

Shell Cheatsheet

Essential commands for command-line navigation and automation

This cheatsheet provides a comprehensive reference to fundamental Shell/Bash commands, syntax, and scripting techniques, ideal for both beginners and experienced users for efficient system administration and automation.

File Operations	Navigation	Text Processing
Create, copy, move and manage files	Navigate directories and file systems	Search, filter and manipulate text
Process Management	System Information	
Control and monitor system processes	View system status and resources	

File & Directory Operations

List Files: `ls`

Display files and directories in the current location.

```
# List files in current directory
ls
# List with detailed information
ls -l
# Show hidden files
ls -a
# List with human-readable file sizes
ls -h
# Sort by modification time
ls -lt
```

Create Files: `touch`

Create empty files or update timestamps.

```
# Create a new file
touch newfile.txt
# Create multiple files
touch file1.txt file2.txt file3.txt
# Update timestamp of existing file
touch existing_file.txt
```

Create Directories: `mkdir`

Create new directories.

```
# Create a directory
mkdir my_directory
# Create nested directories
mkdir -p parent/child/grandchild
# Create multiple directories
mkdir dir1 dir2 dir3
```

Navigation & Path Management

Navigate through the file system efficiently.

01	02	03
Current Directory: `pwd` Print the current working directory path.	Change Directory: `cd` Change to a different directory.	Directory Tree: `tree` Display directory structure in tree format.
<pre># Show current directory pwd # Example output: /home/user/documents</pre>	<pre># Go to home directory cd ~ # Go to parent directory cd .. # Go to previous directory cd - # Go to specific directory cd /path/to/directory</pre>	<pre># Show directory tree tree # Limit depth to 2 levels tree -L 2 # Show only directories tree -d</pre>

Text Processing & Search

View Files: `cat` / `less` / `head` / `tail`

Display file contents in different ways.

```
# Display entire file
cat file.txt
# View file page by page
less file.txt
# Show first 10 lines
head file.txt
# Show last 10 lines
tail file.txt
# Show last 20 lines
tail -n 20 file.txt
# Follow file changes (useful for logs)
tail -f logfile.txt
```

Search in Files: `grep`

Search for patterns in text files.

```
# Search for pattern in file
grep "pattern" file.txt
# Case-insensitive search
grep -i "pattern" file.txt
# Search recursively in directories
grep -r "pattern" directory/
# Show line numbers
grep -n "pattern" file.txt
# Count matching lines
grep -c "pattern" file.txt
```

File Permissions & Ownership

View Permissions: `ls -l`

Display detailed file permissions and ownership.

```
# Show detailed file information
ls -l
# Example output:
# -rw-r--r-- 1 user group 1024 Jan 1 12:00 file.txt
# d = directory, r = read, w = write, x = execute
```

Change Permissions: `chmod`

Modify file and directory permissions.

```
# Give execute permission to owner
chmod +x script.sh
# Set specific permissions (755)
chmod 755 file.txt
# Remove write permission for group/others
chmod go-w file.txt
# Recursive permission change
chmod -R 644 directory/
```

Process Management

Monitor and control running processes.

View Processes: `ps` / `top` / `htop`

Display information about running processes.

```
# Show processes for current user
ps
# Show all processes with details
ps aux
# Show processes in tree format
ps -ef --forest
# Interactive process viewer
top
# Enhanced process viewer (if available)
htop
```

Background Jobs: `&` / `jobs` / `fg` / `bg`

Manage background and foreground processes.

```
# Run command in background
command &
# List active jobs
jobs
# Bring job to foreground
fg %1
# Send job to background
bg %1
# Suspend current process
Ctrl+Z
```

Input/Output Redirection

Control command input, output, and data flow.

Redirection: `>` / `>>` / `<`

Redirect command output and input.

```
# Redirect output to file (overwrite)
command > output.txt
# Append output to file
command >> output.txt
# Redirect input from file
command < input.txt
# Redirect both output and errors
command > output.txt 2>&1
# Discard output
command >/dev/null
```

Pipes: `|`

Chain commands together using pipes.

```
# Basic pipe usage
command1 | command2
# Multiple pipes
cat file.txt | grep "pattern" | sort | uniq
# Count lines in output
ps aux | wc -l
# Page through long output
ps aux | less
```

Variables & Environment

Work with shell variables and environment settings.

Variables: Assignment & Usage

Create and use shell variables.

```
# Assign variables (no spaces around =)
name="John"
count=42

# Use variables
echo $name
echo "$Hello, $name"
echo "Count: ${count}"

# Command substitution
current_dir=$(pwd)
date_today=$(date +%Y-%m-%d)
```

Environment Variables: `export` / `env`

Manage environment variables.

```
# Export variable to environment
export PATH="/new/path:$PATH"
export MY_VAR="value"

# View all environment variables
env
# View specific variable
echo $HOME
echo $PATH
# Unset variable
unset MY_VAR
```

Scripting Basics

Write and execute shell scripts for automation.

Script Structure

Basic script format and execution.

```
#!/bin/bash
# This is a comment
# Variables
greeting="Hello, World!"
user=$(whoami)

# Output
echo $greeting
echo "Current user: $user"

# Make script executable
chmod +x script.sh
# Run script:
./script.sh
```

Conditional Statements: `if`

Control script flow with conditions.

```
#!/bin/bash
if [ -f file.txt ]; then
    echo "file exists"
elif [ -d "directory" ]; then
    echo "Directory exists"
else
    echo "Neither exists"
fi

# String comparison
if [ "$USER" == "root" ]; then
    echo "Running as root"
fi

# Numeric comparison
if [ $count -gt 10 ]; then
    echo "Count is greater than 10"
fi
```

Network & System Commands

Network diagnostics and system information commands.

