A VISUAL REFERENCE · grep

grep,
one change at a time

Select lines, inspect context and search files while keeping matching separate from output. One small input. One focused change. Its actual result.

The canonical Org file owns the programs and synthetic fixtures. Each complete standalone Bash script recreates its exact inputs in an owned temporary directory. The final PDF is versioned at the repository root.

> **Last changes: 2026-11-01.** First edition: 84 deliberately small examples, complete inputs, authored expected outcomes, BSD/GNU comparisons and a separate GNU-extension chapter.

84
worked examples

14
topic families

1
canonical Org source

READING THE TOUR

Follow one change

Read the input

All files consumed by a command are shown completely. Small inputs make every selected, changed or omitted line visible.

Change one feature

Follow the comparison link. Read the focused command and its named switch or instruction before adding another feature.

Check the actual result

Stdout and the complete driver status are checked against independently authored expectations. Empty stdout is displayed explicitly.

The gallery, search, topic filter and stable GRP IDs support quick lookup. Copy the focused command while using the shown fixture, or download the complete script to recreate that fixture. The complete script runs with LC_ALL=C in a fresh directory and removes that owned directory on exit. In file-editing examples, only those reconstructed local copies are changed.

Implementations and reproducibility

The common examples are tested with macOS BSD grep and GNU grep. Common means supported by those tested implementations, not that every option is specified by POSIX. The Implementation row identifies the GNU-only examples; they explicitly use ggrep. Their complete scripts select ggrep when installed, or provide that command name using an ordinary grep executable verified as GNU. The focused GNU commands can use the ordinary name on a GNU/Linux installation. Versions and binary identities participate in the execution fingerprint.

The outer captured status belongs to the Bash driver. Expected failures are caught and their tool statuses are printed explicitly. Implementation-specific diagnostic prose is not a golden output. No stdout means no output bytes, not a missing execution. All fixtures are original invented teaching data.

Related tours and primary references

- Regular-expression tour: BRE, ERE, captures and matching operations
- Bash tour: quoting, pipelines, redirections and exit statuses
- sed: transforming selected text
- GNU grep manual
- POSIX grep specification

The examples, explanations, fixtures and expected outcomes are independently authored. The references establish command and dialect semantics; implementation extensions are labeled separately.

01 - Select your first lines

Keep the same color list while changing one matching option at a time.

GRP-01-001

Select a line containing a word

A match anywhere on a line selects that whole line.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Code

```
grep 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
blueberry
sky blue
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`blue`	Search for these characters anywhere on the line.

Download: GRP-01-001.sh

GRP-01-002

Ignore case

Add only -i to include the capitalized spelling.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-01-001 · Select a line containing a word.

Code

```
grep -i 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
Blue
blueberry
sky blue
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-i`	Ignore letter case for this ASCII input.

Download: GRP-01-002.sh

GRP-01-003

Require a whole word

Return to case-sensitive matching and add -w. The word can still have surrounding text.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-01-001 · Select a line containing a word.

Code

```
grep -w 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
sky blue
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-w`	Require word boundaries around the match.

Download: GRP-01-003.sh

GRP-01-004

Require the whole line

Change -w to -x. Surrounding words now reject a line.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-01-003 · Require a whole word.

Code

```
grep -x 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-x`	Require the pattern to match the complete input line.

Download: GRP-01-004.sh

GRP-01-005

Invert the selection

Change the base search with -v to print its nonmatching lines.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-01-001 · Select a line containing a word.

Code

```
grep -v 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
Blue
red
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-v`	Select lines that do not match.

Download: GRP-01-005.sh

GRP-01-006

Show original line numbers

Add presentation information without changing which lines match.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-01-001 · Select a line containing a word.

Code

```
grep -n 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
1:blue
3:blueberry
4:sky blue
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-n`	Prefix matches with one-based input line numbers.

Download: GRP-01-006.sh

02 - Anchor a basic regular expression

Use simple ASCII inputs to see exactly what each BRE atom and anchor permits.

GRP-02-001

Match a literal substring

The unanchored pattern also selects longer words containing cat.

Input

data/words.txt

```
cat
scatter
bobcat
cat.
cot
cut
dog
```

Code

```
grep 'cat' data/words.txt
```

Actual output

Captured run · exit 0

```
cat
scatter
bobcat
cat.
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`cat`	Ordinary letters match themselves.

Download: GRP-02-001.sh

GRP-02-002

Anchor the start

A caret removes matches that begin later in a line.

Input

data/words.txt

```
cat
scatter
bobcat
cat.
cot
cut
dog
```

Compare with GRP-02-001 · Match a literal substring.

Code

```
grep '^cat' data/words.txt
```

Actual output

Captured run · exit 0

```
cat
cat.
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`^`	Require the beginning of the line.

Download: GRP-02-002.sh

GRP-02-003

Anchor the end

Use the end anchor instead of the start anchor.

Input

data/words.txt

```
cat
scatter
bobcat
cat.
cot
cut
dog
```

Compare with GRP-02-002 · Anchor the start.

Code

```
grep 'cat$' data/words.txt
```

Actual output

Captured run · exit 0

```
cat
bobcat
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`$`	Require the end of the input line.

Download: GRP-02-003.sh

GRP-02-004

Anchor both ends

Put both boundaries around the same literal word.

Input

data/words.txt

```
cat
scatter
bobcat
cat.
cot
cut
dog
```

Compare with GRP-02-003 · Anchor the end.

Code

```
grep '^cat$' data/words.txt
```

Actual output

Captured run · exit 0

```
cat
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`^cat$`	Consume exactly this complete line.

Download: GRP-02-004.sh

GRP-02-005

Allow one character at the dot

Change the middle letter to the BRE wildcard.

