SED Command Reference Guide

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 (file or input from a pipeline).

Basic Syntax

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

Common Options

- `-n` - Suppress automatic printing of pattern space
- `-e` - Add script command to execute
- `-f` - Add script file to execute
- `-i` - Edit files in-place (backup with `-i.bak`)
- `-r` or `-E` - Use extended regular expressions
- `-s` - Treat files as separate (don't concatenate)

Basic Commands

Substitution (s command)

```bash
```

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

# Global substitution (all occurrences on each line)
sed 's/old/new/g' file.txt

# Substitute only on specific line
sed '5s/old/new/' file.txt

# Substitute with different delimiter
sed 's|/old/path|/new/path|g' file.txt

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

Print (p command)

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

# Print range of lines
sed -n '1,5p' file.txt

# Print lines matching pattern
sed -n '/pattern/p' file.txt

# Print lines between patterns
sed -n '/start/,/end/p' file.txt
```

Delete (d command)

```bash
```

```bash
# Delete specific line
sed '5d' file.txt

# Delete range of lines
sed '1,5d' file.txt

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

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

# Delete last line
sed '$d' file.txt
```

Address Patterns

Line Numbers

```bash
# Single line
sed '3s/old/new/' file.txt

# Multiple lines
sed '1,5s/old/new/' file.txt

# From line to end
sed '10,$s/old/new/' file.txt

# Every nth line
sed '1~2s/old/new/' file.txt  # Every 2nd line starting from 1
```

Pattern Matching

```bash
```

```bash
# Lines containing pattern
sed '/pattern/s/old/new/' file.txt

# Lines between two patterns
sed '/start/,/end/s/old/new/' file.txt

# Lines NOT matching pattern
sed '/pattern/!s/old/new/' file.txt

# Multiple patterns
sed -e '/pattern1/s/old/new/' -e '/pattern2/s/foo/bar/' file.txt
```

Advanced Substitution

Back-references

bash

```bash
# Capture groups with \( \) and reference with \1, \2, etc.
sed 's/\([0-9]*\)-\([0-9]*\)/\2-\1/' file.txt

# Swap first and last word
sed 's/^\([^ ]*\) \(.*\) \([^ ]*\)$/\3 \2 \1/' file.txt

# Add parentheses around numbers
sed 's/[0-9][0-9]*/(&)/' file.txt
```

Special Characters in Replacement

bash

```bash
# Use & to represent matched text
sed 's/[0-9]*/[&]/' file.txt

# Escape special characters
sed 's/\./DOT/g' file.txt

# Use different delimiters for paths
sed 's#/old/path#/new/path#g' file.txt
```

Text Insertion and Appending

Insert (i command)

```bash
# Insert text before line 3
sed '3i\New line of text' file.txt

# Insert before lines matching pattern
sed '/pattern/i\Inserted text' file.txt
```

Append (a command)

```bash
# Append text after line 3
sed '3a\New line of text' file.txt

# Append after lines matching pattern
sed '/pattern/a\Appended text' file.txt
```

Change (c command)

```bash
# Replace entire line 3
sed '3c\Replacement line' file.txt

# Replace lines matching pattern
sed '/pattern/c\New content' file.txt
```

Multiple Commands

Using -e flag

```bash
sed -e 's/old1/new1/g' -e 's/old2/new2/g' file.txt
```

Using semicolon separator

```bash
```

```
sed 's/old1/new1/g; s/old2/new2/g' file.txt
```

Using script file

```bash
# Create script.sed:
s/old1/new1/g
s/old2/new2/g
/pattern/d

# Execute script
sed -f script.sed file.txt
```

Practical Examples

Text Cleaning

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

# Remove trailing whitespace
sed 's/[ \t]*$//' file.txt

# Remove both leading and trailing whitespace
sed 's/^[ \t]*//; s/[ \t]*$//' file.txt

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

# Remove comments (lines starting with #)
sed '/^#/d' file.txt

# Remove HTML tags
sed 's/<[^>]*>//g' file.html
```

