Regular expressions, one change at a time

Choose a regex issue, then compare its actual syntax in plain BRE, grep -E, Perl, sed and Emacs. The input stays fixed; the tool dialect is explicit. Groups and replacement references are the centre of the tour.

Each issue ID contains all five dialect views on one shared input. The matched text and replacement result appear once. A changed assumption or additional check stays inside the same issue; commands and verification code are available on demand. All six native programs per issue are checked against authored expectations.

> **Last changes: 2026-11-01.** Added log extraction, DOI fields, filename transformations, text cleanup, date conversion and URL migrations, including counterexamples and calendar checks. One ID contains every dialect variation.

24
regex issues

144
native runs

1
canonical Org source

READING THE TOUR

One issue, five views

Start with the issue

Read the shared input and the comparison table. Identify the group, quantifier or reference before reading the command wrapper.

Compare exact spellings

Plain means POSIX BRE as used by grep without -E. It does not mean a universal regex language. Explicit flags and notes explain the differences.

Replace and verify

grep selects complete matches. Perl, sed and Emacs transform the same input; compare their results and then change one assumption.

Every topic has the same five views. The sed column uses default BRE where it can express the example; it explicitly selects `sed -E` where ERE alternation is needed. A multiline BRE cell lists separate `grep -e` patterns. No fixed-string example is presented as a regex function.

Capture and replacement syntax

Feature	Plain / BRE	grep -E	Perl	sed	Emacs
Capture	`\(...\)`	`(...)`	`(...)`	`\(...\)`; `(...)` with -E	`\(...\)`
One or more	`\{1,\}`	`+`	`+`	`\{1,\}`; `+` with -E	`+`
Optional	`\{0,1\}`	`?`	`?`	`\{0,1\}`; `?` with -E	`?`
Bounded repeat	`\{2,3\}`	`{2,3}`	`{2,3}`	`\{2,3\}`; `{2,3}` with -E	`\{2,3\}`
Replace with group 1	Search only	Search only	`$1`	`\1`	`\1`
Replace with whole match	Search only	Search only	`$&`	`&`	`\&`

There is no universal plain-regex syntax. POSIX BRE has no alternation operator; ERE has no noncapturing groups or named groups. Pattern backreferences in `grep -E` are a tested GNU/BSD extension, not POSIX ERE. Each issue shows supported alternatives without claiming identical syntax.

A reference inside a search pattern constrains what can match. A reference inside replacement text inserts captured text. Number groups by opening parentheses, including nested groups.

Search and replace directly inside Emacs

Open the supplied text in a buffer. With no active region, move to the buffer start using `M-<`. The Emacs examples show two separate, directly copyable fields.

- Run `M-x replace-regexp`, enter Search regexp and press RET, then enter Replacement and press RET. Every match from point onward is replaced.
- To inspect matches first, use the same Search regexp with `M-x re-search-forward`.
- For confirmation at each match, use `C-M-%` (`M-x query-replace-regexp`). Press `!` to replace all remaining matches.

For the case-sensitive results here, use `M-x set-variable` to set `case-fold-search` and `case-replace` to `nil`. Leave `replace-regexp-lax-whitespace` at its default `nil`: one space means one space. Type backslashes literally, without Lisp string quotes.

For Doe, Jane, Search regexp `^\([A-Za-z]+\), \([A-Za-z]+\)$` defines surname as group 1 and given name as group 2. Replacement `\2 \1` produces Jane Doe. To insert literal backslash and 1, type `\\1` instead.

The verified result is the resulting text from an automated native reference run, not an Emacs minibuffer transcript. The exact displayed fields are passed unchanged to Emacs. Its Lisp implementation is confined to the collapsible verification program and downloadable script.

Scope and primary references

Inputs are original ASCII teaching text with LF line endings and LC_ALL=C. grep -o is a GNU/BSD extension. sed -E is stated wherever it is used. Same replacement text is not assumed across tools: Perl uses $1, sed and Emacs use backslash references, and whole-match references differ again. Named groups and minimal quantifiers are not invented for engines that lack them. Date-shaped strings are not calendar validation; flat tags are not an HTML parser. The native examples run offline and need no account or personal editor setup.