Input

data/words.txt

```
cat
scatter
bobcat
cat.
cot
cut
dog
```

Compare with GRP-02-004 · Anchor both ends.

Code

```
grep '^c.t$' data/words.txt
```

Actual output

Captured run · exit 0

```
cat
cot
cut
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`.`	Match one character; grep still processes one line at a time.

Download: GRP-02-005.sh

GRP-02-006

Match a period literally

A bracket expression avoids an additional backslash layer.

Input

data/words.txt

```
cat
scatter
bobcat
cat.
cot
cut
dog
```

Compare with GRP-02-005 · Allow one character at the dot.

Code

```
grep '^cat[.]$' data/words.txt
```

Actual output

Captured run · exit 0

```
cat.
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`[.]`	Treat the period as literal punctuation.

Download: GRP-02-006.sh

03 - Choose extended regex notation

Switch to ERE explicitly before adding alternatives, grouping or interval quantifiers.

GRP-03-001

Turn on extended syntax

An ERE alternative matches either word anywhere on a line.

Input

data/animals.txt

```
cat
dog
cats
dogs
catfish
```

Code

```
grep -E 'cat|dog' data/animals.txt
```

Actual output

Captured run · exit 0

```
cat
dog
cats
dogs
catfish
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-E`	Enable ERE syntax before using alternation.

Download: GRP-03-001.sh

GRP-03-002

Group alternatives before anchoring

Only the two complete singular names remain.

Input

data/animals.txt

```
cat
dog
cats
dogs
catfish
```

Compare with GRP-03-001 · Turn on extended syntax.

Code

```
grep -E '^(cat|dog)$' data/animals.txt
```

Actual output

Captured run · exit 0

```
cat
dog
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`(...)`	Apply both anchors to the grouped alternatives.

Download: GRP-03-002.sh

GRP-03-003

Make a suffix optional

Add an optional s after the same grouped alternatives.

Input

data/animals.txt

```
cat
dog
cats
dogs
catfish
```

Compare with GRP-03-002 · Group alternatives before anchoring.

Code

```
grep -E '^(cat|dog)s?$' data/animals.txt
```

Actual output

Captured run · exit 0

```
cat
dog
cats
dogs
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`?`	Allow the preceding s zero or one time.

Download: GRP-03-003.sh

GRP-03-004

Require one or more digits

Select a complete AB identifier using a repeated ASCII class.

Input

data/ids.txt

```
AB-7
AB-42
AB-123
CD-007
xAB-123
```

Code

```
grep -E '^AB-[0-9]+$' data/ids.txt
```

Actual output

Captured run · exit 0

```
AB-7
AB-42
AB-123
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`+`	Require at least one digit.

Download: GRP-03-004.sh

GRP-03-005

Require exactly three digits

Replace plus with an exact repetition interval.

Input

data/ids.txt

```
AB-7
AB-42
AB-123
CD-007
xAB-123
```

Compare with GRP-03-004 · Require one or more digits.

Code

```
grep -E '^AB-[0-9]{3}$' data/ids.txt
```

Actual output

Captured run · exit 0

```
AB-123
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`{3}`	Require exactly three digits.

Download: GRP-03-005.sh

GRP-03-006

Write the same interval as BRE

Remove -E and use BRE interval escaping. The accepted identifier stays the same.

Input

data/ids.txt

```
AB-7
AB-42
AB-123
CD-007
xAB-123
```

Compare with GRP-03-005 · Require exactly three digits.

Code

```
grep '^AB-[0-9]\{3\}$' data/ids.txt
```

Actual output

Captured run · exit 0

```
AB-123
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`\{3\}`	BRE notation for the same repetition interval.

Download: GRP-03-006.sh

04 - Use fixed strings and pattern lists

Literal patterns and pattern files are selection rules, not shell code or configuration interpolation.

GRP-04-001

Search a metacharacter literally

Fixed-string mode makes plus an ordinary character.

Input

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Code

```
grep -F 'a+b' data/literals.txt
```

Actual output

Captured run · exit 0

```
a+b
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-F`	Interpret the pattern as literal text.

Download: GRP-04-001.sh

GRP-04-002

Add another literal term

Two -e options select the union of matching lines.

Input

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Compare with GRP-04-001 · Search a metacharacter literally.

Code

```
grep -F -e 'a+b' -e 'a.b' data/literals.txt
```

Actual output

Captured run · exit 0

```
a+b
a.b
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-e`	Add an alternative pattern without regex alternation.

Download: GRP-04-002.sh

GRP-04-003

Move the terms into a pattern file

One term per line has the same OR meaning as the preceding -e options.

Input

data/terms.txt

```
a+b
a.b
```

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Code

```
grep -Ff data/terms.txt data/literals.txt
```

Actual output

Captured run · exit 0

```
a+b
a.b
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-f`	Read one pattern per line from the named file.

Download: GRP-04-003.sh

GRP-04-004

Observe an empty pattern

The empty literal occurs on every line, so every line is selected.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Code

```
grep -F '' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
Blue
blueberry
sky blue
red
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`''`	An empty pattern is a pattern that matches everywhere.

Download: GRP-04-004.sh

GRP-04-005

Distinguish an empty pattern file

A file with zero patterns selects no lines. The driver catches status 1.

Input

data/none.txt

```
(empty file)
```

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Code

```
status=0
grep -Ff data/none.txt data/colors.txt || status=$?
printf 'exit=%s\n' "$status"
```

Actual output

Captured run · exit 0

```
exit=1
```

Parameter	Meaning in this example
Implementation	BSD/GNU common
Zero patterns	An empty file differs from a file containing an empty pattern.

Download: GRP-04-005.sh

GRP-04-006

Protect a pattern beginning with a dash

Use -e so a leading dash is interpreted as pattern data.

