Text Processing Tools Cheatsheet: grep, sed, awk

A quick reference for using grep, sed, and awk for text processing in Linux/Unix environments, covering essential commands, options, and syntax with practical examples.

grep - Global Regular Expression Print

Basic Usage

`grep 'pattern' file`	Search for 'pattern' in 'file' and print matching lines. **Example:** `grep 'error' logfile.txt`
`grep -i 'pattern' file`	Case-insensitive search. **Example:** `grep -i 'Error' logfile.txt`
`grep -v 'pattern' file`	Invert the match: print lines that do *not* contain the pattern. **Example:** `grep -v 'success' logfile.txt`
`grep -n 'pattern' file`	Print the line number with each matching line. **Example:** `grep -n 'warning' logfile.txt`
`grep -c 'pattern' file`	Print only a count of matching lines. **Example:** `grep -c 'pattern' file.txt`
`grep -l 'pattern' file1 file2 ...`	List only the names of files that contain the pattern. **Example:** `grep -l 'pattern' file1.txt file2.txt`
`grep -h 'pattern' file1 file2 ...`	Suppress the prefixing of filenames on output when multiple files are searched. **Example:** `grep -h 'pattern' file1.txt file2.txt`

Advanced grep

`grep -E 'pattern' file`	Use extended regular expressions. **Example:** `grep -E 'pattern1|pattern2' file.txt`
`grep -w 'pattern' file`	Search for whole words only. **Example:** `grep -w 'word' file.txt`
`grep -r 'pattern' directory`	Recursively search through files in a directory. **Example:** `grep -r 'pattern' ./path/to/directory`
`grep -o 'pattern' file`	Print only the matching part of the lines. **Example:** `grep -o '[0-9]\+' file.txt`
`grep -A n 'pattern' file`	Print 'n' lines after the matching line. **Example:** `grep -A 2 'pattern' file.txt`
`grep -B n 'pattern' file`	Print 'n' lines before the matching line. **Example:** `grep -B 2 'pattern' file.txt`
`grep -C n 'pattern' file`	Print 'n' lines around the matching line (context). **Example:** `grep -C 2 'pattern' file.txt`

Regular Expressions in grep

`.` - Matches any single character.

Example:

```
grep 'a.c' file.txt
```

`[]` - Matches any character inside the brackets.

Example:

```
grep '[abc]' file.txt
```

`[^]` - Matches any character not inside the brackets.

Example:

```
grep '[^abc]' file.txt
```

`*` - Matches zero or more occurrences of the preceding character.

Example:

```
grep 'ab*c' file.txt
```

`\+` - Matches one or more occurrences of the preceding character (with `-E`).

Example:

```
grep -E 'ab+c' file.txt
```

`?` - Matches zero or one occurrence of the preceding character (with `-E`).

Example:

```
grep -E 'ab?c' file.txt
```

`^` - Matches the beginning of the line.

Example:

```
grep '^abc' file.txt
```

`$` - Matches the end of the line.

Example:

```
grep 'abc$' file.txt
```

`\|` - OR operator (with `-E`).

Example:

```
grep -E 'abc|def' file.txt
```

sed - Stream EDitor

Basic sed Commands

`sed 's/old/new/g' file`	Replace all occurrences of 'old' with 'new' in 'file'. **Example:** `sed 's/apple/orange/g' fruit.txt`
`sed 's/old/new/' file`	Replace the first occurrence of 'old' with 'new' in each line. **Example:** `sed 's/apple/orange/' fruit.txt`
`sed 's/old/new/i' file`	Case-insensitive replacement (GNU sed). **Example:** `sed 's/apple/orange/i' fruit.txt`
`sed '/pattern/d' file`	Delete lines containing 'pattern'. **Example:** `sed '/error/d' logfile.txt`
`sed '2d' file`	Delete the second line. **Example:** `sed '2d' file.txt`
`sed '$d' file`	Delete the last line. **Example:** `sed '$d' file.txt`
`sed '1,3d' file`	Delete lines 1 to 3. **Example:** `sed '1,3d' file.txt`
`sed '1,/pattern/d' file`	Delete from line 1 to the first line containing 'pattern'. **Example:** `sed '1,/error/d' logfile.txt`

sed Substitution Flags

`g` - Global replacement (all occurrences in a line).

Example:

```
sed 's/old/new/g' file.txt
```

`i` - Case-insensitive replacement (GNU sed).

Example:

```
sed 's/old/new/i' file.txt
```

`w file` - Write the result to a file.

Example:

```
sed 's/old/new/w output.txt' file.txt
```

`&` - Represents the matched string.

Example:

```
sed 's/[0-9]+/Number: &/g' file.txt
```

`\1`, `\2`, ... - Backreferences to captured groups.