- GNU grep: basic versus extended regular expressions
- GNU grep: backreferences and subexpressions
- Perl regular-expression tutorial
- Perl regular-expression reference
- GNU sed: BRE versus ERE
- GNU sed: substitution and replacement references
- Emacs: grouping and backreferences
- Emacs: regexp replacement
- grep tour: command options and file searching
- sed tour: substitution and text-processing workflow
- HTML Standard: email input and the scope of its address grammar
- RFC 3986: URI components and dotted-decimal IPv4 syntax
- Crossref: DOI candidates, suffix policy and punctuation bycatch
- RFC 3339: Gregorian leap-year rules
- ORCID: identifier structure and MOD 11-2 check digit

The examples, explanations, data and independently authored expected results are original teaching material. The manuals establish tool semantics.

RGX-01-001

Literal characters and metacharacters

A dot is a wildcard until a character class makes it literal. All five views can use [.] for a period. The search is unanchored, so the prefix of report.csv.bak also matches; replacement leaves its suffix intact.

Shared input · data/periods.txt

```
report.csv
reportXcsv
report.csv.bak
README
REPORT.csv
```

All variations for RGX-01-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`report[.]csv`	Search only	[.] makes the period literal in BRE.
grep -E / ERE	`report[.]csv`	Search only	The literal-period spelling is unchanged in ERE.
Perl	`report[.]csv`	`report.tsv`	[.] is literal in the search; report.tsv is replacement text.
sed / BRE	`report[.]csv`	`report.tsv`	The same [.] search works in default sed BRE.
Emacs	`report[.]csv`	`report.tsv`	Enter [.] literally; do not add string quotes.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
report.csv
report.csv
```

After replacement · Perl, sed and Emacs

```
report.tsv
reportXcsv
report.tsv.bak
README
REPORT.csv
```

Change one assumption: Let a wildcard match a period · see the comparison-check appendix.

Run all variants for this issue

RGX-02-001

Anchor the complete record

Anchors require a complete record. BRE and Emacs escape the repetition braces; ERE and Perl do not. The accepted identifiers are the same, despite those different spellings.

Shared input · data/anchors.txt

```
AB-123
xxAB-123yy
AB-12
xy-123
ZZ-999
AB-123tail
```

All variations for RGX-02-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^[A-Z]\{2\}-[0-9]\{3\}$`	Search only	Escape the count braces: \{2\} and \{3\}.
grep -E / ERE	`^[A-Z]{2}-[0-9]{3}$`	Search only	Use ordinary count braces: {2} and {3}.
Perl	`^[A-Z]{2}-[0-9]{3}$`	`[accepted]`	Anchors and ordinary count braces match the whole record.
sed / BRE	`^[A-Z]\{2\}-[0-9]\{3\}$`	`[accepted]`	Default sed uses escaped BRE count braces.
Emacs	`^[A-Z]\{2\}-[0-9]\{3\}$`	`[accepted]`	Emacs also needs \{2\} and \{3\}.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
AB-123
ZZ-999
```

After replacement · Perl, sed and Emacs

```
[accepted]
xxAB-123yy
AB-12
xy-123
[accepted]
AB-123tail
```

Change one assumption: Search without whole-record anchors · see the comparison-check appendix.

Run all variants for this issue

RGX-03-001

Choose a character class

A character class chooses one character; a quantifier repeats it. Use ASCII [0-9] for this agreed input. Do not copy Perl backslash-d into another engine and assume it means digits there.

Shared input · data/classes.txt

```
t=15
t=7
t=x
t=1.5
t=22
```

All variations for RGX-03-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^t=[0-9]\{1,\}$`	Search only	BRE repeats digits with \{1,\}.
grep -E / ERE	`^t=[0-9]+$`	Search only	ERE uses + for one or more digits.
Perl	`^t=[0-9]+$`	`t=#`	Use ASCII [0-9]; Perl \d is not universal syntax.
sed / BRE	`^t=[0-9]\{1,\}$`	`t=#`	Default sed uses \{1,\}, not ERE +.
Emacs	`^t=[0-9]+$`	`t=#`	Emacs supports + directly.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
t=15
t=7
t=22
```

After replacement · Perl, sed and Emacs