Input

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Code

```
grep -F -e '-error' data/literals.txt
```

Actual output

Captured run · exit 0

```
-error
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-e PATTERN`	Separate a pattern from option parsing.

Download: GRP-04-006.sh

05 - Choose what grep prints

Matching and presentation are different decisions: line numbers, filename prefixes, file lists and occurrences.

GRP-05-001

Show line numbers on log records

Match ERROR at the beginning and retain source positions.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Code

```
grep -n '^ERROR' data/events.txt
```

Actual output

Captured run · exit 0

```
3:ERROR disk
5:ERROR network
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-n`	Keep original one-based line numbers.

Download: GRP-05-001.sh

GRP-05-002

Force a filename prefix

-H adds a filename even with one input file.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Compare with GRP-05-001 · Show line numbers on log records.

Code

```
grep -H '^ERROR' data/events.txt
```

Actual output

Captured run · exit 0

```
data/events.txt:ERROR disk
data/events.txt:ERROR network
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-H`	Always print a filename prefix.

Download: GRP-05-002.sh

GRP-05-003

Suppress prefixes across files

The -h option removes automatic multi-file filename labels.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/b.log

```
WARN beta
ERROR network
```

Code

```
grep -h '^ERROR' data/a.log data/b.log
```

Actual output

Captured run · exit 0

```
ERROR disk
ERROR network
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-h`	Print matching text without filenames.

Download: GRP-05-003.sh

GRP-05-004

List only files containing a match

Each matching file is listed once, regardless of matching-line count.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/b.log

```
WARN beta
ERROR network
```

data/clean.log

```
INFO quiet
INFO ready
```

Code

```
grep -l '^ERROR' data/a.log data/clean.log data/b.log
```

Actual output

Captured run · exit 0

```
data/a.log
data/b.log
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-l`	Print only names of files with at least one selected line.

Download: GRP-05-004.sh

GRP-05-005

List files with no match

Switch from -l to -L using the same file list.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/clean.log

```
INFO quiet
INFO ready
```

data/b.log

```
WARN beta
ERROR network
```

Compare with GRP-05-004 · List only files containing a match.

Code

```
grep -L '^ERROR' data/a.log data/clean.log data/b.log
```

Actual output

Captured run · exit 0

```
data/clean.log
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-L`	Print names of files without a selected line.

Download: GRP-05-005.sh

GRP-05-006

Print the matching portions

The common GNU/BSD -o extension prints three occurrences on separate output lines.

Input

data/counts.txt

```
id=7 id=42
none
id=5
```

Code

```
grep -Eo 'id=[0-9]+' data/counts.txt
```

Actual output

Captured run · exit 0

```
id=7
id=42
id=5
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-o`	Print each nonempty matching portion, not the whole line.

Download: GRP-05-006.sh

06 - Count, stop and interpret status

Limit matching lines separately from counting them. Expected no-match and error statuses are caught explicitly.

GRP-06-001

Count matching lines

Multiple occurrences on the first line contribute only one matching line.

Input

data/counts.txt

```
id=7 id=42
none
id=5
```

Compare with GRP-05-006 · Print the matching portions.

Code

```
grep -Ec 'id=[0-9]+' data/counts.txt
```

Actual output

Captured run · exit 0

```
2
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-c`	Count selected lines rather than occurrences.

Download: GRP-06-001.sh

GRP-06-002

Stop after the first selected line

The match limit counts lines, not individual substrings.

Input

data/counts.txt

```
id=7 id=42
none
id=5
```

Compare with GRP-06-001 · Count matching lines.

Code

```
grep -m 1 'id=' data/counts.txt
```

Actual output

Captured run · exit 0

```
id=7 id=42
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-m 1`	Stop reading this file after one selected line.

Download: GRP-06-002.sh

GRP-06-003

Allow two selected lines

Increase only the line limit.

Input

data/counts.txt

```
id=7 id=42
none
id=5
```

Compare with GRP-06-002 · Stop after the first selected line.

Code

```
grep -m 2 'id=' data/counts.txt
```

Actual output

Captured run · exit 0

```
id=7 id=42
id=5
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-m 2`	Allow two selected lines from this input.

Download: GRP-06-003.sh

GRP-06-004

Ask a quiet yes-or-no question

Quiet matching writes no match data. The shell prints the branch chosen by grep status.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Code

```
if grep -q '^ERROR' data/events.txt; then
  printf 'errors found\n'
else
  printf 'no errors\n'
fi
```

Actual output

Captured run · exit 0

```
errors found
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-q`	Stop at a successful match and communicate through status.

Download: GRP-06-004.sh

GRP-06-005

Capture a no-match status

Status 1 is an expected search outcome, not a syntax or file error.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Compare with GRP-06-004 · Ask a quiet yes-or-no question.

Code

```
status=0
grep '^NEVER$' data/events.txt || status=$?
printf 'no match: exit=%s\n' "$status"
```

Actual output

Captured run · exit 0

```
no match: exit=1
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Exit 1`	A valid search selected no lines.

Download: GRP-06-005.sh

GRP-06-006

Capture a file-error status

The missing path is inside the owned directory. The diagnostic is kept local because wording differs across implementations.

Code

```
status=0
grep 'anything' absent.txt 2>diagnostic.txt || status=$?
printf 'file error: exit=%s\n' "$status"
```

Actual output

Captured run · exit 0

```
file error: exit=2
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Exit 2`	Report a tool error separately from a no-match result.

Download: GRP-06-006.sh

07 - Inspect surrounding context

Add one context option at a time. Context lines are not additional matching lines.

GRP-07-001

Show one line after a match

Only the first matching record is used to keep the context change visible.

Input

data/context.txt

```
one
two
HIT first
four
five
six
HIT last
eight
nine
```

Code

```
grep -A 1 'HIT first' data/context.txt
```

Actual output

Captured run · exit 0

```
HIT first
four
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-A 1`	Append one following context line.