File Format Conversion

```bash
```

```
# Convert DOS to Unix line endings
sed 's/\r$//' dosfile.txt

# Convert Unix to DOS line endings
sed 's/$/\r/' unixfile.txt

# Convert tabs to spaces
sed 's/\t/    /g' file.txt

# Convert spaces to tabs
sed 's/    /\t/g' file.txt
```

Data Extraction and Manipulation

bash

```
# Extract email addresses (simple pattern)
sed -n 's/.*\([a-zA-Z0-9._%+-]*@[a-zA-Z0-9.-]*\.[a-zA-Z]*\).*/\1/p' file.txt

# Extract IP addresses from log
sed -n 's/.*\([0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\).*/\1/p' log.txt

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

# Double space file
sed 'G' file.txt

# Remove double spacing
sed 'n;d' doublespaced.txt
```

Configuration File Editing

bash

```
# Uncomment lines (remove leading #)
sed 's/^#//' config.txt

# Comment out lines containing pattern
sed '/pattern/s/^/#/' config.txt

# Change configuration value
sed 's/^setting=.*/setting=newvalue/' config.txt

# Add configuration if not exists
sed '/^setting=/!s/$/\nsetting=value/' config.txt
```

Advanced Techniques

Hold Space Operations

bash

```
# Print line and next line
sed -n 'N;p' file.txt

# Reverse order of two lines
sed 'N;s/\(.*\)\n\(.*\)/\2\n\1/' file.txt

# Join lines ending with backslash
sed ':a;/\\$/N;s/\\\n//;ta' file.txt
```

Branch and Test Commands

bash

```
# Skip processing if pattern found
sed '/pattern/b' file.txt

# Process only if pattern NOT found
sed '/pattern/!s/old/new/' file.txt
```

Multi-line Patterns

bash

```bash
# Delete lines containing pattern and next line
sed '/pattern/{N;d;}' file.txt

# Replace pattern across multiple lines
sed ':a;N;$!ba;s/pattern\nmultiline/replacement/g' file.txt
```

Useful One-Liners

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

# Print last line
sed -n '$p' file.txt

# Print every 2nd line
sed -n '1~2p' file.txt

# Remove first and last line
sed '1d;$d' file.txt

# Replace word only if at beginning of line
sed 's/^word/replacement/' file.txt

# Add text at beginning of each line
sed 's/^/PREFIX: /' file.txt

# Add text at end of each line
sed 's/$/ SUFFIX/' file.txt

# Number non-empty lines
sed '/./=' file.txt | sed '/./N;s/\n/ /'

# Reverse lines (tac alternative)
sed '1!G;h;$!d' file.txt

# Center text (40 character width)
sed 's/.*/                &/' file.txt | sed 's/\(.\{40\}\).*/\1/'
```

In-place Editing

Basic in-place editing

```bash
# Edit file directly (dangerous!)
sed -i 's/old/new/g' file.txt

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

# Edit multiple files
sed -i 's/old/new/g' *.txt
```

Safe in-place editing

```bash
# Always test first without -i
sed 's/old/new/g' file.txt | head

# Then apply changes
sed -i.$(date +%Y%m%d) 's/old/new/g' file.txt
```

Tips and Best Practices

1. **Always test without -i** before in-place editing
2. **Use different delimiters** when working with paths (s#old#new# instead of s/old/new/)
3. **Escape special characters** properly (. * [ ] etc.)
4. **Use single quotes** to avoid shell interpretation
5. **Combine with other tools** for complex processing
6. **Create backups** when editing files in-place
7. **Use -n with p** to suppress default printing when needed
8. **Test with small files** first

Common Pitfalls

- Forgetting to escape special regex characters
- Using double quotes and having shell variables expanded unexpectedly

- Not creating backups before in-place editing
- Confusing basic and extended regular expressions
- Not understanding greedy vs non-greedy matching
- Forgetting that sed processes line by line

See Also

- awk - Pattern scanning and processing language
- grep - Pattern matching and searching
- tr - Translate or delete characters
- cut - Extract specific columns
- perl - More powerful text processing