```
t=#
t=#
t=x
t=1.5
t=#
```

Change one assumption: Try the POSIX digit class · see the comparison-check appendix.

Run all variants for this issue

RGX-04-001

Make parts optional or bounded

The optional u accepts color and colour. BRE spells optional repetition as backslash-brace 0,1; ERE, Perl and Emacs use ?. Bounded counts still need escaped braces in BRE and Emacs.

Shared input · data/quantifiers.txt

```
color=12
colour=123
colouur=12
colour=1
color=1234
colour=99
```

All variations for RGX-04-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^colou\{0,1\}r=[0-9]\{2,3\}$`	Search only	Optional u is u\{0,1\}; digit counts use \{2,3\}.
grep -E / ERE	`^colou?r=[0-9]{2,3}$`	Search only	Optional u is u?; digit counts use {2,3}.
Perl	`^colou?r=[0-9]{2,3}$`	`color=#`	Perl uses the same ? and {2,3} spelling as ERE.
sed / BRE	`^colou\{0,1\}r=[0-9]\{2,3\}$`	`color=#`	BRE counts stay escaped in default sed.
Emacs	`^colou?r=[0-9]\{2,3\}$`	`color=#`	Emacs uses u? but escapes the count braces.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
color=12
colour=123
colour=99
```

After replacement · Perl, sed and Emacs

```
color=#
color=#
colouur=12
colour=1
color=1234
color=#
```

Change one assumption: Narrow the repetition bound · see the comparison-check appendix.

Run all variants for this issue

RGX-05-001

Group alternatives before anchoring

Group the cat/dog choice before anchoring the whole record. POSIX BRE has no alternation operator, so its view uses two separate patterns. sed explicitly enables ERE with -E. Emacs uses backslashed parentheses and alternation.

Shared input · data/alternation.txt

```
cat
dog
cats
dogs
bird
catdog
catX
```

All variations for RGX-05-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^cats\{0,1\}$` `^dogs\{0,1\}$`	Search only	BRE has no alternation operator; pass two separate -e patterns.
grep -E / ERE	`^(cat|dog)s?$`	Search only	Ordinary parentheses group | before the anchors.
Perl	`^(cat|dog)s?$`	`kind=$1`	The ordinary capture group supplies $1.
sed -E / ERE	`^(cat|dog)s?$`	`kind=\1`	-E enables ERE alternation; replacement uses \1.
Emacs	`^\(cat|dog\)s?$`	`kind=\1`	Use | for alternation and \(... \) for the capture.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
cat
dog
cats
dogs
```

After replacement · Perl, sed and Emacs

```
kind=cat
kind=dog
kind=cat
kind=dog
bird
catdog
catX
```

Change one assumption: Remove the alternation group · see the comparison-check appendix.

Run all variants for this issue

RGX-06-001

Define groups and swap their values

Define surname as group 1 and given name as group 2. BRE, default sed and Emacs use backslashed parentheses; ERE and Perl use ordinary parentheses. Perl replaces with $2 $1; sed and Emacs use backslash-2, space, backslash-1. Doe, Jane becomes Jane Doe.

Shared input · data/names.txt

```
Doe, Jane
Smith, Sam
No comma
Li, Ada
```

All variations for RGX-06-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([A-Za-z]\{1,\}\), \([A-Za-z]\{1,\}\)$`	Search only	Use \(... \) and \{1,\}; grep prints the whole match.
grep -E / ERE	`^([A-Za-z]+), ([A-Za-z]+)$`	Search only	Use (...) and +; grep still cannot replace captures.
Perl	`^([A-Za-z]+), ([A-Za-z]+)$`	`$2 $1`	Replacement $2 $1 swaps the two captures.
sed / BRE	`^\([A-Za-z]\{1,\}\), \([A-Za-z]\{1,\}\)$`	`\2 \1`	Default sed uses BRE groups; replacement is \2 \1.
Emacs	`^\([A-Za-z]+\), \([A-Za-z]+\)$`	`\2 \1`	Emacs groups are \(... \); replacement is \2 \1.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
Doe, Jane
Smith, Sam
Li, Ada
```

After replacement · Perl, sed and Emacs

```
Jane Doe
Sam Smith
No comma
Ada Li
```

Change one assumption: Reuse a capture twice · see the comparison-check appendix.

