Bash CLI Tools Cheat Sheet

Bash CLI Tools Cheat Sheet

A practical quick-reference guide for essential Unix/Linux command-line tools. Commands are organized by category with common flags and realistic examples.

find

Search for files and directories with powerful filtering options.

Basic Search

```
# Find by name (case-sensitive)
find /path -name "*.txt"

# Find by name (case-insensitive)
find /path -iname "*.TXT"

# Find files only
find /path -type f

# Find directories only
find /path -type d

# Find with max depth
find /path -maxdepth 3 -name "*.js"
```

Search by Size

```
# Find files larger than 100MB
find /path -size +100M

# Find files smaller than 1KB
find /path -size -1k

# Find files between 1MB and 10MB
find /path -size +1M -size -10M
```

Search by Time

```
# Modified in last 7 days
find /path -mtime -7

# Modified more than 30 days ago
find /path -mtime +30

# Accessed in last hour
find /path -amin -1
```

Execute Commands

```
# Delete found files
find /path -name "*.tmp" -delete

# Execute command on each file
find /path -name "*.log" -exec rm {} \;

# Execute with confirmation
find /path -name "*.bak" -ok -exec rm {} \;
```

Combine Conditions

```
# Find JS files modified in last week,find /path -name "*.js" -mtime -7

# Find large log files
find /path -name "*.log" -size +10M

# Find empty directories
find /path -type d -empty

# Find files by permission
find /path -perm 755
```

grep

Search text patterns in files.

Basic Search

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

# Search in multiple files
grep "pattern" *.txt

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

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

```
# Count matches
grep -c "pattern" file.txt
```

Recursive Search

```
# Search recursively
grep -r "pattern" /path

# Search with file pattern
grep -r --include="*.js" "pattern" /path

# Show matching files only
grep -rl "pattern" /path
```

Invert and Context

```
# Invert match (lines NOT matching)
grep -v "pattern" file.txt

# Show context (3 lines before/after)
grep -C 3 "pattern" file.txt

# Show 2 lines before
grep -B 2 "pattern" file.txt

# Show 2 lines after
grep -A 2 "pattern" file.txt
```

Extended Regex

```
# Extended regex
grep -E "pattern1|pattern2" file.txt

# Perl-compatible regex
grep -P "\d{3}-\d{3}-\d{4}" file.txt

# Match whole words only
grep -w "word" file.txt
```

Output Control

```
# Show only matched part
grep -o "pattern" file.txt

# Colorize output
grep --color=auto "pattern" file.txt

# Suppress errors
grep -s "pattern" /path/*
```

sed

Stream editor for text transformation.

Basic Substitution

```
# Replace first occurrence per line
sed 's/old/new/' file.txt

# Replace all occurrences
sed 's/old/new/g' file.txt

# Case-insensitive replace
sed 's/old/new/gi' file.txt
```

In-Place Editing

```
# Edit file in-place
sed -i 's/old/new/g' file.txt

# Create backup before editing
sed -i.bak 's/old/new/g' file.txt
```

Delete Lines

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

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

# Delete range of lines
sed '5,10d' file.txt

# Delete empty lines
sed '/^$/d' file.txt
```

Print Lines

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

# Print range
sed -n '5,10p' file.txt

# Print from line to end
sed -n '5,$p' file.txt
```

Multiple Commands

```
# Multiple substitutions
sed -e 's/old/new/g' -e 's/foo/bar/g' file.txt
```

```
# Using command file
sed -f commands.sed file.txt
```

awk

Text processing and data extraction.

Basic Usage

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

# Print all columns
awk '{print $0}' file.txt

# Print with custom separator
awk -F: '{print $1}' /etc/passwd
```

Conditions

```
# Filter by condition
awk '$3 > 100 {print $0}' file.txt

# String comparison
awk '$1 == "error" {print $0}' logfile.txt

# Multiple conditions
awk '$3 > 100 && $3 < 200 {print $0}' file.txt
```

Built-in Variables

```
# NR - line number
awk '{print NR, $0}' file.txt

# NF - number of fields
awk '{print NF}' file.txt

# FILENAME - current file
awk '{print FILENAME, NR}' *.txt

# FS/OFS - field separators
awk 'BEGIN{FS=","; OFS="\t"} {print $1, $2}' file.csv
```

BEGIN/END Blocks

```
# Header and footer
awk 'BEGIN {print "Header"} {print $0} END {print "Total:", NR}' file.txt

# Sum column
awk '{sum += $1} END {print sum}' file.txt
```

```
# Average
awk '{sum += $1; count++} END {print sum/count}' file.txt
```

xargs

Build and execute commands from standard input.

Basic Usage

