# Linux Commands Cheat Sheet

A comprehensive guide to essential Linux commands for DevOps professionals, system administrators, and developers.

---

## Table of Contents



---

## File and Directory Operations

### pwd (Print Working Directory)

**Theory:** Displays the absolute path of the current working directory. Essential for understanding your location in the file system hierarchy.

```
pwd
```

### cd (Change Directory)

**Theory:** Navigates between directories in the file system. Supports both absolute and relative paths.

```
# Navigate to home directory
cd ~

# Navigate to parent directory
cd ..

# Navigate to root directory
cd /

# Navigate to a specific directory
cd /usr/local/bin

# Navigate back to previous directory
cd -
```

### ls (List Directory Contents)

**Theory:** Lists files and directories with various options for detailed information, including permissions, ownership, and timestamps.

```
# Simple list
ls
```

```
# Long format with details
ls -l

# Include hidden files
ls -la

# Human-readable file sizes
ls -lh

# Sort by modification time
ls -lt

# Recursive listing
ls -R

# List only directories
ls -d */
```

### mkdir (Make Directory)

**Theory:** Creates new directories. Can create parent directories automatically with recursive option.

```
# Create single directory
mkdir my_folder

# Create multiple directories
mkdir dir1 dir2 dir3

# Create nested directories
mkdir -p /path/to/nested/directories

# Create with specific permissions
mkdir -m 755 my_folder
```

### cp (Copy)

**Theory:** Copies files or directories from source to destination. Preserves original files while creating duplicates.

```
# Copy single file
cp source.txt destination.txt

# Copy with verbose output
cp -v source.txt destination.txt

# Copy directory recursively
cp -r /source/dir /destination/dir

# Copy preserving attributes
cp -p source.txt destination.txt

# Copy with confirmation for overwrites
cp -i source.txt destination.txt
```

### mv (Move)

**Theory:** Moves or renames files and directories. Works both within the same directory (rename) and across directories (move).

```
# Rename file
mv old_name.txt new_name.txt

# Move file to different directory
mv file.txt /path/to/destination/
```

```
# Move directory
mv old_directory new_directory

# Move with confirmation
mv -i source.txt destination/
```

### rm (Remove)

**Theory:** Deletes files and directories. Use with caution as deletion is permanent without recovery options in most cases.

```
# Remove single file
rm file.txt

# Remove multiple files
rm file1.txt file2.txt file3.txt

# Remove directory and contents recursively
rm -r my_directory

# Force removal without confirmation
rm -f file.txt

# Remove with verbose output
rm -v file.txt

# Remove directory recursively with confirmation
rm -ri my_directory
```

### touch (Create Empty File)

**Theory:** Creates empty files or updates file access and modification timestamps. Useful for creating placeholder files.

```
# Create new empty file
touch new_file.txt

# Create multiple files
touch file1.txt file2.txt file3.txt

# Update timestamps without creating if exists
touch -c existing_file.txt

# Set specific access time
touch -a file.txt
```

---

## File Viewing and Editing

### cat (Concatenate and Display)

**Theory:** Displays file contents, concatenates multiple files, and can create new files. Simple and straightforward viewing tool.

```
# Display file contents
cat file.txt

# Display multiple files
cat file1.txt file2.txt

# Display with line numbers
cat -n file.txt
```

```
# Display non-printable characters
cat -A file.txt

# Create file with cat
cat > new_file.txt << EOF
This is line 1
This is line 2
EOF
```

### more and less (Paged File Viewing)

**Theory:** Allows viewing large files one page at a time. `less` is more powerful with backward navigation capabilities.

```
# View file with more
more large_file.txt

# View file with less (recommended)
less large_file.txt

# Jump to specific line
less +100 file.txt

# Search within less
# Press '/' then type search term
```