Run all variants for this issue

RGX-07-001

Number nested capture groups

Nested opening parentheses determine numbering. The outer year-month group is 1; year is 2, month is 3 and day is 4. Keep the same intended groups while changing syntax between dialects. These date-shaped strings are not calendar validation.

Shared input · data/nested.txt

```
2026-11-01
2025-02-03
2026/11/01
date=2026-11-01
```

All variations for RGX-07-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\(\([0-9]\{4\}\)-\([0-9]\{2\}\)\)-\([0-9]\{2\}\)$`	Search only	The outer year-month capture is group 1.
grep -E / ERE	`^(([0-9]{4})-([0-9]{2}))-([0-9]{2})$`	Search only	Ordinary parentheses keep the same nested numbering.
Perl	`^(([0-9]{4})-([0-9]{2}))-([0-9]{2})$`	`$4/$3/$2`	Reorder day/month/year with $4/$3/$2.
sed / BRE	`^\(\([0-9]\{4\}\)-\([0-9]\{2\}\)\)-\([0-9]\{2\}\)$`	`\4/\3/\2`	BRE groups keep numbering; replace with \4/\3/\2.
Emacs	`^\(\([0-9]\{4\}\)-\([0-9]\{2\}\)\)-\([0-9]\{2\}\)$`	`\4/\3/\2`	Use \4/\3/\2; do not count non-group punctuation.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
2026-11-01
2025-02-03
```

After replacement · Perl, sed and Emacs

```
01/11/2026
03/02/2025
2026/11/01
date=2026-11-01
```

Change one assumption: Choose the wrong nested group numbers · see the comparison-check appendix.

Run all variants for this issue

RGX-08-001

Refer back to text already matched

A backreference inside the pattern requires the same word twice. It is distinct from inserting a group in replacement text. BRE, Perl, sed and Emacs support the example; grep -E uses a tested GNU/BSD extension outside POSIX ERE.

Shared input · data/duplicates.txt

```
red red
blue blue
red blue
red red red
red  red
```

All variations for RGX-08-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([a-z]\{1,\}\) \1$`	Search only	\1 in the pattern is a POSIX BRE backreference.
grep -E / ERE	`^([a-z]+) \1$`	Search only	Pattern backreferences are a GNU/BSD extension to ERE, not POSIX ERE.
Perl	`^([a-z]+) \1$`	`$1`	\1 in the pattern tests equality; $1 inserts the word.
sed / BRE	`^\([a-z]\{1,\}\) \1$`	`\1`	\1 tests equality in the pattern and inserts text in replacement.
Emacs	`^\([a-z]+\) \1$`	`\1`	The pattern and replacement both use \1, for different jobs.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
red red
blue blue
```

After replacement · Perl, sed and Emacs

```
red
blue
red blue
red red red
red  red
```

Change one assumption: Capture words without requiring equality · see the comparison-check appendix.

Run all variants for this issue

RGX-09-001

Handle an unmatched optional capture

The title group is optional. An absent title contributes an empty string during replacement. BRE uses separate Dr/Prof patterns because it has no alternation; the other views express the alternatives in one pattern. A malformed title is left unchanged by replacement.

Shared input · data/optional.txt

```
Ada:Dr
Bob
Clara:Prof
Dan:
Eve:Sir
```

All variations for RGX-09-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([A-Za-z]\{1,\}\)\(:\(Dr\)\)\{0,1\}$` `^\([A-Za-z]\{1,\}\)\(:\(Prof\)\)\{0,1\}$`	Search only	Use separate optional Dr/Prof patterns instead of alternation.
grep -E / ERE	`^([A-Za-z]+)(:(Dr|Prof))?$`	Search only	ERE uses ? on the title group and | for the choice.
Perl	`^([A-Za-z]+)(:(Dr|Prof))?$`	`$1 [$3]`	An absent group 3 contributes empty text to $1 [$3].
sed -E / ERE	`^([A-Za-z]+)(:(Dr|Prof))?$`	`\1 [\3]`	-E enables the alternatives; \3 is empty when absent.
Emacs	`^\([A-Za-z]+\)\(:\(Dr|Prof\)\)?$`	`\1 [\3]`	The optional title is group 3; insert it using \3.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
Ada:Dr
Bob
Clara:Prof
```

After replacement · Perl, sed and Emacs

