmp/
cp file.txt newfile.txt # Copy file and rename it
cp -r dir1 dir2     # Copy directory recursively
```

Move/Rename: `mv`

Move or rename files and directories.

```
mv file.txt /tmp/   # Move file to /tmp/
mv oldname.txt newname.txt # Rename file
```

File Permissions

Manage access rights to files and directories.

01

Change Permissions: `chmod`

Modifies file/directory permissions (read, write, execute) for owner, group, and others.

```
# Symbolic mode
chmod u+x script.sh # Add execute permission for user
chmod go-w file.txt # Remove write for group and others
chmod a=rwx dir/    # Set all permissions (user, group, others) to rwx

# Numeric (octal) mode
# r=4, w=2, x=1
chmod 755 script.sh # rwx for owner, r-x for group/others
chmod 644 file.txt  # rw for owner, r for group/others
```

02

Change Owner: `chown`

Changes the user owner of a file or directory.

```
chown user file.txt     # Change owner to 'user'
chown user:group file.txt # Change owner and group to 'user' and 'group'
chown -R user dir/      # Recursively change owner for a directory
```

03

Change Group: `chgrp`

Changes the group owner of a file or directory.

```
chgrp group file.txt    # Change group to 'group'
chgrp -R group dir/     # Recursively change group for a directory
```

Text Processing

Concatenate/Display: `cat`

Displays file content and concatenates files.

```
cat file.txt        # Display content of file.txt
cat file1.txt file2.txt # Concatenate and display two files
cat -n file.txt     # Display content with line numbers
```

Search Text: `grep`

Searches for patterns in files.

```
grep "pattern" file.txt  # Search for 'pattern' in file.txt
grep -i "pattern" file.txt # Case-insensitive search
grep -r "pattern" dir/   # Recursive search in directory
grep -l "pattern" dir/   # List files containing the pattern
grep -v "pattern" file.txt # Invert match (show lines without pattern)
```

Stream Editor: `sed`

Filters and transforms text streams.

```
# Replace "old" with "new" in file.txt and print to stdout
sed 's/old/new/' file.txt
# Replace all occurrences of "old" with "new" in each line
sed 's/old/new/g' file.txt
# Replace "old" with "new" in file.txt and save changes back to file (in-place)
sed -i 's/old/new/g' file.txt
```

Pattern Scanning/Processing: `awk`

Powerful text processing language, often used for data extraction and reporting.

```
# Print the first and third fields (columns) of each line in file.txt
awk '{print $1, $3}' file.txt
# Print lines where the second field equals "some_value"
awk '$2 == "some_value" {print $0}' file.txt
# Calculate sum of a column
awk '{sum += $1} END {print sum}' numbers.txt
```

System Information

Process Status: `ps` / `top`

View currently running processes.

```
ps aux          # List all running processes
ps -ef          # Display all processes in full format
top             # Real-time view of running processes
top -u username # Show processes for a specific user
```

Disk Free: `df`

Report filesystem disk space usage.

```
df -h           # Display disk space in human-readable format
df -i           # Display inode usage
```

Disk Usage: `du`

Estimate file space usage.

```
du -h /path/to/dir # Summarize disk usage for a directory in human-readable format
du -sh /path/to/dir # Summarize only total disk usage for a directory
```

System Name: `uname`

Print system information.

```
uname -a        # Print all system information
uname -s        # Print the kernel name
uname -r        # Print the kernel release
```

Process Management

Control the lifecycle of running programs.

Terminate Process: `kill`

Send signals to processes to terminate or manage them.

```
kill PID        # Send SIGTERM (graceful termination) to process ID
kill -9 PID     # Send SIGKILL (forceful termination)
killall processname # Kill all processes with a given name
```

Run Detached: `nohup`

Run a command immune to hangup signals, typically in the background.

```
nohup script.sh &  # Run script in background, detached from terminal
```

Background/Foreground Jobs: `bg`, `fg`

Manage jobs running in the background or foreground.

```
# Run a command in background
sleep 60 &
jobs            # List background jobs
fg %1           # Bring job 1 to foreground
bg              # Resume stopped job in background
```

Network Commands

Diagnose network connectivity and transfer data.

Test Connectivity: `ping`

Send ICMP ECHO_REQUEST packets to network hosts.

```
ping google.com  # Ping a host
ping -c 4 google.com # Ping a host 4 times
```

Download from Web: `wget`

Non-interactive network downloader.

```
wget https://example.com/file.zip # Download a file
```

Transfer Data: `curl`

Tool for transferring data with URL syntax, supporting various protocols.