Download: GRP-07-001.sh

GRP-07-002

Show one line before a match

Change -A to -B.

Input

data/context.txt

```
one
two
HIT first
four
five
six
HIT last
eight
nine
```

Compare with GRP-07-001 · Show one line after a match.

Code

```
grep -B 1 'HIT first' data/context.txt
```

Actual output

Captured run · exit 0

```
two
HIT first
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-B 1`	Prepend one preceding context line.

Download: GRP-07-002.sh

GRP-07-003

Show context on both sides

-C combines the same-sized before and after windows.

Input

data/context.txt

```
one
two
HIT first
four
five
six
HIT last
eight
nine
```

Compare with GRP-07-002 · Show one line before a match.

Code

```
grep -C 1 'HIT first' data/context.txt
```

Actual output

Captured run · exit 0

```
two
HIT first
four
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-C 1`	Show one context line on either side.

Download: GRP-07-003.sh

GRP-07-004

Distinguish numbered context from matches

With -n, a colon labels a matching line and a dash labels context.

Input

data/context.txt

```
one
two
HIT first
four
five
six
HIT last
eight
nine
```

Compare with GRP-07-003 · Show context on both sides.

Code

```
grep -nC 1 'HIT first' data/context.txt
```

Actual output

Captured run · exit 0

```
2-two
3:HIT first
4-four
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Number prefixes`	Read matching and context line labels differently.

Download: GRP-07-004.sh

GRP-07-005

See overlapping windows merge

Adjacent matches share their context, so the same source line is not printed twice.

Input

data/near.txt

```
one
HIT a
HIT b
four
five
```

Code

```
grep -C 1 'HIT' data/near.txt
```

Actual output

Captured run · exit 0

```
one
HIT a
HIT b
four
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Merged context`	Overlapping context windows form one output group.

Download: GRP-07-005.sh

GRP-07-006

See separated context groups

Distant matches retain separate groups with the normal separator.

Input

data/context.txt

```
one
two
HIT first
four
five
six
HIT last
eight
nine
```

Code

```
grep -C 1 'HIT' data/context.txt
```

Actual output

Captured run · exit 0

```
two
HIT first
four
--
six
HIT last
eight
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`--`	Separate nonadjacent context groups.

Download: GRP-07-006.sh

08 - Read files and standard input

Keep explicit input ordering and distinguish file arguments, redirected stdin and a pipeline.

GRP-08-001

Search two explicitly ordered files

Automatic prefixes identify each source. Explicit argument order controls the shown sequence.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/b.log

```
WARN beta
ERROR network
```

Code

```
grep '^ERROR' data/a.log data/b.log
```

Actual output

Captured run · exit 0

```
data/a.log:ERROR disk
data/b.log:ERROR network
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`File arguments`	Process the files in the provided argument order.

Download: GRP-08-001.sh

GRP-08-002

Count each input file separately

-c produces one per-file count with the normal multi-file prefix.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/b.log

```
WARN beta
ERROR network
```

Compare with GRP-08-001 · Search two explicitly ordered files.

Code

```
grep -c '^ERROR' data/a.log data/b.log
```

Actual output

Captured run · exit 0

```
data/a.log:1
data/b.log:1
```

Parameter	Meaning in this example
Implementation	BSD/GNU common
Per-file count	Do not mistake several per-file counts for a single total.

Download: GRP-08-002.sh

GRP-08-003

Read the same log through redirection

With no filename argument, grep consumes standard input and has no filename label.

Input

data/a.log

```
INFO alpha
ERROR disk
```

Code

```
grep '^ERROR' < data/a.log
```

Actual output

Captured run · exit 0

```
ERROR disk
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`< FILE`	Let the shell connect a file to standard input.

Download: GRP-08-003.sh

GRP-08-004

Read standard input from a pipeline

cat supplies the same bytes as the previous redirection. The extra process is used here only to make the pipeline visible.

Input

data/a.log

```
INFO alpha
ERROR disk
```

Compare with GRP-08-003 · Read the same log through redirection.

Code

```
cat data/a.log | grep '^ERROR'
```

Actual output

Captured run · exit 0

```
ERROR disk
```

Parameter	Meaning in this example
Implementation	BSD/GNU common
Pipeline input	grep reads bytes produced by the left-hand command.

Download: GRP-08-004.sh

GRP-08-005

Name standard input with a dash

A dash explicitly inserts stdin into the file argument list. -h keeps the display independent of stdin label wording.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/b.log

```
WARN beta
ERROR network
```

Code

```
grep -h '^ERROR' data/a.log - < data/b.log
```

Actual output

Captured run · exit 0

```
ERROR disk
ERROR network
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-`	Read stdin at this position in the input list.

Download: GRP-08-005.sh

GRP-08-006

Reverse the explicit input order

Changing only the file order changes the order of the matching records.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/b.log

```
WARN beta
ERROR network
```

Compare with GRP-08-005 · Name standard input with a dash.

Code

```
grep -h '^ERROR' data/b.log data/a.log
```

Actual output

Captured run · exit 0

```
ERROR network
ERROR disk
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Argument order`	grep does not sort an explicit list of file arguments.

Download: GRP-08-006.sh

09 - Search a directory tree

The fixtures form a tiny tree. Sorting recursive output makes presentation deterministic; it does not define traversal order.

GRP-09-001

Recurse into a tiny directory tree

The final sort stabilizes presentation; filesystem traversal order itself is not promised.

Input

data/tree/a.txt

```
target alpha
quiet
```

data/tree/b.tmp

```
target cache
```

data/tree/sub/c.txt

```
target gamma
```

data/tree/cache/d.txt

```
target saved
```

Code

```
grep -r 'target' data/tree | sort
```

Actual output

Captured run · exit 0

```
data/tree/a.txt:target alpha
data/tree/b.tmp:target cache
data/tree/cache/d.txt:target saved
data/tree/sub/c.txt:target gamma
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-r`	Recurse through this tree of ordinary files.