### head and tail (View File Portions)

**Theory:** Display the beginning (head) or end (tail) of files. Extremely useful for log files and large datasets.

```
# Display first 10 lines (default)
head file.txt

# Display first 20 lines
head -n 20 file.txt

# Display last 10 lines
tail file.txt

# Display last 20 lines
tail -n 20 file.txt

# Follow file as new content is added (useful for logs)
tail -f /var/log/syslog

# Display lines 10-20
head -n 20 file.txt | tail -n 11
```

### nano and vim (Text Editors)

**Theory:** Command-line text editors for creating and editing files. `nano` is beginner-friendly; `vim` is powerful but has a steep learning curve.

```
# Open file with nano (easy to use)
nano file.txt

# Open file with vim (advanced)
vim file.txt

# Create new file with nano
nano new_file.txt

# Vim: Press 'i' to enter insert mode, 'ESC' to exit, ':wq' to save
and quit
```

### grep (Search Text)

**Theory:** Searches for patterns in files or text streams. Fundamental tool for text processing and log analysis.

```
# Search for pattern in file
grep "pattern" file.txt

# Case-insensitive search
grep -i "pattern" file.txt

# Show line numbers
grep -n "pattern" file.txt

# Invert match (show non-matching lines)
grep -v "pattern" file.txt

# Recursive search in directory
grep -r "pattern" /path/to/directory

# Count matching lines
grep -c "pattern" file.txt

# Show context around matches
grep -B 2 -A 2 "pattern" file.txt
```

---

## Text Processing

### sed (Stream Editor)

**Theory:** Performs text transformations on input streams or files. Powerful for substitutions, deletions, and pattern-based edits.

```
# Simple substitution
sed 's/old/new/' file.txt

# Global substitution (replace all occurrences)
sed 's/old/new/g' file.txt

# In-place editing
sed -i 's/old/new/g' file.txt

# Delete lines matching pattern
sed '/pattern/d' file.txt

# Delete specific line number
sed '5d' file.txt

# Print specific lines
sed -n '5,10p' file.txt
```

### awk (Text Processing Language)

**Theory:** Powerful tool for processing and analyzing structured text and data. Works line-by-line and supports field-based operations.

```
# Print specific fields
awk '{print $1, $3}' file.txt

# Print lines with specific pattern
awk '/pattern/' file.txt

# Print specific field from CSV
```

```
awk -F',' '{print $2}' data.csv

# Sum values in a column
awk '{sum += $1} END {print sum}' numbers.txt

# Print lines with field matching condition
awk '$3 > 100' file.txt

# Custom field separator
awk -F':' '{print $1}' /etc/passwd
```

### sort (Sort Lines)

**Theory:** Sorts lines of text files. Supports numeric sorting, reverse sorting, and key-based sorting.

```
# Simple sort alphabetically
sort file.txt

# Reverse sort
sort -r file.txt

# Numeric sort
sort -n numbers.txt

# Sort by specific column
sort -k 2 file.txt

# Sort and remove duplicates
sort -u file.txt

# Case-insensitive sort
sort -f file.txt
```

### uniq (Remove Duplicates)

**Theory:** Removes or counts duplicate adjacent lines. Usually used with `sort` for effective duplicate removal.

```
# Remove duplicate lines (must be sorted)
sort file.txt | uniq

# Count occurrences of each line
sort file.txt | uniq -c

# Show only duplicated lines
sort file.txt | uniq -d

# Show only unique lines
sort file.txt | uniq -u
```

### cut (Extract Columns)

**Theory:** Extracts specific columns or fields from each line of input. Works with delimiters for structured data.

```
# Extract specific characters
cut -c 1-5 file.txt

# Extract fields by delimiter
cut -d',' -f 1,3 data.csv

# Extract fields with space delimiter
cut -d' ' -f 2 file.txt
```

```
# Exclude specific fields
cut -d',' -f 1-2,4- data.csv
```

