GREP Command Reference Guide

Overview

GREP (Global Regular Expression Print) is a command-line utility for searching text patterns within files. It's one of the most essential tools for text processing and log analysis.

Basic Syntax

```bash
grep [options] pattern [file...]
grep [options] [-e pattern | -f file] [file...]
```

Essential Options

- `-i` - Ignore case distinctions
- `-v` - Invert match (select non-matching lines)
- `-n` - Show line numbers
- `-c` - Count matching lines
- `-l` - List filenames with matches
- `-L` - List filenames without matches
- `-w` - Match whole words only
- `-x` - Match whole lines only
- `-r` or `-R` - Recursive search in directories
- `-H` - Print filename with matches (default with multiple files)
- `-h` - Suppress filename in output
- `-o` - Show only matching part of line
- `-A n` - Show n lines after match
- `-B n` - Show n lines before match
- `-C n` - Show n lines before and after match

Basic Usage

Simple Pattern Matching

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

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

# Search in multiple files
grep "pattern" file1.txt file2.txt

# Search in all files in directory
grep "pattern" *

# Search recursively in directory
grep -r "pattern" /path/to/directory
```

Line Numbers and Context

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

# Show 3 lines before and after match
grep -C 3 "pattern" file.txt

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

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

Regular Expressions

Basic Regular Expressions

```bash
```

```bash
# Match beginning of line
grep "^pattern" file.txt

# Match end of line
grep "pattern$" file.txt

# Match any single character
grep "p.ttern" file.txt

# Match zero or more occurrences
grep "colou*r" file.txt  # Matches "color" and "colour"

# Match one or more occurrences
grep "colou\+r" file.txt

# Match specific number of characters
grep "^.\{5\}$" file.txt  # Lines with exactly 5 characters
```

Extended Regular Expressions (-E flag)

```bash
# One or more occurrences
grep -E "colou+r" file.txt

# Zero or one occurrence
grep -E "colou?r" file.txt

# Alternation (OR)
grep -E "cat|dog" file.txt

# Grouping
grep -E "(cat|dog)s?" file.txt

# Character ranges
grep -E "[0-9]+" file.txt  # Numbers
grep -E "[a-zA-Z]+" file.txt  # Letters

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

Character Classes

```bash
# Digits
grep "[0-9]" file.txt
grep "[[:digit:]]" file.txt

# Letters
grep "[a-zA-Z]" file.txt
grep "[[:alpha:]]" file.txt

# Alphanumeric
grep "[[:alnum:]]" file.txt

# Whitespace
grep "[[:space:]]" file.txt

# Punctuation
grep "[[:punct:]]" file.txt
```

Advanced Usage

Multiple Patterns

```bash
# Multiple patterns (OR)
grep -e "pattern1" -e "pattern2" file.txt
grep -E "pattern1|pattern2" file.txt

# Patterns from file
grep -f patterns.txt file.txt

# Match all patterns (AND) using pipeline
grep "pattern1" file.txt | grep "pattern2"
```

Inverting Matches

```bash
```

```bash
# Lines NOT containing pattern
grep -v "pattern" file.txt

# Files NOT containing pattern
grep -L "pattern" *.txt

# Lines not matching regex
grep -v "^#" config.txt  # Remove comments
```

Counting and Listing

bash

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

# Count total matches (including multiple per line)
grep -o "pattern" file.txt | wc -l

# List files containing pattern
grep -l "pattern" *.txt

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

Practical Examples

Log File Analysis

bash

```bash
# Find error messages
grep -i "error" /var/log/syslog

# Find errors in the last hour
grep "$(date '+%b %d %H')" /var/log/syslog | grep -i error

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

# Find failed login attempts
grep "Failed password" /var/log/auth.log

# Count unique IP addresses
grep -o -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" access.log | sort | uniq -c
```

Code and Configuration Files

```bash
# Find function definitions
grep -n "^function\|^def " *.py

# Find TODO comments
grep -r -n "TODO\|FIXME" src/

# Find empty lines
grep "^$" file.txt

# Find non-empty lines
grep -v "^$" file.txt

# Find configuration settings
grep "^[^#]*=" config.txt  # Non-comment lines with =
```

Text Processing

```bash
```

```
# Find email addresses
grep -E "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}" file.txt

# Find phone numbers (simple pattern)
grep -E "[0-9]{3}-[0-9]{3}-[0-9]{4}" file.txt

# Find URLs
grep -E "https?://[a-zA-Z0-9./?=_%:-]*" file.txt

# Find words with specific length
grep -E "\b\w{5}\b" file.txt  # 5-letter words
```