Download: GRP-09-001.sh

GRP-09-002

List recursive matching filenames

Replace matched records with file names while retaining deterministic display.

Input

data/tree/a.txt

```
target alpha
quiet
```

data/tree/b.tmp

```
target cache
```

data/tree/sub/c.txt

```
target gamma
```

data/tree/cache/d.txt

```
target saved
```

Compare with GRP-09-001 · Recurse into a tiny directory tree.

Code

```
grep -rl 'target' data/tree | sort
```

Actual output

Captured run · exit 0

```
data/tree/a.txt
data/tree/b.tmp
data/tree/cache/d.txt
data/tree/sub/c.txt
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-rl`	Combine recursion with one name per matching file.

Download: GRP-09-002.sh

GRP-09-003

Include only text filenames

The quoted glob is interpreted by grep rather than expanded by the shell.

Input

data/tree/a.txt

```
target alpha
quiet
```

data/tree/b.tmp

```
target cache
```

data/tree/sub/c.txt

```
target gamma
```

data/tree/cache/d.txt

```
target saved
```

Compare with GRP-09-002 · List recursive matching filenames.

Code

```
grep -rl --include='*.txt' 'target' data/tree | sort
```

Actual output

Captured run · exit 0

```
data/tree/a.txt
data/tree/cache/d.txt
data/tree/sub/c.txt
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`--include`	Filter recursive input names using this glob.

Download: GRP-09-003.sh

GRP-09-004

Exclude temporary filenames

Use exclusion to remove only the temporary-file case.

Input

data/tree/a.txt

```
target alpha
quiet
```

data/tree/b.tmp

```
target cache
```

data/tree/sub/c.txt

```
target gamma
```

data/tree/cache/d.txt

```
target saved
```

Compare with GRP-09-002 · List recursive matching filenames.

Code

```
grep -rl --exclude='*.tmp' 'target' data/tree | sort
```

Actual output

Captured run · exit 0

```
data/tree/a.txt
data/tree/cache/d.txt
data/tree/sub/c.txt
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`--exclude`	Skip filenames matching the glob.

Download: GRP-09-004.sh

GRP-09-005

Skip the cache directory

A directory exclusion removes its subtree, not every file whose content mentions cache.

Input

data/tree/a.txt

```
target alpha
quiet
```

data/tree/b.tmp

```
target cache
```

data/tree/sub/c.txt

```
target gamma
```

data/tree/cache/d.txt

```
target saved
```

Compare with GRP-09-004 · Exclude temporary filenames.

Code

```
grep -rl --exclude-dir=cache 'target' data/tree | sort
```

Actual output

Captured run · exit 0

```
data/tree/a.txt
data/tree/b.tmp
data/tree/sub/c.txt
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`--exclude-dir`	Skip directories matching this name.

Download: GRP-09-005.sh

GRP-09-006

Combine two independent exclusions

Add the temporary-file exclusion to the cache-directory exclusion.

Input

data/tree/a.txt

```
target alpha
quiet
```

data/tree/b.tmp

```
target cache
```

data/tree/sub/c.txt

```
target gamma
```

data/tree/cache/d.txt

```
target saved
```

Compare with GRP-09-005 · Skip the cache directory.

Code

```
grep -rl --exclude-dir=cache --exclude='*.tmp' 'target' data/tree | sort
```

Actual output

Captured run · exit 0

```
data/tree/a.txt
data/tree/sub/c.txt
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Two exclusions`	Apply both filename and directory selection rules.

Download: GRP-09-006.sh

10 - Make case, word and line rules explicit

Whole words, whole lines, literal strings and locale-dependent classes answer different questions.

GRP-10-001

Compare word and substring selection

A whole-word test excludes blueberry but retains sky blue.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-01-001 · Select a line containing a word.

Code

```
grep -w 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
sky blue
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-w`	Match a complete word inside a potentially longer line.

Download: GRP-10-001.sh

GRP-10-002

Add case folding to the word test

Only case sensitivity changes.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-10-001 · Compare word and substring selection.

Code

```
grep -iw 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
Blue
sky blue
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-iw`	Combine case-insensitive and whole-word matching.

Download: GRP-10-002.sh

GRP-10-003

Use literal whole-line matching

The fixed-string mode and whole-line rule solve different parts of this search.

Input

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Code

```
grep -Fx 'a+b' data/literals.txt
```

Actual output

Captured run · exit 0

```
a+b
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-Fx`	Use a complete literal line as the matching rule.

Download: GRP-10-003.sh

GRP-10-004

Select complete ASCII alphabetic lines

The driver fixes LC_ALL=C, so the class is interpreted under that locale.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Code

```
grep -E '^[[:alpha:]]+$' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
Blue
blueberry
red
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`[[:alpha:]]`	Use the alphabetic class in the selected locale.

Download: GRP-10-004.sh

GRP-10-005

Exclude uppercase ASCII spellings

An explicit lowercase range removes Blue from the previous result.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-10-004 · Select complete ASCII alphabetic lines.

Code

```
grep -E '^[a-z]+$' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
blueberry
red
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`[a-z]`	Constrain the accepted ASCII spelling in the C locale.

Download: GRP-10-005.sh

GRP-10-006

Remember that a word is not a field parser

The word boundary accepts blue inside sky blue. A whole-line rule can enforce the intended complete value.

Input

data/colors.txt

```
blue
Blue
blueberry
sky blue
red
```

Compare with GRP-10-001 · Compare word and substring selection.

