# Text Processing Utilities: sed, awk, and grep

Linux offers powerful text processing utilities that are commonly used for manipulating and searching text. In this article, we'll dive into the syntax, options, and examples of three essential commands: sed, awk, and grep.

BY sravani



# Sed - Stream Editor

Sed is a text editor that operates on a per-line basis, making it ideal for performing text transformations. With sed, you can easily modify input text or files.

**Syntax:** sed [options] 'command' input\_file

Options: -i or --in-place, -e or --expression, -n or --quiet or --silent

**Example:** Replace all occurrences of "old" with "new" in a file called

text.txt

#### AWK command in Linux

```
awk [options] [--] 'program' file ...
awk [options] -f program-file [--] file ...
awk 'program' file ...
program:
  pattern { action }
  pattern { action }
pattern:

    regular expression

    conditional expression like length ($2) > 12
```

# Advanced Sed Techniques

In-Place Editing

(\_i)
Edit files directly, without creating backups

Multiple
Commands (-e)
Apply multiple sed

commands

Suppress Printing
(-n)
Prevent automatic printing
of lines

## Sed Commands

- Substitution
  (s/old/new/)
  Replace occurrences of "old" with "new" sed 's/old/new/g' text.txt
- 3 Print (p)

  Print the selected line(s)

  sed 's/apple/orange/' fruits.txt

- Delete (d)

  Delete the selected line(s)

  sed '/remove/d' data.txt
- Insert (i and

  Sert text before or after the selected

  line(s) 's/old/new/' -e 's/apples/oranges/'
  input.txt

# ja@sofija-VirtualBox:-\$ sed 's/box/bin/' foxinbox.txt Knox on fox in socks in bin. Socks on Knox and Knox in bin. Fox in socks on bin on Knox.

# Awk - Text Processing

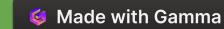
Tool

Awk is a versatile text processing tool that allows you to perform various text manipulation tasks. It processes text line by line, making it perfect for extracting information from structured data.

**Syntax:** awk [options] 'pattern {action}' input\_file

**Options:** -F or --field-separator

**Example:** Print the second column of a CSV file (data.csv)



# Awk Patterns and Actions

awk is a versatile text processing tool in Linux and Unix-like operating systems. It's named after its authors: Alfred Aho, Peter Weinberger, and Brian Kernighan. awk operates on text files and is commonly used for text manipulation, data extraction, and report generation.

- Pattern

  Specifies a condition that triggers the associated action
- 2 Action

  Block of awk code that executes when the pattern is true

### Awk Examples

#### Printing Specific

Extract and print specific columns from a file

awk '{print \$2}' data.txt

#### Calculating

Totals Perform calculations and display the result

awk '{sum+=\$3} END {print sum}' data.txt

#### Filtering Data

Display lines based on certain conditions

awk '\$4 > 50' data.txt

#### Modifying

Fields Replace or manipulate values in specific

fields awk -F ',' '{print \$2, \$4}' data.csv

# Grep - Search Text with Regular Expressions

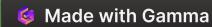
Grep is a powerful tool used to search text using regular expressions within files or input streams. It allows you to easily find specific patterns or words within your text.

**Syntax:** grep [options] 'pattern' [file(s)]

**Options:** -i or --ignore-case, -r or --recursive, -l or --files-with-matches

**Example:** Search for lines containing the word "error" in a file called

logfile.log

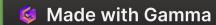


#### Grep

grep is a powerful and widely-used command-line utility in Linux and Unix-like operating systems for searching text patterns within files or input streams. The name "grep" stands for "Global Regular Expression Print," which reflects its primary function: to search for and print lines that match a specified pattern (regular expression) in a text source.

#### Options:

- -i or --ignore-case: Perform a case-insensitive search.
- -r or --recursive: Recursively search in directories.
- -l or --files-with-matches: Display filenames with matching lines.
- -v or --invert-match: Invert the match, i.e., print lines that do not match the pattern.
- -n or --line-number: Display line numbers along with matching lines.
- -w or --word-regexp: Match only whole words, not substrings.
- -c or --count: Display the count of matching lines.
- -A NUM or --after-context=NUM: Display NUM lines of context after each matching line.
- -B NUM or --before-context=NUM: Display NUM lines of context before each matching line.
- -C NUM or --context=NUM: Display NUM lines of context before and after each matching line.



## Grep Examples

Case-Insensitive Search

Perform a search that

ignores case

grep -i 'ERROR' logfile.log

Recursive

Search or a pattern in files

within directories

grep -l 'search'

/path/to/directory/\*

Displaying File Names

Show filenames that

contain matching lines

grep -n 'pattern' file.txt

# Real-World Examples

Discover how sed, awk, and grep are used in real-world scenarios, ranging from log analysis to data manipulation. Unlock the full potential of text processing in your projects.

```
cat test.txt
#cat five black
#dog four white
this is start
#rats ten black
ants hundred black
insect hundred black
 sed -e '1,/start/ s/#.*//' test.txt
this is start
```

this is start #rats ten black ants hundred black insect hundred black

#### Conclusion

Sed, awk, and grep are indispensable text processing utilities that can save you valuable time and effort. By harnessing their power, you can efficiently manipulate and search text, opening up endless possibilities.

