grep Command Reference

Global Regular Expression Print · Unix/Linux Text Search Tool

```
grep [options] pattern
```

WHAT IS GREP?

Searches text (files or stdin) for
lines matching a pattern, then prints them.
Supports plain strings and full regex.

BASIC SYNTAX

```
grep [options] pattern [file...]
```

```
grep "error" app.log
grep -i "error" app.log   # case-insensitive
```

COMMON OPTIONS

`-i`	Ignore case
`-r`	Recursive (search dirs)
`-n`	Show line numbers
`-v`	Invert — non-matching lines
`-c`	Count matches
`-w`	Whole word only
`-l`	List matching filenames
`-A 3`	3 lines after match
`-B 3`	3 lines before match
`-E`	Extended regex (ERE)

Mental Model

Input → grep (filter) → Matching lines

CORE USE CASES

Search in a file

```
grep "login failed" server.log
```

Search multiple files

```
grep "TODO" *.js
```

Show line numbers

```
grep -n "error" app.log
```

Count matches

```
grep -c "error" app.log
```

Invert match

```
grep -v "success" app.log
```

Whole word

```
grep -w "cat" file.txt
```

RECURSIVE SEARCH

```
grep -r "function" .       # all files
grep -rl "API_KEY" .       # filenames only
grep -r "saveUser" src/    # in a subdir
```

PIPES (where it shines)

```
ps aux | grep python
tail -f app.log | grep ERROR
cat server.log | grep -i "fail"
history | grep git
```

DEV WORKFLOW EXAMPLES

Find a function in codebase

```
grep -r "function saveUser" .
```

Real-time error monitoring

```
tail -f app.log | grep "ERROR"
```

Find env var usage

```
grep -r "API_KEY" .
```

Search with context

```
grep -A 3 -B 3 "crash" app.log
```

REGEX PATTERNS

`^`	Start of line
`$`	End of line
`.`	Any single character
`*`	0 or more of previous
`+`	1 or more (use -E)
`?`	0 or 1 (use -E)
`[0-9]`	Any digit
`[a-z]`	Any lowercase letter
`\w`	Word character
`\s`	Whitespace
`\b`	Word boundary
`(a|b)`	a or b (use -E)

REGEX EXAMPLES

```
grep "^ERROR" app.log     # starts with ERROR
grep "failed$" app.log    # ends with failed
grep "[0-9]" file.txt     # contains a digit
grep -E "err|warn" log    # err OR warn
grep -E "^[0-9]{3}-" f   # starts 3 digits-
```

PRO TIPS

- Use grep -r instead of opening files manually
- Combine with tail -f for live log debugging
- Learn -A / -B flags for context around matches
- Use -l to quickly find which files match
- For big codebases, try rg (ripgrep) for speed
- Use -E for extended regex (no need for \+, \?)

Alternatives to grep

`rg`	ripgrep — much faster, respects .gitignore
`ag`	the_silver_searcher — fast, .gitignore aware
`ack`	grep-like, optimized for source code