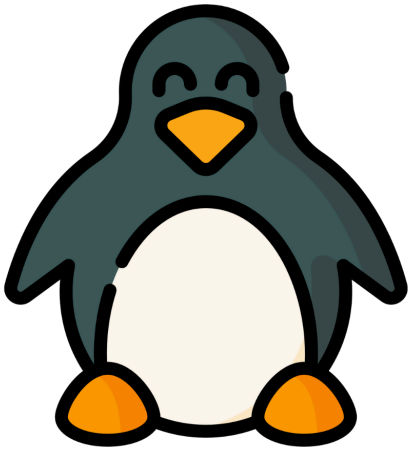
TECHWORLD WITH NANA



Essential Linux Commands Cheatsheet

by

TECHWORLD WITH NANA

I'm Nana, Co-Founder of TechWorld with Nana.

As a DevOps engineer and educator, I'm passionate about helping engineers build efficient, streamlined workflows and become more valuable with highly-demanded DevOps and Cloud skills

Through my YouTube channel and my comprehensive DevOps bootcamps, **I've helped 1,000,000s of engineers** master the tools and practices that drive modern software development.

About this Cheat Sheet

This Linux Commands Cheatsheet is your **go-to reference for navigating the Linux command line** with confidence.

Whether you're just starting your Linux journey or looking to refresh your command line knowledge, this resource provides the commands you need for everyday tasks and server administration.

Linux proficiency is a foundational skill for any IT professional. By mastering these commands, you'll work more efficiently in terminal environments, automate repetitive tasks, and build a solid foundation for more advanced technical skills in DevOps and Cloud engineering.

Keep this cheatsheet handy during your learning journey and refer to it whenever you need a quick reference. Remember: *mastering the command line is an investment that pays off throughout your entire tech career!*

Happy learning!

Nana Janashia

Co-Founder TechWorld with Nana

Navigation & File Management

`pwd` - Print working directory

`ls` - List files and directories
- `ls -l` - Long format
- `ls -a` - Show hidden files
- `ls -h` - Human-readable sizes

`cd [directory]` - Change directory
- `cd ..` - Go up one level
- `cd ~` - Go to home directory
- `cd -` - Go to previous directory

`mkdir [dirname]` - Create directory

`rm [filename]` - Remove file
- `rm -r` - Remove directory and contents
- `rm -f` - Force remove without confirmation

`cp [source] [dest]` - Copy files

`mv [source] [dest]` - Move/rename files

File Operations

`cat [filename]` - Display file contents

`less [filename]` - View with pagination

`head [filename]` - Show first 10 lines

`tail [filename]` - Show last 10 lines
- `tail -f` - Follow file updates in real-time

`grep [pattern] [file]` - Search for pattern
- `grep -i` - Case insensitive
- `grep -r` - Recursive search

`diff [file1] [file2]` - Compare files

`wc [file]` - Count lines, words, bytes
- `wc -l` - Prints the number of lines in file
- `wc -w` - Prints the number of words in file
- `wc -m` - Prints the count of characters

`find` - Search files with various options

Common options: `type` , `name` , `size`

Example:

`find . -name "*.log"` - Starting from the current directory, search through all subdirectories and find any files that have a name ending with '.log'

File Permissions

Format: `rwx rwx rwx` (owner group others)

- `r` (4): Read permission
- `w` (2): Write permission
- `x` (1): Execute permission

Common examples:

- `chmod 755` - Owner full access, others read/execute
- `chmod 644` - Owner read/write, others read only
- `chmod 700` - Owner full access, others none

Vim Text Editor

Normal Mode (default):
`h,j,k,l` - Move cursor
`dd` - Delete line
`yy` - Copy line
`p` - Paste

Insert Mode (press `i`):
Type to insert text
`Esc` - Return to Normal mode

Command Mode (press `:`):
`:w` - Save
`:q` - Quit
`:wq` - Save and quit

TECHWORLD WITH NANA

Piping & Redirection

`|` - Pipe output to next command

`> [file]` - Redirect output (overwrite)

