grep Search text lines

CONCEPTS

grep selects whole lines. Add -o to print only the matched part.

Quote patterns in single quotes so the shell leaves *, $ and | alone.

Exit status: 0 = a line matched, 1 = none, 2 = error. Handy in if.

MATCHING

`grep "<pattern>" <file>`
Lines matching a pattern

`grep -i "<pattern>" <file>`
Ignore case

`grep -v "<pattern>" <file>`
Invert: lines that do not match

`grep -w "<word>" <file>`
Match whole words only

`grep -F "<text>" <file>`
Fixed string, no regex

`grep -E "<a>|<b>" <file>`
Extended regex, alternation

`grep -e <a> -e <b> <file>`
Several patterns

`grep -P "<regex>" <file>`
Perl-style regex

OUTPUT

`grep -n "<pattern>" <file>`
Prefix line numbers

`grep -c "<pattern>" <file>`
Count matching lines

`grep -l "<pattern>" <files>`
List files with a match

`grep -o "<regex>" <file>`
Print only the matched text

`grep -q "<pattern>" <file>`
Silent, exit status only

CONTEXT

`grep -A <n> "<pattern>" <file>`
n lines after each match

`grep -B <n> "<pattern>" <file>`
n lines before each match

`grep -C <n> "<pattern>" <file>`
n lines on both sides

FILES & RECURSION

`grep -r "<pattern>" <dir>`
Search a directory tree

`grep -r --include="*.py" "<p>" <dir>`
Only files matching a glob

`grep -r --exclude-dir=<dir> "<p>" .`
Skip a directory

REGEX SYNTAX

Basic building blocks

Syntax	Matches
`.`	any one character
`*`	zero or more of previous
`^ $`	start / end of line
`[abc] [^abc]`	one of / none of
`[a-z]`	a range
`[[:digit:]]`	POSIX class
`\< \>`	word start / end
`\b`	word boundary

ERE (-E) additions

Syntax	Matches
`a|b`	a or b
`( )`	group
`+`	one or more
`?`	zero or one
`{n,m}`	n to m repeats
`\1`	back-reference to group 1

BRE vs ERE
In plain grep (BRE) write \| \+ \? \(\) \{ \} for the special forms. -E drops the backslashes.