Complete Unix/Linux Command Reference Guide

AWK • SED • GREP • CRON

Table of Contents

AWK - Pattern Scanning and Processing

Overview

AWK is a powerful pattern-scanning and processing language for text files. It processes files line by line and can perform complex text manipulation, calculations, and reporting tasks.

Basic Syntax

```bash
awk 'pattern { action }' filename
awk -f script.awk filename
```

Built-in Variables

- `NR` - Number of records (line number)
- `NF` - Number of fields in current record
- `$0` - Entire current record
- `$1, $2, $3...` - Field 1, 2, 3, etc.
- `FS` - Field separator (default: whitespace)
- `RS` - Record separator (default: newline)

- `OFS` - Output field separator (default: space)
- `ORS` - Output record separator (default: newline)

Essential AWK Commands

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

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

# Count occurrences
awk '{ count[$1]++ } END { for(i in count) print i, count[i] }' file.txt

# Conditional processing
awk '$1 > 100 && $1 ~ /^[0-9]+$/' file.txt

# String functions
awk '{ print toupper($0) }' file.txt
awk '{ gsub(/old/, "new"); print }' file.txt
```

SED - Stream Editor

Overview

SED (Stream Editor) is a powerful stream editor for filtering and transforming text in a pipeline. It performs basic text transformations on an input stream.

Basic Syntax

```bash
sed 'command' filename
sed -e 'command1' -e 'command2' filename
sed -f script.sed filename
```

Common Options

- `-n` - Suppress automatic printing

- `-e` - Add script command
- `-f` - Add script file
- `-i` - Edit files in-place
- `-r` or `-E` - Extended regular expressions

Essential SED Commands

```bash
# Basic substitution
sed 's/old/new/g' file.txt

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

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

# Insert/append text
sed '3i\New line of text' file.txt
sed '3a\New line of text' file.txt

# Multiple commands
sed -e 's/old1/new1/g' -e 's/old2/new2/g' file.txt
```

Useful SED One-liners

```bash
# Remove leading whitespace
sed 's/^[ \t]*//' file.txt

# Remove blank lines
sed '/^$/d' file.txt

# Add line numbers
sed = file.txt | sed 'N;s/\n/\t/'

# Double space file
sed 'G' file.txt
```

GREP - Global Regular Expression Print

Overview

GREP is a command-line utility for searching text patterns within files. It's essential for text processing and log analysis.

Basic Syntax

```bash
grep [options] pattern [file...]
```

Essential Options

- `-i` - Ignore case
- `-v` - Invert match
- `-n` - Show line numbers
- `-r` - Recursive search
- `-c` - Count matches
- `-l` - List filenames with matches
- `-A n` - Show n lines after match
- `-B n` - Show n lines before match
- `-C n` - Show n lines around match

Essential GREP Commands

```bash
```

```
# Basic search
grep "pattern" file.txt

# Case-insensitive with line numbers
grep -in "pattern" file.txt

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

# Multiple patterns
grep -E "pattern1|pattern2" file.txt

# Find IP addresses
grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" file.txt

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

Regular Expressions

```bash
# Beginning/end of line
grep "^start" file.txt
grep "end$" file.txt

# Character classes
grep "[0-9]+" file.txt
grep "[a-zA-Z]" file.txt

# Word boundaries
grep "\bword\b" file.txt
```

CRON - Job Scheduler

Overview

Cron is a time-based job scheduler in Unix-like systems for running commands automatically at specified times.

Crontab Format

```
# ┌───────────── minute (0 - 59)
# │ ┌───────────── hour (0 - 23)
# │ │ ┌───────────── day of month (1 - 31)
# │ │ │ ┌───────────── month (1 - 12)
# │ │ │ │ ┌───────────── day of week (0 - 6)
# │ │ │ │ │
# * * * * * command to execute
```

Crontab Management

```bash
# View crontab
crontab -l

# Edit crontab
crontab -e

# Remove crontab
crontab -r

# Install from file
crontab filename
```

Common Cron Expressions

```bash
# Every minute
* * * * * /path/to/command

# Daily at 2:30 AM
30 2 * * * /path/to/command

# Every weekday at 9 AM
0 9 * * 1-5 /path/to/command

# Every 15 minutes
*/15 * * * * /path/to/command

# First day of every month
0 0 1 * * /path/to/command
```

Special Strings

```bash
@reboot     # Run at startup
@yearly     # Run once a year
@monthly    # Run once a month
@weekly     # Run once a week
@daily      # Run once a day
@hourly     # Run once an hour
```

Integration Examples

Combining AWK, SED, and GREP

```bash
# Log analysis pipeline
grep "ERROR" /var/log/app.log | \
sed 's/.*\[\([0-9-]*\)\].*/\1/' | \
awk '{count[$0]++} END {for(date in count) print date, count[date]}' | \
sort

# Process CSV data
grep -v "^#" data.csv | \
sed 's/,/ /g' | \
awk '$3 > 1000 {sum += $3; count++} END {print "Average:", sum/count}'

# Clean and analyze config files
sed '/^#/d; /^$/d' config.txt | \
grep "=" | \
awk -F= '{gsub(/[ \t]/, "", $1); gsub(/[ \t]/, "", $2); print $1 ": " $2}'
```

