GREP QUICK REFERENCE

Pattern matching, regex, recursive search, context, filtering

Basic Usage

Running grep

```
grep "pattern" file.txt            # search in file
grep "error" *.log                 # search multiple files
grep "hello" file1.txt file2.txt   # explicit file list
cat file.txt | grep "pattern"      # pipe input
dmesg | grep -i "usb"              # filter command output
```

Common Flags

`-i`	Case-insensitive matching
`-v`	Invert match — print non-matching lines
`-c`	Print count of matching lines
`-n`	Show line numbers
`-l`	List filenames with matches only
`-L`	List filenames without matches
`-w`	Match whole words only
`-x`	Match whole lines only

Regex Patterns

Basic Regular Expressions (BRE)

`.`	Any single character
`*`	Zero or more of preceding element
`^`	Start of line
`$`	End of line
`[abc]`	Character class — any of a, b, c
`[^abc]`	Negated class — anything except a, b, c
`[a-z]`	Range — lowercase letters
`\<, \>`	Word boundaries (GNU)
`\( \), \1`	Capture group and back-reference

BRE Examples

```
grep '^#' file.conf                # lines starting with #
grep 'error$' file.log             # lines ending with error
grep '^$' file.txt                 # blank lines
grep 'col[ou]r' file.txt           # match color or colour
```

Extended Regex

Extended Regular Expressions (ERE)

`+`	One or more of preceding element
`?`	Zero or one of preceding element
`{n}`	Exactly n repetitions
`{n,m}`	Between n and m repetitions
`(a|b)`	Alternation — match a or b
`( )`	Grouping (no backslash needed)

ERE Examples

```
grep -E '[0-9]{3}-[0-9]{4}' f     # phone number pattern
grep -E '(error|warn|fatal)' f     # multiple patterns
grep -E '^[A-Z][a-z]+' f           # capitalized words
grep -P '\d{1,3}\.\d{1,3}' f      # Perl regex: IP fragments
```

Context Lines

Context Examples

```
grep -B 3 "error" app.log          # 3 lines before match
grep -A 5 "FAIL" test.log          # 5 lines after match
grep -C 2 "crash" kern.log         # 2 lines before and after
grep --group-separator="---" -C 1 "err" f  # custom separator
```

Context Flags

`-B N`	Show N lines before each match
`-A N`	Show N lines after each match
`-C N`	Show N lines before and after (context)
`--group-separator=str`	Separator between match groups (default `--`)
`--color=auto`	Highlight matches in terminal

Recursive Search

Recursive Examples

```
grep -r "TODO" .                   # recursive from current dir
grep -rn "FIXME" src/              # recursive with line numbers
grep -r --include="*.py" "import" .  # only .py files
grep -r --exclude="*.log" "error" .  # skip .log files
grep -r --exclude-dir=node_modules "require" .
```

Recursive Flags

`-r / --recursive`	Search directories recursively
`-R`	Like `-r` but follows symlinks
`--include=glob`	Search only files matching glob
`--exclude=glob`	Skip files matching glob
`--exclude-dir=dir`	Skip directories matching name
`--include-dir=dir`	Only search directories matching name

Counting & Listing

Count & List Examples

```
grep -c "error" *.log              # count matches per file
grep -l "TODO" src/*.py            # list files with TODOs
grep -L "test" src/*.py            # files missing "test"
grep -o "http[^ ]*" page.html      # extract matching parts only
grep -c '' file.txt                # count total lines (like wc -l)
```

Output Flags

`-c`	Print count of matching lines per file
`-l`	Print only names of files with matches
`-L`	Print only names of files without matches
`-o`	Print only the matched parts of lines
`-H / -h`	Show / hide filename prefix
`-Z`	Null-delimited output (for xargs -0)

Inverse Match

Invert & Exclude

```
grep -v "^#" config.conf           # remove comment lines
grep -v "^$" file.txt              # remove blank lines
grep -v -e "debug" -e "trace" app.log  # exclude two patterns
grep -v "pattern" f | grep "other" # chain: NOT A, then B
```

Filtering Strategies

`-v`	Invert match — select non-matching lines
`-v with -e`	Exclude multiple patterns
`pipe chain`	Chain grep calls for complex filtering
`grep -v '^$' | grep -v '^#'`	Remove blanks and comments
`-v with -c`	Count non-matching lines

Multiple Patterns

Multiple Pattern Examples

```
grep -e "error" -e "warning" app.log
grep -E "error|warning|fatal" app.log
grep -f patterns.txt file.txt      # patterns from file
grep -w -e "GET" -e "POST" access.log
```

Pattern Options

`-e pattern`	Specify a pattern (use multiple times)
`-f file`	Read patterns from file (one per line)
`-E 'a|b|c'`	ERE alternation for multiple patterns
`-F`	Fixed strings — no regex, faster matching
`-G`	Basic regex (default mode)
`-P`	Perl-compatible regex (PCRE)

Performance

Performance Tips

`-F (fgrep)`	Fixed-string mode — fastest for literal strings
`LC_ALL=C grep`	Bypass locale for 2-10x speedup on ASCII data
`--include/--exclude`	Reduce files searched before opening
`-m N`	Stop after N matches per file
`-q`	Quiet mode — exit on first match (for scripts)
`ripgrep (rg)`	Drop-in replacement; faster on large repos

Performance Examples

```
LC_ALL=C grep -F "exact string" huge.log
grep -r -m 1 "needle" /var/log/    # stop after first hit
grep -rq "pattern" . && echo "found" # boolean test
grep -r --include="*.go" "func main" .
```

Common Patterns

One-Liners

```
grep -rn "TODO\|FIXME\|HACK" src/      # find code markers
grep -oP '(?<=")[^"]+(?=")' f          # extract quoted strings
grep -E '^\s*$' f | wc -l              # count blank lines
grep -c '' *.py | sort -t: -k2 -rn     # sort files by line count
grep -rn --include="*.yaml" "password" . # audit for secrets
```

Recipes

`IP addresses`	`grep -oE '[0-9]{1,3}(\.[0-9]{1,3}){3}'`
`Email addresses`	`grep -oE '[a-zA-Z0-9._%+-]+@[a-z.-]+'`
`URLs`	`grep -oE 'https?://[^ ]+'`
`Lines between markers`	`grep -A999 'START' f | grep -B999 'END'`
`Unique matches`	`grep -oE 'pattern' f | sort -u`
`Count per pattern`	`grep -c 'pat1' f; grep -c 'pat2' f`