System Administration

bash

```
# Find running processes
ps aux | grep "process_name"

# Find files modified today
find . -type f -name "*.log" | xargs grep -l "$(date '+%Y-%m-%d')"

# Search in compressed files
zgrep "pattern" *.gz

# Find SUID files
find / -perm -4000 2>/dev/null | grep -v "Permission denied"
```

Combining with Other Tools

With find

bash

```
# Search pattern in specific file types
find . -name "*.py" -exec grep -l "pattern" {} \;

# Search in files modified in last 7 days
find . -mtime -7 -type f -exec grep -l "pattern" {} \;
```

With xargs

bash

```
# Search in multiple files efficiently
find . -name "*.txt" | xargs grep "pattern"

# Safe version with null separator
find . -name "*.txt" -print0 | xargs -0 grep "pattern"
```

With awk and sed

```
bash

# Extract and process matches
grep -o "Error: [0-9]*" log.txt | awk -F: '{print $2}' | sort -n

# Replace found patterns
grep -l "old_pattern" *.txt | xargs sed -i 's/old_pattern/new_pattern/g'
```

Performance Optimization

Fast Searching

```
bash

# Use fixed strings for better performance
grep -F "literal_string" large_file.txt

# Stop after first match
grep -m 1 "pattern" file.txt

# Search binary files
grep -a "pattern" binary_file

# Exclude binary files
grep -I "pattern" *

# Use parallel processing
find . -name "*.txt" | parallel grep "pattern" {}
```

Memory Considerations

```
bash
```

```bash
# Process large files line by line
grep --line-buffered "pattern" huge_file.txt

# Search without loading entire file
grep -F "pattern" large_file.txt

# Use memory-mapped files (GNU grep)
grep --mmap "pattern" file.txt
```

Useful One-Liners

```bash
# Find duplicate lines
grep -n ".*" file.txt | sort -k2 | uniq -f1 -D

# Find lines with only numbers
grep "^[0-9]*$" file.txt

# Find lines with mixed case
grep "[a-z].*[A-Z]\|[A-Z].*[a-z]" file.txt

# Find blank lines and line numbers
grep -n "^$" file.txt

# Count words per line
grep -o "\w\+" file.txt | wc -l

# Find longest line
grep -E ".{$(wc -L < file.txt)}" file.txt

# Remove trailing spaces (with sed)
grep -l "[ \t]$" *.txt | xargs sed -i 's/[ \t]*$//'

# Find files with Windows line endings
grep -l $'\r' *

# Find non-ASCII characters
grep -P "[^\x00-\x7F]" file.txt
```

GREP Variants

Basic vs Extended vs Perl Regular Expressions

```bash
# Basic Regular Expressions (default)
grep "pattern\+" file.txt

# Extended Regular Expressions
grep -E "pattern+" file.txt
egrep "pattern+" file.txt

# Perl-Compatible Regular Expressions
grep -P "pattern+" file.txt
```

Related Commands

```bash
# fgrep - fixed strings (same as grep -F)
fgrep "literal_string" file.txt

# egrep - extended regex (same as grep -E)
egrep "pattern+" file.txt

# rgrep - recursive (same as grep -r)
rgrep "pattern" directory/

# zgrep - search compressed files
zgrep "pattern" file.gz
```

Environment Variables

```bash
```

```
# Set default options
export GREP_OPTIONS="--color=auto -n"

# Customize colors
export GREP_COLORS="mt=1;31:fn=1;32:ln=33"

# Set default pattern file
export GREP_FILE="~/.greprc"
```

Tips and Best Practices

1. **Use quotes** around patterns to prevent shell expansion
2. **Test patterns** with simple examples before using on large files
3. **Use -F for literal strings** when you don't need regex features
4. **Combine options** effectively (e.g., -rni for recursive, numbered, case-insensitive)
5. **Use context options** (-A, -B, -C) for better understanding
6. **Consider performance** with large files and complex patterns
7. **Use appropriate regex type** (basic, extended, or Perl)
8. **Escape special characters** when needed

Common Pitfalls

- Forgetting to escape special regex characters
- Using basic regex syntax with -E flag or vice versa
- Not quoting patterns with spaces or special shell characters
- Expecting grep to work with binary files (use -a if needed)
- Not considering case sensitivity
- Using inefficient patterns for simple literal string searches
- Forgetting that grep returns non-zero exit code when no matches found

See Also

- awk - Pattern scanning and processing language
- sed - Stream editor for filtering and transforming text
- find - Search for files and directories
- cut - Extract specific columns

- `sort` - Sort lines of text
- `uniq` - Remove duplicate lines