Code

```
grep -wx 'blue' data/colors.txt
```

Actual output

Captured run · exit 0

```
blue
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-wx`	Add the complete-line constraint to a whole-word search.

Download: GRP-10-006.sh

11 - Quote patterns and handle filenames

Preserve literal spaces, dollar signs and metacharacters. All filename copies are inside the owned example directory.

GRP-11-001

Preserve a literal space in the pattern

Single shell quotes keep the two words in one pattern argument.

Input

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Code

```
grep -F 'red blue' data/literals.txt
```

Actual output

Captured run · exit 0

```
red blue
```

Parameter	Meaning in this example
Implementation	BSD/GNU common
Shell quotes	Preserve the exact single space between the words.

Download: GRP-11-001.sh

GRP-11-002

Preserve a literal dollar sign

Single quotes prevent shell expansion of HOME in the search term.

Input

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Compare with GRP-11-001 · Preserve a literal space in the pattern.

Code

```
grep -F '$HOME' data/literals.txt
```

Actual output

Captured run · exit 0

```
home=$HOME
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`'$HOME'`	Pass the literal dollar sign and letters to grep.

Download: GRP-11-002.sh

GRP-11-003

Search a literal term stored in a variable

Quote the expansion so one variable value becomes one argument.

Input

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Compare with GRP-11-002 · Preserve a literal dollar sign.

Code

```
term='a+b'
grep -F "$term" data/literals.txt
```

Actual output

Captured run · exit 0

```
a+b
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`"$term"`	Keep variable data in one literal-pattern argument.

Download: GRP-11-003.sh

GRP-11-004

Read a filename containing spaces

Create a local copy with the teaching filename, then quote it as one file argument.

Input

data/a.log

```
INFO alpha
ERROR disk
```

Code

```
cp data/a.log 'notes with spaces.txt'
grep '^ERROR' 'notes with spaces.txt'
```

Actual output

Captured run · exit 0

```
ERROR disk
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Quoted filename`	Keep a filename with spaces in one argument.

Download: GRP-11-004.sh

GRP-11-005

End option parsing before a filename

The reconstructed copy begins with a dash. The option terminator prevents it from becoming an option.

Input

data/a.log

```
INFO alpha
ERROR disk
```

Compare with GRP-11-004 · Read a filename containing spaces.

Code

```
cp data/a.log ./-notes.txt
grep '^ERROR' -- -notes.txt
```

Actual output

Captured run · exit 0

```
ERROR disk
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`--`	Terminate option parsing before a leading-dash filename.

Download: GRP-11-005.sh

GRP-11-006

Protect both a pattern and a filename

Use -e for the leading-dash pattern and -- for the leading-dash input filename.

Input

data/literals.txt

```
a+b
aaab
a.b
aXb
red blue
red  blue
home=$HOME
-error
```

Code

```
cp data/literals.txt ./-notes.txt
grep -F -e '-error' -- -notes.txt
```

Actual output

Captured run · exit 0

```
-error
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-e and --`	Assign pattern and filename roles unambiguously.

Download: GRP-11-006.sh

12 - Compose small pipeline steps

Use a second filter for an AND condition. Capture status deliberately when pipefail is active.

GRP-12-001

Require two independent conditions

The second grep sees only lines selected by the first.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Code

```
grep '^ERROR' data/events.txt | grep 'disk'
```

Actual output

Captured run · exit 0

```
ERROR disk
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Two filters`	Compose an AND condition across sequential line filters.

Download: GRP-12-001.sh

GRP-12-002

Use two alternatives instead

Multiple -e patterns select lines satisfying either condition.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Compare with GRP-12-001 · Require two independent conditions.

Code

```
grep -e '^WARN' -e '^ERROR' data/events.txt
```

Actual output

Captured run · exit 0

```
WARN retry
ERROR disk
ERROR network
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Alternative patterns`	Select WARN OR ERROR records.

Download: GRP-12-002.sh

GRP-12-003

Remove one subset after selecting errors

The second inverted filter removes network errors from the selected records.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Compare with GRP-12-002 · Use two alternatives instead.

Code

```
grep '^ERROR' data/events.txt | grep -v 'network'
```

Actual output

Captured run · exit 0

```
ERROR disk
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-v in a pipeline`	Subtract one subset from an already selected set.

Download: GRP-12-003.sh

GRP-12-004

Capture an expected pipeline no-match

The complete driver enables pipefail. Catch the expected unsuccessful pipeline as a unit.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Compare with GRP-12-003 · Remove one subset after selecting errors.

Code

```
status=0
grep '^ERROR' data/events.txt | grep 'missing' || status=$?
printf 'pipeline exit=%s\n' "$status"
```

Actual output

Captured run · exit 0

```
pipeline exit=1
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`pipefail`	Preserve a nonzero pipeline outcome and handle it explicitly.

Download: GRP-12-004.sh

GRP-12-005

Branch on a quiet pipeline result

The condition handles a no-match outcome without treating it as an unexpected driver failure.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Compare with GRP-12-004 · Capture an expected pipeline no-match.

Code

```
if grep '^ERROR' data/events.txt | grep -q 'disk'; then
  printf 'disk error present\n'
fi
```

Actual output

Captured run · exit 0

```
disk error present
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`if pipeline`	Use the selected result as a shell condition with this tiny finite input.

Download: GRP-12-005.sh

GRP-12-006

Extract values after selecting the records

grep selects complete records; cut performs the simple delimiter-based extraction.

Input

data/settings.txt

```
# demo


PORT=8080
  # note
HOST=local
```

Code

```
grep '^PORT=' data/settings.txt | cut -d= -f2
```

Actual output

Captured run · exit 0

```
8080
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Selection then extraction`	Give each tool one clearly specified processing step.

Download: GRP-12-006.sh

13 - GNU grep extensions

These programs explicitly select GNU grep. NUL separators are converted to visible characters only to present the result; that display conversion is not a filename-processing recipe.