---

# File Permissions and Ownership

## chmod (Change Permissions)

**Theory:** Modifies file and directory permissions using either symbolic (u/g/o/a ±rwx) or octal (0-7) notation.

```
# Symbolic notation: add executable for owner
chmod u+x script.sh

# Symbolic: remove write for others
chmod o-w file.txt

# Symbolic: set specific permissions
chmod u=rwx,g=rx,o=r file.txt

# Octal notation (4=r, 2=w, 1=x)
# 755 = rwxr-xr-x (owner full, others read+execute)
chmod 755 script.sh

# Octal: 644 = rw-r--r-- (common for files)
chmod 644 file.txt

# Recursive chmod for directories
chmod -R 755 /path/to/directory
```

## chown (Change Ownership)

**Theory:** Changes file and directory owner and/or group. Requires appropriate privileges.

```
# Change owner
chown new_owner file.txt

# Change owner and group
chown new_owner:new_group file.txt

# Recursive change
chown -R new_owner:new_group /path/to/directory

# Change only group
chown :new_group file.txt
```

## chgrp (Change Group)

**Theory:** Changes only the group ownership of files. Simpler than chown when only group needs modification.

```
# Change group
chgrp new_group file.txt

# Recursive group change
chgrp -R new_group /path/to/directory
```

## umask (Set Default Permissions)

**Theory:** Sets default permissions for newly created files and directories. Mask value is subtracted from base permissions.

```
# Display current umask
umask

# Set umask (common: 022 removes write for group/others)
umask 022

# Apply umask in shell startup file
echo "umask 077" >> ~/.bashrc
```

---

# System Information

## uname (System Information)

**Theory:** Displays system information including kernel name, version, hardware platform, and processor type.

```
# All system information
uname -a

# Kernel name only
uname -s

# Kernel release
uname -r

# Hardware platform
uname -m

# Processor information
uname -p
```

## hostnamectl (Hostname Management)

**Theory:** Manages hostname and deployment environment information. Modern replacement for older hostname command.

```
# Display current hostname
hostnamectl

# Set new hostname
sudo hostnamectl set-hostname new-hostname

# Set hostname and pretty hostname
sudo hostnamectl set-hostname new-hostname --pretty
```

## whoami (Current User)

**Theory:** Displays the current logged-in user. Useful in scripts to determine execution context.

```
# Display current user
whoami

# Display user ID
id

# Display full user information
id -u
```

## df (Disk Space Usage)

**Theory:** Reports file system disk space usage. Essential for monitoring storage capacity.

```
# Display disk usage
df

# Human-readable format
df -h

# Show file system type
df -T

# Display in KB
df -k

# Exclude specific file systems
df -h --exclude-type=tmpfs
```

## du (Directory Usage)

**Theory:** Estimates directory space usage recursively. Helps identify large directories consuming disk space.

```
# Directory space usage
du /path/to/directory

# Human-readable format
du -h /path/to/directory

# Summary only (total)
du -sh /path/to/directory

# Show per-directory usage
du -h /path/to/directory/*

# Limit depth of recursion
du -h --max-depth=1 /path/to/directory
```

## free (Memory Usage)

**Theory:** Displays RAM and swap memory usage. Critical for understanding system memory availability.

```
# Display memory usage
free

# Human-readable format
free -h

# Display in MB
free -m

# Display in GB
free -g

# Continuous monitoring (every 2 seconds)
free -h -s 2
```

## uptime (System Uptime)

**Theory:** Shows how long the system has been running and current load average.

```
# Display uptime and load average
uptime
```

```
# Just uptime
uptime -p

# Pretty format
uptime -p
```

### date (System Date and Time)

**Theory:** Displays or sets system date and time. Essential for logging and time-based operations.

```
# Display current date and time
date

# Display in specific format
date '+%Y-%m-%d %H:%M:%S'

# Display date only
date '+%Y-%m-%d'

# Display time only
date '+%H:%M:%S'

# Set system time (requires sudo)
sudo date -s "2025-12-24 14:21:18"
```