Automated System Maintenance with CRON

```bash
```

```
# Daily log cleanup and analysis
0 2 * * * grep "ERROR\|WARN" /var/log/app.log | \
    awk '{print strftime("%Y-%m-%d"), $0}' > /tmp/daily_errors.log && \
    sed -i '/ERROR\|WARN/d' /var/log/app.log

# Weekly report generation
0 1 * * 0 grep "user_login" /var/log/auth.log | \
    sed 's/.*user=\([^ ]*\).*/\1/' | \
    awk '{count[$0]++} END {for(user in count) print user, count[user]}' | \
    sort -k2 -nr > /reports/weekly_logins.txt
```

Advanced Text Processing Pipeline

```bash
# Extract, clean, and summarize data
grep -E "^[0-9]{4}-[0-9]{2}-[0-9]{2}" logfile.txt | \
sed 's/\[DEBUG\]//g; s/\[INFO\]//g' | \
awk '
BEGIN { FS="|" }
/ERROR/ { errors++ }
/SUCCESS/ { success++ }
{
  gsub(/^ +| +$/, "", $2)  # trim whitespace
  if (length($2) > 0) operations[$2]++
}
END {
  print "=== SUMMARY ==="
  print "Errors:", errors
  print "Success:", success
  print "=== OPERATIONS ==="
  for (op in operations) {
    printf "%-20s: %d\n", op, operations[op]
  }
}'
```

Quick Reference Cards

AWK Quick Reference

Operation	Syntax	Example
Print fields	`{ print $1, $2 }`	`awk '{ print $1, $2 }' file.txt`
Sum column	`{ sum += $1 } END { print sum }`	`awk '{ sum += $3 } END { print sum }' sales.txt`
Count pattern	`/pattern/ { count++ } END { print count }`	`awk '/error/ { count++ } END { print count }' log.txt`
Field separator	`-F'separator'`	`awk -F',' '{ print $1 }' data.csv`
Conditional	`$1 > 100 { print }`	`awk '$2 > 50 { print $1 }' scores.txt`

SED Quick Reference

Operation	Syntax	Example
Substitute	`s/old/new/g`	`sed 's/foo/bar/g' file.txt`
Delete lines	`/pattern/d`	`sed '/^#/d' config.txt`
Print lines	`-n 'Np'`	`sed -n '5,10p' file.txt`
Insert text	`Ni\text`	`sed '3i\New line' file.txt`
In-place edit	`-i`	`sed -i 's/old/new/g' file.txt`

GREP Quick Reference

Operation	Syntax	Example
Basic search	`grep pattern file`	`grep "error" log.txt`
Case insensitive	`grep -i pattern file`	`grep -i "ERROR" log.txt`
Line numbers	`grep -n pattern file`	`grep -n "function" code.py`
Recursive	`grep -r pattern dir/`	`grep -r "TODO" src/`
Count matches	`grep -c pattern file`	`grep -c "warning" log.txt`
Context lines	`grep -A3 -B3 pattern file`	`grep -A2 -B2 "error" log.txt`

CRON Quick Reference

Schedule	Cron Expression	Description
Every minute	`* * * * *`	Runs every minute
Every hour	`0 * * * *`	Runs at the start of every hour
Daily at 2 AM	`0 2 * * *`	Runs at 2:00 AM every day

Schedule	Cron Expression	Description
Weekly	`0 0 * * 0`	Runs at midnight every Sunday
Monthly	`0 0 1 * *`	Runs at midnight on the 1st of every month
Weekdays 9 AM	`0 9 * * 1-5`	Runs at 9:00 AM Monday through Friday

Common Patterns and Use Cases

Log Analysis

bash

```bash
# Find top IP addresses in access log
grep -o -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" access.log | \
sort | uniq -c | sort -nr | head -10

# Extract error messages with timestamps
sed -n '/ERROR/p' app.log | \
awk '{print $1, $2, $NF}' | \
sort | uniq -c
```

Data Processing

bash

```bash
# CSV processing with validation
grep -v "^#" data.csv | \
sed 's/,/ /g' | \
awk 'NF==5 && $3~/^[0-9]+$/ {sum+=$3; count++} END {print "Avg:", sum/count}'

# Configuration file processing
sed '/^#/d; /^$/d' /etc/config | \
grep "=" | \
awk -F= '{gsub(/^[ \t]+|[ \t]+$/, "", $2); print $1 "=" $2}'
```

System Monitoring (Cron Jobs)

bash

```
# Disk space monitoring
0 * * * * df -h | awk '$5 > 80 {print}' | mail -s "Disk Space Alert" admin@domain.com

# Process monitoring
*/5 * * * * ps aux | awk '$3 > 80 {print}' > /tmp/high_cpu_processes.log

# Log rotation and cleanup
0 0 * * * find /var/log -name "*.log" -size +100M -exec gzip {} \;
```

This comprehensive guide provides all the essential information for mastering these four crucial Unix/Linux command-line tools. Each tool has its strengths, and combining them creates powerful text processing and automation workflows.