GRP-13-001

Label a standard-input stream

GNU --label gives the stdin stream a meaningful name when filename prefixes are forced.

Input

data/events.txt

```
INFO start
WARN retry
ERROR disk
INFO ready
ERROR network
```

Code

```
ggrep --label=stream -H '^ERROR' < data/events.txt
```

Actual output

Captured run · exit 0

```
stream:ERROR disk
stream:ERROR network
```

Parameter	Meaning in this example
`Implementation`	GNU grep
`--label`	Name standard input in prefixed GNU output.

Download: GRP-13-001.sh

GRP-13-002

Customize the context-group separator

Change only the separator between the two nonadjacent context groups.

Input

data/context.txt

```
one
two
HIT first
four
five
six
HIT last
eight
nine
```

Compare with GRP-07-006 · See separated context groups.

Code

```
ggrep -C 1 --group-separator=NEXT 'HIT' data/context.txt
```

Actual output

Captured run · exit 0

```
two
HIT first
four
NEXT
six
HIT last
eight
```

Parameter	Meaning in this example
`Implementation`	GNU grep
`--group-separator`	Choose a visible marker between separated groups.

Download: GRP-13-002.sh

GRP-13-003

Suppress context-group separators

Keep the same windows but remove the group marker.

Input

data/context.txt

```
one
two
HIT first
four
five
six
HIT last
eight
nine
```

Compare with GRP-13-002 · Customize the context-group separator.

Code

```
ggrep -C 1 --no-group-separator 'HIT' data/context.txt
```

Actual output

Captured run · exit 0

```
two
HIT first
four
six
HIT last
eight
```

Parameter	Meaning in this example
`Implementation`	GNU grep
`--no-group-separator`	Omit the marker between context groups.

Download: GRP-13-003.sh

GRP-13-004

Terminate listed filenames with NUL

The display-only tr converts NUL terminators to newlines. Real filename consumers should retain the NUL framing.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/b.log

```
WARN beta
ERROR network
```

Code

```
ggrep -lZ '^ERROR' data/a.log data/b.log | tr '\000' '\n'
```

Actual output

Captured run · exit 0

```
data/a.log
data/b.log
```

Parameter	Meaning in this example
`Implementation`	GNU grep
`-Z`	End filename labels with NUL rather than a line terminator.

Download: GRP-13-004.sh

GRP-13-005

Match a NUL-delimited record

GNU -z treats the first embedded newline as data within one record. The final pipe is the displayed NUL terminator.

Code

```
printf 'first\nalpha\000beta\000' | ggrep -z 'alpha' | tr '\000' '|'
printf '\n'
```

Actual output

Captured run · exit 0

```
first
alpha|
```

Parameter	Meaning in this example
`Implementation`	GNU grep
`-z`	Use NUL as the record separator; make the terminator visible only for display.

Download: GRP-13-005.sh

GRP-13-006

Select the PCRE matching mode explicitly

This optional GNU example requires grep built with PCRE support. The lookbehind is not POSIX ERE syntax.

Input

data/counts.txt

```
id=7 id=42
none
id=5
```

Code

```
ggrep -Po '(?<=id=)[0-9]+' data/counts.txt
```

Actual output

Captured run · exit 0

```
7
42
5
```

Parameter	Meaning in this example
`Implementation`	GNU grep
`-P`	Select GNU grep PCRE mode before using lookbehind.

Download: GRP-13-006.sh

14 - Build useful checks from small pieces

Apply the already introduced options to complete, small workflows with synthetic inputs.

GRP-14-001

Remove blank and whitespace-only lines

One anchored expression defines the complete set of blank records to omit.

Input

data/settings.txt

```
# demo


PORT=8080
  # note
HOST=local
```

Code

```
grep -v '^[[:space:]]*$' data/settings.txt
```

Actual output

Captured run · exit 0

```
# demo
PORT=8080
  # note
HOST=local
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Blank-line predicate`	Omit only records containing zero or more whitespace characters.

Download: GRP-14-001.sh

GRP-14-002

Also remove complete comment lines

Add the comment alternative to the blank-line rule. This invented format has no quoting or multiline values.

Input

data/settings.txt

```
# demo


PORT=8080
  # note
HOST=local
```

Compare with GRP-14-001 · Remove blank and whitespace-only lines.

Code

```
grep -Ev '^[[:space:]]*(#|$)' data/settings.txt
```

Actual output

Captured run · exit 0

```
PORT=8080
HOST=local
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Comment alternative`	Treat indentation followed by # as a comment line.

Download: GRP-14-002.sh

GRP-14-003

Keep only assignment-shaped lines

Use a positive complete record shape instead of merely discarding comments.

Input

data/settings.txt

```
# demo


PORT=8080
  # note
HOST=local
```

Compare with GRP-14-002 · Also remove complete comment lines.

Code

```
grep -E '^[A-Z_]+=[^=]+$' data/settings.txt
```

Actual output

Captured run · exit 0

```
PORT=8080
HOST=local
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Assignment shape`	Recognize this small format; it is not a general configuration parser.

Download: GRP-14-003.sh

GRP-14-004

Filter identifiers through an exact allowlist

Fixed-string whole-line matching avoids substring acceptance and regex interpretation of list entries.

Input

data/allow.txt

```
AB-123
CD-007
```

data/ids.txt

```
AB-7
AB-42
AB-123
CD-007
xAB-123
```

Code

```
grep -Fx -f data/allow.txt data/ids.txt
```

Actual output

Captured run · exit 0

```
AB-123
CD-007
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-Fx -f`	Treat each allowlist entry as a complete literal value.

Download: GRP-14-004.sh

GRP-14-005

Report acceptance for every candidate

Unlike grep selection, the shell loop emits a decision for both accepted and rejected inputs.