`>> [file]` - Append output

`2> [file]` - Redirect errors

Examples:

`ls -l | grep "\.txt"` - Shows detailed listings of only .txt files

`command > out.txt 2> errors.txt` - Saves normal output to one file and error messages to another file

System & Process

`ps` - Show processes

- `ps -ef` - All processes in full format
- `ps aux` - Detailed process list

`top` - Real-time process monitor

`kill pid` - Terminate process

`df -h` - Show disk space

`free -h` - Show memory usage

SSH & Remote Access

`ssh user@hostname` - Connect to server

`ssh-keygen` - Generate SSH key pair

`ssh-copy-id user@host` - Copy SSH key

`scp file user@host:path` - Copy file

Command History

`history` - Show history

`!!` - Execute last command

`Ctrl + R` - Search history

Package Management

`apt update` - Update package list

`apt install pkg` - Install package

`apt remove pkg` - Remove package

`apt upgrade` - Upgrade packages

Root Access

`sudo command` - Run as root

`sudo su` - Become root

`passwd` - Change password

Network Commands

`ping` - Test connectivity

Example:
`ping -c 4 google.com` - Tests the connectivity between your computer and Google's servers. `-c 4` limits the test to 4 packets

`curl` - Transfer data from/to server

Example:
`curl -I https://example.com` - Checks if a website is up without downloading the entire page content. `-I` tells curl to send a HEAD request instead of the default GET request

`netstat` - Displays network statistics

Example:
`netstat -tuln`
`-t` shows TCP connections
`-u` shows UDP connections
`-l` displays only listening sockets (servers)
`-n` shows numerical addresses instead of resolving hostnames and port numbers

System Information

`uname` - Show system information

`who` - Show who is logged in

`hostname` - Show or set hostname

`free` - Display memory usage

Chaining Commands - The Real Power

Linux commands become incredibly powerful when you chain them together. Each simple command can pipe its output to the next, creating complex workflows from basic building blocks.

Here's how to combine commands to solve real-world problems.

1. Find All Text Files

```
find . -type f -name "*.txt"
```

Search your current directory and all subdirectories for files ending in .txt. The dot means 'start here', -type f means 'files only', and -name specifies the pattern.

2. Find and Read All Text Files

```
find . -type f -name "*.txt" | xargs cat
```

Now we pipe those filenames to xargs, which runs cat on each file. This displays all the contents of all .txt files one after another.

3. Find Error Lines in All Text Files

```
find . -type f -name "*.txt" | xargs cat | grep "ERROR"
```

Add grep to search through all that content and show only lines containing ERROR. Great for scanning log files to find problems.

4. Sort and Remove Duplicate Errors

```
find . -type f -name "*.txt" | xargs cat | grep "ERROR" | sort -k4
| uniq -f3
```

Now we organize the results:

- *sort -k4 sorts lines by the 4th column (useful if timestamps are there)*
- *uniq -f3 removes duplicate lines while skipping the first 3 fields (so you see each unique error type once)*

5. Save Results to a File

```
find . -type f -name "*.txt" | xargs cat | grep "ERROR" | sort -k4 |
uniq -f3 > errors.log
```

The > symbol redirects all that output into a file called errors.log instead of showing it on screen. Now you have a clean report of unique errors.

6. Copy Log Files to Backup Folder

```
find . -type f -name "*.log" -exec cp {} ./backup \;
```

Uses -exec to run a command on each file found. The {} gets replaced with each filename, and \; marks the end of the command. This copies all .log files to your backup folder.

7. Backup Files with Directory Structure

```
find . -type f -name "*.txt" -exec rsync -R {} ./backup \;
```

Similar to the previous command, but rsync with -R flag preserves the directory structure. So if you have logs/app/error.txt, it creates backup/logs/app/error.txt instead of dumping everything in one folder.