Network Commands

Test connectivity and network configuration.

```
# Test network connectivity
ping google.com
ping -c 4 google.com # Send only 4 packets

# DNS lookup
nslookup google.com
dig google.com

# Network configuration
ip addr show # Show IP addresses
ip route show # Show routing table

# Download files
wget https://example.com/file.txt
curl -O https://example.com/file.txt
```

System Information: `uname` / `whoami` / `date`

Get system and user information.

```
# System information
uname -a # All system info
uname -r # Kernel version
hostname # Computer name
whoami # Current username
id # User ID and groups

# Date and time
date # Current date/time
date +%Y-%m-%d # Custom format
uptime # System uptime
```

Command History & Shortcuts

Navigate command history and use keyboard shortcuts efficiently.

Command History: `history`

View and reuse previous commands.

```
# Show command history
history
# Show last 10 commands
history 10
# Execute command by number
!123
# Execute last command starting with 'ls'
!ls
# Search history interactively
Ctrl+R
```

History Expansion

Reuse parts of previous commands.

```
# Last command's arguments
! # Last argument of previous command
!^ # First argument of previous command
!* # All arguments of previous command

# Example usage:
ls /very/long/path/to/file.txt
cd !$ # Goes to /very/long/path/to/file.txt
```

Command Combinations & Tips

Advanced command combinations and productivity tips.

Useful Command Combinations

Powerful one-liners for common tasks.

```
# Find and replace text in multiple files
find . -name "*.txt" -exec sed -i 's/old/new/g' {} \;
# Find largest files in current directory
du -ah | sort -rh | head -10
# Monitor log file for specific pattern
tail -f /var/log/syslog | grep "ERROR"
# Count files in directory
ls -1 | wc -l
# Create backup with timestamp
cp file.txt file.backup.$(date +%Y%m%d-%H%M%S)
# Redirect output to file (overwrite)
command > output.txt
# Append output to file
command >> output.txt
# Redirect input from file
command < input.txt
# Redirect both output and errors
command > output.txt 2>&1
# Discard output
command >/dev/null
```

Aliases & Functions

Create shortcuts for frequently used commands.

```
# Create aliases (add to ~/.bashrc)
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -C'
alias ...='cd ..'
alias ...='cd ./..'
alias ...='cd ././..'

# View all aliases
alias
```

Redirection: `>` / `>>` / `<`

Redirect command output and input.

```
# Redirect output to file (overwrite)
command > output.txt
# Append output to file
command >> output.txt
# Redirect input from file
command < input.txt
# Redirect both output and errors
command > output.txt 2>&1
# Discard output
command >/dev/null
```

Variables & Environment

Work with shell variables and environment settings.

Variables: Assignment & Usage

Create and use shell variables.

```
# Assign variables (no spaces around =)
name="John"
count=42

# Use variables
echo $name
echo "$Hello, $name"
echo "Count: ${count}"

# Command substitution
current_dir=$(pwd)
date_today=$(date +%Y-%m-%d)
```

Environment Variables: `export` / `env`

Manage environment variables.

```
# Export variable to environment
export PATH="/new/path:$PATH"
export MY_VAR="value"

# View all environment variables
env
# View specific variable
echo $HOME
echo $PATH
# Unset variable
unset MY_VAR
```

Scripting Basics

Write and execute shell scripts for automation.

Script Structure

Basic script format and execution.

```
#!/bin/bash
# This is a comment
# Variables
greeting="Hello, World!"
user=$(whoami)

# Output
echo $greeting
echo "Current user: $user"

# Make script executable
chmod +x script.sh
# Run script:
./script.sh
```

Conditional Statements: `if`

Control script flow with conditions.

```
#!/bin/bash
if [ -f file.txt ]; then
    echo "file exists"
elif [ -d "directory" ]; then
    echo "Directory exists"
else
    echo "Neither exists"
fi

# String comparison
if [ "$USER" == "root" ]; then
    echo "Running as root"
fi

# Numeric comparison
if [ $count -gt 10 ]; then
    echo "Count is greater than 10"
fi
```

Network & System Commands

Network diagnostics and system information commands.

Network Commands

Test connectivity and network configuration.

```
# Test network connectivity
ping google.com
ping -c 4 google.com # Send only 4 packets

# Network configuration
ip addr show # Show IP addresses
ip route show # Show routing table

# Download files
wget https://example.com/file.txt
curl -O https://example.com/file.txt
```

System Information: `uname` / `whoami` / `date`

Get system and user information.

```
# System information
uname -a # All system info
uname -r # Kernel version
hostname # Computer name
whoami # Current username
id # User ID and groups

# Date and time
date # Current date/time
date +%Y-%m-%d # Custom format
uptime # System uptime
```

Command History & Shortcuts

Navigate command history and use keyboard shortcuts efficiently.

Command History: `history`

View and reuse previous commands.

```
# Show command history
history
# Show last 10 commands
history 10
# Execute command by number
!123
# Execute last command starting with 'ls'
!ls
# Search history interactively
Ctrl+R
```

History Expansion

Reuse parts of previous commands.

```
# Last command's arguments
! # Last argument of previous command
!^ # First argument of previous command
!* # All arguments of previous command

# Example usage:
ls /very/long/path/to/file.txt
cd !$ # Goes to /very/long/path/to/file.txt
```

Input/Output Redirection

Control command input, output, and data flow.

Redirection: `>` / `>>` / `<`

Redirect command output and input.

```
# Redirect output to file (overwrite)
command > output.txt
# Append output to file
command >> output.txt
# Redirect input from file
command < input.txt
# Redirect both output and errors
command > output.txt 
```