```
curl https://example.com/api # Make a GET request
curl -X POST -d "data" https://example.com/api # Make a POST request
```

Secure Shell: `ssh`

Securely connect to remote servers.

```
ssh user@remote_host # Connect to a remote host
ssh -p 2222 user@remote_host # Connect using a specific port
```

Archive & Compression

Bundle and compress files for storage and transfer.

Tape Archiver: `tar`

Creates and extracts archives.

```
tar -cvf archive.tar dir/ # Create a .tar archive
tar -xvf archive.tar      # Extract from a .tar archive
tar -czvf archive.tar.gz dir/ # Create a gzipped .tar archive
tar -xzvf archive.tar.gz  # Extract from a gzipped .tar archive
```

Compress/Decompress: `gzip`, `zip`

Tools for compressing files.

```
gzip file.txt       # Compress file.txt to file.txt.gz
gunzip file.txt.gz  # Decompress file.txt.gz
zip archive.zip file1 file2 # Create a .zip archive
unzip archive.zip   # Extract from a .zip archive
```

Search & Find

Locate files and commands within the filesystem.

Find Files: `find`

Search for files in a directory hierarchy.

```
find . -name "*.log"    # Find files ending with .log in current dir
find / -type d -name "config" # Find directories named config from root
find . -size +1G        # Find files larger than 1GB
```

Locate Files: `locate`

Find files by name using a prebuilt database (faster, but may be outdated).

```
sudo updatedb           # Update the database (run periodically)
locate filename         # Find files matching filename
```

Locate Command: `which`

Shows the full path of (shell) commands.

```
which bash      # Show path to bash executable
which python    # Show path to python executable
```

System Monitoring

Interactive Process Viewer: `htop`

An interactive process viewer, an enhanced version of `top`.

```
htop            # Start htop (install if not available: sudo apt install htop)
```

Network Statistics: `netstat`

Display network connections, routing tables, interface statistics, etc.

```
netstat -tulnp  # List all TCP/UDP connections with program name
netstat -r      # Display routing table
```

I/O Statistics: `iostat`

Report CPU utilization and disk I/O statistics.

```
iostat -x       # Extended statistics for devices
```

Package Management

Install, update, and remove software packages on various distributions.

Debian/Ubuntu: `apt`

Advanced Package Tool for Debian-based systems.

```
sudo apt update     # Update package lists
sudo apt upgrade    # Upgrade installed packages
sudo apt install apache2 # Install a package
sudo apt remove apache2  # Remove a package
```

Red Hat/CentOS: `yum` / `dnf`

Package managers for RHEL-based systems. `dnf` is the successor to `yum`.

```
# Using yum (older)
sudo yum check-update  # Check for updates
sudo yum install nginx # Install nginx

# Using dnf (modern)
sudo dnf update     # Update packages
sudo dnf install vim # Install vim
```

Others: `pacman` (Arch), `zypper` (OpenSUSE)

Different distributions use different package managers.

```
# Arch Linux (pacman)
sudo pacman -Syu    # Sync, refresh, and upgrade
sudo pacman -S htop # Install htop

# OpenSUSE (zypper)
sudo zypper update  # Update packages
sudo zypper install git # Install git
```

Environment Variables

Dynamic named values that affect the way processes run.

Set Variable: `export`

Set an environment variable for child processes.

```
export MY_VAR="hello" # Set a temporary environment variable
echo $MY_VAR        # Access the variable's value
```

Path Variable: `PATH`

A list of directories the shell searches for executable commands.

```
echo $PATH          # Display current PATH
# Add a directory to the PATH (temporary for current session)
export PATH=$PATH:/usr/local/bin/my_scripts
```

Display Variables: `env`

Displays all environment variables.

```
env             # List all environment variables
env | grep USER # Find specific variables
```

Redirection & Pipes

Control input/output streams and chain commands.

Output Redirection: `>`, `>>`

Redirect command output to a file.

```
ls > files.txt  # Redirect stdout to files.txt (overwrites)
ls -l >> files.txt # Redirect stdout to files.txt (appends)
command 2> error.log # Redirect stderr to error.log
command &> output.log # Redirect both stdout and stderr to output.log
```

Pipes: `|`

Send the output of one command as input to another.

```
ls -l | grep "myfile" # Find "myfile" in the output of `ls -l`
cat access.log | awk '{print $1}' | sort | uniq # Extract, sort, and count unique IPs
```

Split Output: `tee`

Read from standard input and write to standard output and files.

```
ls -l | tee output.txt # Display output and save to output.txt
```

Reference: This cheatsheet covers essential Linux commands and modern practices for efficient system administration and development.