```
# Execute command for each input
find . -name "*.txt" | xargs rm

# With placeholder
find . -name "*.txt" | xargs -I {} mv {} /backup/

# Limit arguments per command
find . -name "*.txt" | xargs -n 1 rm

# Parallel execution
find . -name "*.txt" | xargs -P 4 -n 1 command
```

Handle Special Characters

```
# Handle spaces in filenames
find . -name "*.txt" -print0 | xargs -0 rm

# Using null delimiter
find . -type f -print0 | xargs -0 ls -la
```

sort & uniq

Sort and deduplicate.

Sort

```
# Sort alphabetically
sort file.txt

# Sort numerically
sort -n numbers.txt

# Reverse sort
sort -r file.txt

# Sort by column
sort -k2 file.txt

# Sort by multiple keys
sort -k1,1 -k2,2n file.txt
```

```
# Unique sort
sort -u file.txt
```

Uniq

```
# Remove consecutive duplicates
uniq file.txt

# Show count of duplicates
uniq -c file.txt

# Show only duplicates
uniq -d file.txt

# Show only unique
uniq -u file.txt
```

cut & tr

Extract and transform text.

cut

```
# Extract columns by position
cut -c1-10 file.txt

# Extract by delimiter
cut -d: -f1,3 file.txt

# Extract multiple fields
cut -d, -f2-4 file.csv

# Complement (exclude)
cut -d: -f1 --complement file.txt
```

tr

```
# Convert to uppercase
tr 'a-z' 'A-Z' < file.txt

# Delete characters
tr -d '0-9' < file.txt

# Squeeze repeated characters
tr -s ' ' < file.txt

# Replace newline with space
tr '\n' ' ' < file.txt
```

head & tail & wc

View file parts and statistics.

head/tail

```
# First 10 lines
head file.txt

# First N lines
head -n 20 file.txt

# Last 10 lines
tail file.txt

# Last N lines
tail -n 20 file.txt

# Follow file updates
tail -f logfile.txt

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

wc

```
# Count lines
wc -l file.txt

# Count words
wc -w file.txt

# Count characters
wc -c file.txt

# All statistics
wc file.txt
```

Combining Tools

Real-world pipeline examples.

Log Analysis

```
# Find unique IPs from access log
awk '{print $1}' access.log | sort | uniq -c | sort -rn | head

# Count errors by hour
grep "ERROR" app.log | awk '{print substr($2,1,2)}' | sort | uniq -c

# Find slow requests
awk '$5 > 1000 {print $0}' access.log | sort -k5 -rn | head -10
```

File Management

```
# Find and delete old logs
find /var/log -name "*.log" -mtime +30 -exec rm {} \;

# Find large files
find . -type f -size +100M -exec ls -lh {} \;

# Count files by extension
find . -type f | sed 's/.*\.//' | sort | uniq -c | sort -rn
```

Text Processing

```
# Extract emails from file
grep -oE '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}' file.txt

# CSV to TSV
tr ',' '\t' < file.csv > file.tsv

# Remove duplicate lines keeping order
awk '!seen[$0]++' file.txt
```

Quick Reference

Tool	Purpose	Key Flags
find	Search files	`-name`, `-type`, `-size`, `-mtime`, `-exec`
grep	Search text	`-r`, `-i`, `-n`, `-v`, `-E`, `-c`
sed	Edit streams	`s///g`, `-i`, `/d`, `-n`
awk	Process data	`-F`, `$N`, `NR`, `NF`, `BEGIN/END`
xargs	Build commands	`-I`, `-n`, `-P`, `-0`
sort	Sort lines	`-n`, `-r`, `-k`, `-u`
uniq	Deduplicate	`-c`, `-d`, `-u`
cut	Extract columns	`-d`, `-f`, `-c`
tr	Transform chars	`-d`, `-s`
head	First lines	`-n`
tail	Last lines	`-n`, `-f`
wc	Count	`-l`, `-w`, `-c`

Tips

Performance Tips

```
# Use grep before awk for speed
grep "pattern" file | awk '{print $1}'

# Use LC_ALL=C for faster sorting
LC_ALL=C sort largefile.txt
```

```
# Parallel xargs for multi-core
find . -type f | xargs -P 8 -n 1 command
```

Common Patterns

```
# Find and replace in multiple files
grep -rl "oldtext" . | xargs sed -i 's/oldtext/newtext/g'

# Monitor log in real-time
tail -f /var/log/app.log | grep --line-buffered "ERROR"

# Extract column and get unique values
awk -F, '{print $2}' data.csv | sort -u
```

Debugging

```
# Test sed before applying
sed 's/old/new/g' file.txt | head

# Dry-run find delete
find . -name "*.tmp" -print  # Add -delete after verifying

# Verbose xargs
echo "file1 file2" | xargs -t rm
```

Next Steps

- Git CLI Cheat Sheet — Version control commands
- Docker CLI Cheat Sheet — Container management reference
- Kubernetes kubectl Cheat Sheet — K8s command reference