---

## Process Management

### ps (Process Status)

**Theory:** Lists running processes with various information including PID, status, and resource usage.

```
# List all processes
ps aux

# List processes for current user
ps

# Show process tree
ps -ef --forest

# Show specific process
ps aux | grep process_name

# Show detailed information
ps -ef

# Show processes with custom columns
ps aux --sort=-%cpu
```

### top (Interactive Process Monitor)

**Theory:** Real-time system monitoring showing CPU, memory, and process information. Interactive interface with sorting capabilities.

```
# Launch top
top

# Batch mode (non-interactive)
top -b -n 1
```

```
# Show specific user processes
top -u username

# Sort by CPU usage
top -o %CPU

# Sort by memory usage
top -o %MEM

# Exit top by pressing 'q'
```

## kill (Terminate Process)

**Theory:** Sends signals to processes. Signal 15 (TERM) is graceful; signal 9 (KILL) is forceful but risky.

```
# Kill by PID (graceful)
kill 1234

# Force kill
kill -9 1234

# Kill by process name (using pkill)
pkill process_name

# Kill all instances of a process
pkill -9 process_name

# List available signals
kill -l
```

## pkill (Process Kill by Name)

**Theory:** Kills processes by name or pattern. More convenient than ps | grep | kill pipeline.

```
# Kill process by name
pkill process_name

# Force kill
pkill -9 process_name

# Kill with pattern matching
pkill -f "python script.py"

# Kill processes for specific user
pkill -u username
```

## bg and fg (Background and Foreground)

**Theory:** Manages job control. Move processes between foreground and background execution.

```
# Send running process to background
# Press Ctrl+Z then type: bg

# Bring background job to foreground
fg

# List background jobs
jobs

# Bring specific job to foreground
fg %1
```

```
# Continue stopped job in background
bg %1
```

### nohup (No Hangup)

**Theory:** Runs processes immune to terminal disconnection. Essential for long-running remote processes.

```
# Run command and ignore hangup signal
nohup long_running_command &

# Redirect output to file
nohup long_running_command > output.log 2>&1 &

# View nohup output
cat nohup.out
```

---

## Networking Commands

### ping (Test Connectivity)

**Theory:** Tests connectivity to remote hosts using ICMP packets. Verifies network accessibility and latency.

```
# Ping remote host
ping google.com

# Send specific number of packets
ping -c 5 google.com

# Set packet size
ping -s 56 google.com

# Set timeout (Linux)
ping -w 5000 google.com
```

### ifconfig (Interface Configuration)

**Theory:** Displays or configures network interfaces. Shows IP addresses, MAC addresses, and network statistics.

```
# Display all interfaces
ifconfig

# Display specific interface
ifconfig eth0

# Assign IP address
sudo ifconfig eth0 192.168.1.100

# Enable interface
sudo ifconfig eth0 up

# Disable interface
sudo ifconfig eth0 down
```

### ip (IP Configuration)

**Theory:** Modern replacement for ifconfig. More powerful tool for network configuration and management.

```
# Display all interfaces and addresses
```

```
ip addr show

# Display specific interface
ip addr show eth0

# Add IP address
sudo ip addr add 192.168.1.100/24 dev eth0

# Remove IP address
sudo ip addr del 192.168.1.100/24 dev eth0

# Show routing table
ip route show

# Add route
sudo ip route add 192.168.2.0/24 via 192.168.1.1
```

### netstat (Network Statistics)

**Theory:** Displays network connections, routing tables, and interface statistics. Essential for troubleshooting network issues.

```
# List all connections
netstat -a

# List listening ports
netstat -l

# List TCP connections only
netstat -t

# List UDP connections only
netstat -u

# Show process IDs and names
netstat -p

# Show continuous updates
netstat -c

# Show statistics
netstat -s
```

### ss (Socket Statistics)

**Theory:** Modern replacement for netstat. More efficient tool for viewing socket statistics and connections.

```
# Show all sockets
ss -a

# Show listening sockets
ss -l

# Show TCP sockets
ss -t

# Show UDP sockets
ss -u

# Show with process information
ss -p

# Show statistics
ss -s
```

```
# Show detailed socket information
ss -ed
```

