GNU Grep Cheat Sheet

A comprehensive cheat sheet for GNU grep, covering essential options, patterns, and usage examples for efficient text searching in files and streams.

Grep Basics and Usage

Basic Syntax

`grep [OPTIONS] PATTERN [FILE...]`

Searches for PATTERN in each FILE. If no files are specified, grep searches standard input. PATTERN can be a string or a regular expression.

Example:
`grep 'hello' file.txt` - Searches for 'hello' in file.txt

`grep 'error' server.log` - Searches for the word `error` in the `server.log` file.

`grep -i 'warning' config.txt` - Searches for the word `warning` case-insensitively in the `config.txt` file.

Common Options

`-i, --ignore-case`	Ignore case distinctions in both the PATTERN and the input files.
`-v, --invert-match`	Select non-matching lines.
`-c, --count`	Print only a count of matching lines per file.
`-n, --line-number`	Prefix each line of output with the line number within its input file.
`-r, --recursive`	Recursively search directories.
`-l, --files-with-matches`	Print only the names of files containing matches.

Examples with Options

`grep -i 'error' *.log` - Searches for 'error' case-insensitively in all .log files.

`grep -v 'success' app.log` - Shows lines that do NOT contain 'success' in app.log.

`grep -c '404' access.log` - Counts lines containing '404' in access.log.

`grep -n 'function' script.js` - Shows lines containing 'function' with line numbers in script.js.

`grep -r 'TODO' .` - Recursively searches for 'TODO' in the current directory.

Regular Expressions in Grep

Basic Regular Expressions (BRE)

`^`	Matches the beginning of a line. Example: `^hello` matches lines starting with 'hello'.
`$`	Matches the end of a line. Example: `world$` matches lines ending with 'world'.
`.`	Matches any single character. Example: `a.c` matches 'abc', 'aec', etc.
`*`	Matches zero or more occurrences of the preceding character. Example: `ab*c` matches 'ac', 'abc', 'abbc', etc.
`[ ]`	Matches any single character within the brackets. Example: `[aeiou]` matches any vowel.
`[^ ]`	Matches any single character NOT within the brackets. Example: `[^0-9]` matches any non-digit.

Extended Regular Expressions (ERE)

`+`	Matches one or more occurrences of the preceding character. Example: `ab+c` matches 'abc', 'abbc', but not 'ac'.
`?`	Matches zero or one occurrence of the preceding character. Example: `ab?c` matches 'ac' or 'abc'.
`|`	Specifies an alternative. Example: `cat|dog` matches either 'cat' or 'dog'.
`()`	Groups regular expressions. Example: `(ab)+c` matches 'abc', 'ababc', etc.
`{n }`	Matches exactly n occurrences of the preceding character/group. Example: `a{3}` matches 'aaa'.
`{n, m}`	Matches between n and m occurrences of the preceding character/group. Example: `a{1,3}` matches 'a', 'aa', or 'aaa'.

ERE Examples

`grep -E '^(cat|dog)' file.txt` - Finds lines starting with 'cat' or 'dog'.

`grep -E '[0-9]+$' data.txt` - Finds lines ending with one or more digits.

`grep -E 'a(bc)+d' file.txt` - Finds lines containing 'a' followed by one or more 'bc' and then 'd'.

`grep -E 'colou?r' text.txt` - Finds lines containing 'color' or 'colour'.

Advanced Grep Usage

Context Control

`-A NUM, --after-context=NUM`	Print NUM lines of trailing context after matching lines.
`-B NUM, --before-context=NUM`	Print NUM lines of leading context before matching lines.
`-C NUM, --context=NUM`	Print NUM lines of output context.
`--group-separator=SEP`	Use SEP as a group separator. The default is `--`.

File and Directory Options

`-d ACTION, --directories=ACTION`	How to handle directories; ACTION can be read, skip, or recurse.
`--exclude=GLOB`	Skip files matching GLOB.
`--include=GLOB`	Search only files matching GLOB.
`--exclude-dir=GLOB`	Skip directories matching GLOB for recursive searches.

Examples of Context and File Options

`grep -A 2 'error' logfile.txt` - Shows 'error' lines and 2 lines after each match.

`grep -B 1 'warning' code.txt` - Shows 'warning' lines and 1 line before each match.

`grep -C 3 'exception' debug.log` - Shows 'exception' lines and 3 lines of context around each match.

`grep --exclude='*.o' 'main' *` - Searches for 'main' in all files except those ending with '.o'.

`grep --include='*.txt' 'data' .` - Searches for 'data' only in '.txt' files in the current directory.

More Grep Pattern Options

Pattern Control Options

`-e PATTERN, --regexp=PATTERN`	Use PATTERN as the pattern; useful to protect patterns beginning with `-`.
`-f FILE, --file=FILE`	Obtain PATTERN from FILE, one per line.
`-w, --word-regexp`	Select only those lines containing matches that form whole words.
`-x, --line-regexp`	Select only those matches that exactly match the whole line.

Output Control Options

`-m NUM, --max-count=NUM`	Stop reading a file after NUM matching lines.
`-o, --only-matching`	Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.
`-q, --quiet, --silent`	Quiet; do not write anything to standard output. Exit immediately with zero status if any match is found, even if an error was detected.
`--color[=WHEN], --colour[=WHEN]`	Surround the matching string with escape sequences to display it in color; WHEN is `always`, `never`, or `auto`.

Pattern Option Examples

`grep -e '^abc' file.txt` - Searches for lines starting with ‘abc’.

`grep -f patterns.txt data.txt` - Uses patterns from patterns.txt to search data.txt.

`grep -w 'error' logfile.txt` - Searches for the whole word ‘error’ in logfile.txt.

`grep -x 'exact match' file.txt` - Finds lines that exactly match ‘exact match’.

`grep -m 10 'keyword' bigfile.txt` - Stops after finding 10 lines containing ‘keyword’.

`grep -o '[0-9]+' data.txt` - Prints only the matching numbers in data.txt.