Example:

```
sed 's/\(group1\) \(group2\)/\2 \1/' file.txt
```

sed - Advanced

`sed -n 'p' file`	Suppress automatic printing and print only specific lines. **Example:** `sed -n '4p' file.txt # Prints only the 4th line`
`sed -n '/pattern/p' file`	Print only lines matching a pattern. **Example:** `sed -n '/error/p' logfile.txt`
`sed 'y/abc/xyz/' file`	Translate 'a' to 'x', 'b' to 'y', 'c' to 'z'. **Example:** `sed 'y/abc/xyz/' file.txt`
`sed -i 's/old/new/g' file`	Modify the file in-place (use with caution!). **Example:** `sed -i 's/apple/orange/g' fruit.txt`
`sed '1i\new line' file`	Insert 'new line' before the first line **Example:** `sed '1i\new line' file.txt`
`sed '$a\new line' file`	Append 'new line' after the last line **Example:** `sed '$a\new line' file.txt`

awk - Pattern Scanning and Processing Language

Basic awk Syntax

`awk '{print $1}' file` - Print the first field of each line.

Example:

```
awk '{print $1}' data.txt
```

`awk '{print $1, $3}' file` - Print the first and third fields, separated by a space.

Example:

```
awk '{print $1, $3}' data.txt
```

`awk -F',' '{print $1}' file` - Use ',' as the field separator.

Example:

```
awk -F',' '{print $1}' data.csv
```

`awk 'BEGIN {FS=","} {print $1}' file` - Alternative way to set field separator.

Example:

```
awk 'BEGIN {FS=","} {print $1}' data.csv
```

`awk '$1 ~ /pattern/ {print $0}' file` - Print lines where the first field matches 'pattern'.

Example:

```
awk '$1 ~ /error/ {print $0}' logfile.txt
```

`awk '/pattern/ {print $0}' file` - Print lines matching 'pattern' in any field.

Example:

```
awk '/error/ {print $0}' logfile.txt
```

awk Built-in Variables

`NF` Number of fields in the current record.

Example:

```
awk '{print NF}' file.txt
```

`NR` Number of the current record (line number).

Example:

```
awk '{print NR, $0}' file.txt
```

`FS` Field separator (default is whitespace).

Example:

```
awk 'BEGIN {FS=","} {print $1}' file.csv
```

`OFS` Output field separator (default is whitespace).

Example:

```
awk 'BEGIN {OFS="-"} {print $1, $2}' file.txt
```

`ORS` Output record separator (default is newline).

Example:

```
awk 'BEGIN {ORS="\n\n"} {print $0}' file.txt
```

awk - Conditional Statements and Loops

`awk '{if ($1 > 10) print $0}' file` - Print lines where the first field is greater than 10.

Example:

```
awk '{if ($1 > 10) print $0}' data.txt
```

`awk '{if ($1 == "error") print $0}' file` - Print lines where the first field equals "error".

Example:

```
awk '{if ($1 == "error") print $0}' logfile.txt
```

`awk '{for (i=1; i<=NF; i++) print $i}' file` - Print each field on a separate line.

Example:

```
awk '{for (i=1; i<=NF; i++) print $i}' data.txt
```

`awk 'BEGIN {total = 0} {total += $1} END {print "Total: ", total}' file` - Calculate the sum of the first field.

Example:

```
awk 'BEGIN {total = 0} {total += $1} END {print "Total: ", total}' numbers.txt
```

Combining Tools

Piping and Redirection

`grep 'pattern' file | sed 's/old/new/g'` - Find lines matching 'pattern' and then replace 'old' with 'new'.

Example:

```
grep 'error' logfile.txt | sed 's/error/warning/g'
```

`grep 'pattern' file | awk '{print $1}'` - Find lines matching 'pattern' and then print the first field.

Example:

```
grep 'user:' /etc/passwd | awk -F':' '{print $1}'
```

`sed 's/old/new/g' file > output.txt` - Redirect the output of sed to a file.

Example:

```
sed 's/apple/orange/g' fruit.txt > new_fruit.txt
```

`awk '{print $1}' file >> output.txt` - Append the output of awk to a file.

Example:

```
awk '{print $1}' data.txt >> output.txt
```

Complex Examples

Extract all unique IP addresses from a log file:

```
grep -oE '([0-9]{1,3}\.){3}[0-9]{1,3}' logfile.txt | sort -u
```

Count the occurrences of each word in a file:

```
tr -cs 'a-zA-Z0-9' '\n' < file.txt | sort | uniq -c | sort -nr
```

Find lines in a file that do not contain a specific word:

```
grep -v 'specific_word' file.txt
```

Replace all occurrences of a pattern in multiple files:

```
find . -type f -name '*.txt' -exec sed -i 's/old_pattern/new_pattern/g' {} \;
```

Calculate the average of a column in a CSV file:

```
awk -F',' 'BEGIN {sum=0; count=0} {sum+=$1; count++} END {if (count > 0) print "Average = ", sum/count}' data.csv
```