### nslookup and dig (DNS Lookup)

**Theory:** Query DNS nameservers to resolve domain names. Essential for DNS troubleshooting and verification.

```
# Simple DNS lookup
nslookup google.com

# Query specific nameserver
nslookup google.com 8.8.8.8

# Reverse DNS lookup
nslookup 8.8.8.8

# Detailed DNS query
dig google.com

# Query specific record type
dig google.com MX

# Short answer only
dig +short google.com
```

### traceroute (Trace Network Path)

**Theory:** Shows the route packets take to reach a destination. Useful for identifying network path issues.

```
# Trace route to host
traceroute google.com

# Trace with ICMP packets
traceroute -I google.com

# Set maximum hops
traceroute -m 20 google.com

# Set packet size
traceroute -s 56 google.com
```

### wget and curl (Download Files)

**Theory:** Download files from the internet using HTTP/HTTPS protocols. `wget` is simpler; `curl` is more flexible.

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

# Download and save with specific name
wget -O custom_name.zip https://example.com/file.zip

# Resume interrupted download
wget -c https://example.com/large_file.zip

# Download file with curl
curl -O https://example.com/file.zip

# Download and save with specific name
curl -o custom_name.zip https://example.com/file.zip

# Show headers only
curl -I https://example.com
```

---

# User and Group Management

## useradd (Add User)

**Theory:** Creates new user accounts. Requires root privileges and follows system-wide user naming conventions.

```
# Create basic user
sudo useradd new_user

# Create user with home directory
sudo useradd -m new_user

# Create user with specific shell
sudo useradd -m -s /bin/bash new_user

# Create user with specific UID
sudo useradd -m -u 1000 new_user

# Create user and add to group
sudo useradd -m -G sudo new_user
```

## usermod (Modify User)

**Theory:** Modifies existing user account properties including groups, shell, and home directory.

```
# Add user to group
sudo usermod -aG groupname username

# Change user shell
sudo usermod -s /bin/bash username

# Change home directory
sudo usermod -d /new/home username

# Lock user account
sudo usermod -L username

# Unlock user account
sudo usermod -U username
```

## userdel (Delete User)

**Theory:** Removes user accounts. Use caution as this operation is permanent.

```
# Remove user (keep home directory)
sudo userdel username

# Remove user and home directory
sudo userdel -r username

# Remove user and all files
sudo userdel -rf username
```

## passwd (Change Password)

**Theory:** Changes user password. Can set expiry and complexity requirements.

```
# Change current user password
```

```
passwd

# Change another user's password (root only)
sudo passwd username

# Force password change on next login
sudo passwd -e username

# Lock account
sudo passwd -l username

# Unlock account
sudo passwd -u username
```

### groupadd (Add Group)

**Theory:** Creates new user groups. Used to organize users with shared permissions.

```
# Create basic group
sudo groupadd groupname

# Create group with specific GID
sudo groupadd -g 1000 groupname
```

### groupdel (Delete Group)

**Theory:** Removes user groups. Group must not be primary group of any user.

```
# Delete group
sudo groupdel groupname
```

### sudo (Execute as Superuser)

**Theory:** Executes commands with superuser privileges. Primary method for privilege escalation in Linux.

```
# Execute command as root
sudo command

# Execute as specific user
sudo -u username command

# Run interactive shell as root
sudo -i

# Run shell as specific user
sudo -u username -i

# List sudo privileges
sudo -l
```

---

## Package Management

### apt (Debian/Ubuntu)

**Theory:** Advanced Package Tool for Debian-based systems. Manages software packages with dependency resolution.