Input

data/ids.txt

```
AB-7
AB-42
AB-123
CD-007
xAB-123
```

Code

```
while IFS= read -r value; do
  if printf '%s\n' "$value" | grep -Eq '^[A-Z]{2}-[0-9]{3}$'; then
    printf 'accept %s\n' "$value"
  else
    printf 'reject %s\n' "$value"
  fi
done < data/ids.txt
```

Actual output

Captured run · exit 0

```
reject AB-7
reject AB-42
accept AB-123
accept CD-007
reject xAB-123
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`Complete-value check`	Keep rejected near-matches visible in a validation report.

Download: GRP-14-005.sh

GRP-14-006

Produce an error-location report

Combine already learned matching, filename and line-number controls without adding a new regex feature.

Input

data/a.log

```
INFO alpha
ERROR disk
```

data/b.log

```
WARN beta
ERROR network
```

Code

```
grep -nH '^ERROR' data/a.log data/b.log
```

Actual output

Captured run · exit 0

```
data/a.log:2:ERROR disk
data/b.log:2:ERROR network
```

Parameter	Meaning in this example
`Implementation`	BSD/GNU common
`-nH`	Retain both source filename and original line position in the report.

Download: GRP-14-006.sh

Example and command index

- GRP-01-001 · Select a line containing a word
- GRP-01-002 · Ignore case
- GRP-01-003 · Require a whole word
- GRP-01-004 · Require the whole line
- GRP-01-005 · Invert the selection
- GRP-01-006 · Show original line numbers
- GRP-02-001 · Match a literal substring
- GRP-02-002 · Anchor the start
- GRP-02-003 · Anchor the end
- GRP-02-004 · Anchor both ends
- GRP-02-005 · Allow one character at the dot
- GRP-02-006 · Match a period literally
- GRP-03-001 · Turn on extended syntax
- GRP-03-002 · Group alternatives before anchoring
- GRP-03-003 · Make a suffix optional
- GRP-03-004 · Require one or more digits
- GRP-03-005 · Require exactly three digits
- GRP-03-006 · Write the same interval as BRE
- GRP-04-001 · Search a metacharacter literally
- GRP-04-002 · Add another literal term
- GRP-04-003 · Move the terms into a pattern file
- GRP-04-004 · Observe an empty pattern
- GRP-04-005 · Distinguish an empty pattern file
- GRP-04-006 · Protect a pattern beginning with a dash
- GRP-05-001 · Show line numbers on log records
- GRP-05-002 · Force a filename prefix
- GRP-05-003 · Suppress prefixes across files
- GRP-05-004 · List only files containing a match
- GRP-05-005 · List files with no match
- GRP-05-006 · Print the matching portions
- GRP-06-001 · Count matching lines
- GRP-06-002 · Stop after the first selected line
- GRP-06-003 · Allow two selected lines
- GRP-06-004 · Ask a quiet yes-or-no question
- GRP-06-005 · Capture a no-match status
- GRP-06-006 · Capture a file-error status
- GRP-07-001 · Show one line after a match
- GRP-07-002 · Show one line before a match
- GRP-07-003 · Show context on both sides
- GRP-07-004 · Distinguish numbered context from matches
- GRP-07-005 · See overlapping windows merge
- GRP-07-006 · See separated context groups
- GRP-08-001 · Search two explicitly ordered files
- GRP-08-002 · Count each input file separately
- GRP-08-003 · Read the same log through redirection
- GRP-08-004 · Read standard input from a pipeline
- GRP-08-005 · Name standard input with a dash
- GRP-08-006 · Reverse the explicit input order
- GRP-09-001 · Recurse into a tiny directory tree
- GRP-09-002 · List recursive matching filenames
- GRP-09-003 · Include only text filenames
- GRP-09-004 · Exclude temporary filenames
- GRP-09-005 · Skip the cache directory
- GRP-09-006 · Combine two independent exclusions

- GRP-10-001 · Compare word and substring selection
- GRP-10-002 · Add case folding to the word test
- GRP-10-003 · Use literal whole-line matching
- GRP-10-004 · Select complete ASCII alphabetic lines
- GRP-10-005 · Exclude uppercase ASCII spellings
- GRP-10-006 · Remember that a word is not a field parser
- GRP-11-001 · Preserve a literal space in the pattern
- GRP-11-002 · Preserve a literal dollar sign
- GRP-11-003 · Search a literal term stored in a variable
- GRP-11-004 · Read a filename containing spaces
- GRP-11-005 · End option parsing before a filename
- GRP-11-006 · Protect both a pattern and a filename
- GRP-12-001 · Require two independent conditions
- GRP-12-002 · Use two alternatives instead
- GRP-12-003 · Remove one subset after selecting errors
- GRP-12-004 · Capture an expected pipeline no-match
- GRP-12-005 · Branch on a quiet pipeline result
- GRP-12-006 · Extract values after selecting the records
- GRP-13-001 · Label a standard-input stream
- GRP-13-002 · Customize the context-group separator
- GRP-13-003 · Suppress context-group separators
- GRP-13-004 · Terminate listed filenames with NUL
- GRP-13-005 · Match a NUL-delimited record
- GRP-13-006 · Select the PCRE matching mode explicitly
- GRP-14-001 · Remove blank and whitespace-only lines
- GRP-14-002 · Also remove complete comment lines
- GRP-14-003 · Keep only assignment-shaped lines
- GRP-14-004 · Filter identifiers through an exact allowlist
- GRP-14-005 · Report acceptance for every candidate
- GRP-14-006 · Produce an error-location report

Content: CC0 1.0. Structure and publishing framework: CC BY 4.0 by Lukas C. Bossert. Repository source · Publishing framework source · License scope and third-party notices.