```
# Update package lists
sudo apt update
```

```
# Upgrade all packages
sudo apt upgrade

# Install package
sudo apt install package_name

# Remove package
sudo apt remove package_name

# Remove package and configuration
sudo apt purge package_name

# Search for package
apt search package_name

# Show package information
apt show package_name

# List installed packages
apt list --installed
```

### yum/dnf (RedHat/CentOS/Fedora)

**Theory:** Package managers for RedHat-based systems. `dnf` is the modern replacement for `yum`.

```
# Update package lists
sudo dnf check-update

# Install package
sudo dnf install package_name

# Remove package
sudo dnf remove package_name

# Search for package
dnf search package_name

# Show package information
dnf info package_name

# List installed packages
dnf list installed

# Update all packages
sudo dnf upgrade
```

### snap (Universal Package Manager)

**Theory:** Universal package manager working across different Linux distributions. Self-contained application bundles.

```
# Install snap package
sudo snap install package_name

# Remove snap package
sudo snap remove package_name

# List installed snaps
snap list

# Update snap package
sudo snap refresh package_name

# Update all snaps
```

```
sudo snap refresh
```

---

# Archive and Compression

## tar (Tape Archive)

**Theory:** Creates and extracts archive files. Fundamental tool for file distribution and backups.

```
# Create tar archive
tar -cvf archive.tar file1 file2 directory

# List tar contents
tar -tvf archive.tar

# Extract tar archive
tar -xvf archive.tar

# Extract to specific directory
tar -xvf archive.tar -C /path/to/destination

# Create compressed tar (gzip)
tar -czvf archive.tar.gz file1 directory

# Extract tar.gz
tar -xzvf archive.tar.gz

# Create compressed tar (bzip2)
tar -cjvf archive.tar.bz2 file1 directory

# Extract tar.bz2
tar -xjvf archive.tar.bz2
```

## gzip (GNU Zip)

**Theory:** Compresses and decompresses files using gzip format. Efficient compression for single files.

```
# Compress file
gzip file.txt

# Decompress file
gunzip file.txt.gz

# Keep original file
gzip -k file.txt

# Show compression ratio
gzip -v file.txt

# Set compression level (1-9)
gzip -9 file.txt
```

## zip and unzip

**Theory:** Creates and extracts ZIP archives. Widely compatible across different operating systems.

```
# Create zip archive
zip archive.zip file1.txt file2.txt

# Create zip recursively
zip -r archive.zip directory
```

```
# Extract zip file
unzip archive.zip

# Extract to specific directory
unzip archive.zip -d /path/to/destination

# List zip contents
unzip -l archive.zip

# Test zip integrity
unzip -t archive.zip
```

---

# Searching and Finding Files

## find (Find Files)

**Theory:** Searches for files and directories based on various criteria. Powerful tool with extensive filtering options.

```
# Find files in directory
find /path -name "filename"

# Case-insensitive search
find /path -iname "filename"

# Find by file type (f=file, d=directory, l=link)
find /path -type f -name "*.txt"

# Find by size
find /path -size +100M

# Find files modified in last 7 days
find /path -mtime -7

# Find empty files
find /path -type f -empty

# Execute command on found files
find /path -name "*.log" -delete

# Find with multiple conditions
find /path -type f -name "*.txt" -size +1M
```

## locate (Locate Files)

**Theory:** Uses database index for fast file location. Much faster than find but requires periodic database updates.

```
# Find file by name
locate filename

# Case-insensitive search
locate -i filename

# Count matching files
locate -c filename

# Update database
sudo updatedb

# Search in specific directory
locate -r "^/home.*\.conf$"
```

### which (Locate Command)

**Theory:** Shows the location of command executables in PATH. Useful for debugging command conflicts.

```
# Find command location
which python

# Find all instances in PATH
which -a python

# Show all matching executables
type python
```

### whereis (Locate Command Source)

**Theory:** Locates command binary, source, and manual pages. Provides comprehensive command location information.

```
# Find command and documentation
whereis python

# Find only binary
whereis -b python

# Find only source
whereis -s python

# Find only manual
whereis -m python
```

---

## System Monitoring

### watch (Monitor Changes)

**Theory:** Executes command repeatedly and displays output. Useful for monitoring system changes in real-time.

```
# Watch command output (default 2 seconds)
watch command

# Set custom interval (1 second)
watch -n 1 command

# Highlight changes
watch -d command

# Watch disk usage
watch df -h

# Watch process list
watch 'ps aux | grep process'
```

### systemctl (System Service Management)

**Theory:** Controls systemd services and system state. Essential for managing daemons and system services.

```
# Start service
sudo systemctl start service_name

# Stop service
sudo systemctl stop service_name
```

```
# Restart service
sudo systemctl restart service_name

# Enable service at boot
sudo systemctl enable service_name

# Disable service at boot
sudo systemctl disable service_name

# Check service status
sudo systemctl status service_name

# List all services
systemctl list-units --type=service

# View service logs
journalctl -u service_name
```

## journalctl (View System Logs)

**Theory:** Queries and displays systemd journal logs. Modern logging system with powerful filtering.

```
# Display recent logs
journalctl

# Follow live logs
journalctl -f

# Show logs for specific service
journalctl -u service_name

# Show logs since time
journalctl --since "2025-12-24 14:00:00"

# Show logs until time
journalctl --until "2025-12-24 15:00:00"

# Show last 100 lines
journalctl -n 100

# Show logs with priority ERROR or higher
journalctl -p err
```

## iostat (I/O Statistics)

**Theory:** Reports CPU and input/output statistics. Essential for identifying disk and CPU bottlenecks.

```
# Display I/O statistics
iostat

# Display with interval (every 2 seconds)
iostat 2

# Show extended statistics
iostat -x

# Show specific device
iostat -x sda

# Display CPU statistics
iostat -c
```

### iotop (I/O Top)

**Theory:** Real-time monitoring of disk I/O usage per process. Similar to top but for I/O operations.

```
# Monitor I/O usage
sudo iotop

# Non-interactive mode
sudo iotop -b

# Batch mode for 10 iterations
sudo iotop -b -n 10

# Monitor specific process
sudo iotop -p PID
```

---

## Additional Useful Commands

### history (Command History)

**Theory:** Shows previously executed commands. Helps recall and reuse past commands.

```
# Display command history
history

# Display last 20 commands
history 20

# Search history
history | grep command_name

# Clear history
history -c

# Clear specific line
history -d 100
```

### alias (Create Command Alias)

**Theory:** Creates shortcuts for frequently used commands. Improves efficiency and reduces typing.

```
# Create alias
alias ll='ls -lah'

# Create temporary alias
alias gs='git status'

# List all aliases
alias

# Remove alias
unalias ll

# Make alias permanent
echo "alias ll='ls -lah'" >> ~/.bashrc
```

### echo (Print Text)

**Theory:** Outputs text to standard output. Fundamental tool for displaying messages and variables.

```bash
# Print simple text
echo "Hello World"

# Print variable
echo $VARIABLE

# Print with newline
echo -e "Line1\nLine2"

# Print without newline
echo -n "Text"

# Print to file
echo "Text" > file.txt

# Append to file
echo "Text" >> file.txt
```

### man (Manual Pages)

**Theory:** Displays system documentation. Essential resource for learning command options and syntax.

```bash
# Display command manual
man command_name

# Search in manual
man -k keyword

# Display specific section
man 1 command_name

# View man page for library function
man 3 function_name

# List manual sections
man man
```

---

## Summary

This cheat sheet covers the most essential Linux commands for system administration, DevOps work, and development. Regular practice with these commands will build proficiency in Linux command-line operations.

**Key Tips:** - Always use `man` for detailed command documentation - Practice in a safe environment before running potentially destructive commands - Use `--help` flag for quick command option reference - Combine commands using pipes (`|`) for powerful operations - Use tab completion to speed up command typing

---

**Last Updated:** 2025-12-24 **Maintainer:** laxmandudhate