```
Ada [Dr]
Bob []
Clara [Prof]
Dan:
Eve:Sir
```

Change one assumption: Make the optional title mandatory · see the comparison-check appendix.

Run all variants for this issue

RGX-10-001

Name captures in Perl and map Emacs numbers

Perl gives the prefix and number names. The equivalent BRE, ERE, sed and Emacs patterns use numbered groups. Named replacement references survive an extra outer capture; numeric positions can change.

Shared input · data/named.txt

```
AB-17
XY-003
bad-17
AB-x
```

All variations for RGX-10-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([A-Z]\{1,\}\)-\([0-9]\{1,\}\)$`	Search only	Numbered groups: prefix is 1 and number is 2.
grep -E / ERE	`^([A-Z]+)-([0-9]+)$`	Search only	Ordinary numbered groups; no named-group syntax.
Perl	`^(?<prefix>[A-Z]+)-(?<number>[0-9]+)$`	`$+{number}/$+{prefix}`	Use prefix/number names and $+{number}/$+{prefix}.
sed / BRE	`^\([A-Z]\{1,\}\)-\([0-9]\{1,\}\)$`	`\2/\1`	Use numbered BRE groups and \2/\1.
Emacs	`^\([A-Z]+\)-\([0-9]+\)$`	`\2/\1`	Use numbered Emacs groups and \2/\1.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
AB-17
XY-003
```

After replacement · Perl, sed and Emacs

```
17/AB
003/XY
bad-17
AB-x
```

Change one assumption: Add an outer named capture · see the comparison-check appendix.

Run all variants for this issue

RGX-11-001

Group structure without capturing it

A group can organize a choice without capturing it in Perl and Emacs. ERE has no noncapturing group: the color takes group 1 and the number moves to group 2. sed -E therefore needs backslash-2 in replacement. The BRE view splits the choice into separate patterns.

Shared input · data/noncapturing.txt

```
red:12
blue:7
green:9
red:x
blue:007
```

All variations for RGX-11-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^red:\([0-9]\{1,\}\)$` `^blue:\([0-9]\{1,\}\)$`	Search only	No noncapturing group: separate patterns each capture digits as group 1.
grep -E / ERE	`^(red|blue):([0-9]+)$`	Search only	The color captures group 1, so digits become group 2.
Perl	`^(?:red|blue):([0-9]+)$`	`n=$1`	(?:...) does not capture; digits stay group 1.
sed -E / ERE	`^(red|blue):([0-9]+)$`	`n=\2`	-E captures color as 1 and digits as 2; use n=\2.
Emacs	`^\(?:red|blue\):\([0-9]+\)$`	`n=\1`	\(?:...\) does not capture; use n=\1.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
red:12
blue:7
blue:007
```

After replacement · Perl, sed and Emacs

```
n=12
n=7
green:9
red:x
n=007
```

Change one assumption: Let a structural group shift the numbers · see the comparison-check appendix.

Run all variants for this issue

RGX-12-001

Distinguish the whole match from group 1

The whole match is order=7 while capture 1 is just 7. Perl inserts the whole match with $&, sed with &, and Emacs with backslash-ampersand. Literal backslashes in replacement text need their own escaping.

Shared input · data/whole.txt

```
order=7
order=42
other=3
```

All variations for RGX-12-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`order=\([0-9]\{1,\}\)`	Search only	grep selects the complete order=number match.
grep -E / ERE	`order=([0-9]+)`	Search only	The selected capture still cannot be emitted separately by -o.
Perl	`order=([0-9]+)`	`[$&]`	Insert the whole match using $&.
sed / BRE	`order=\([0-9]\{1,\}\)`	`[&]`	In sed replacement, unescaped & means the whole match.
Emacs	`order=\([0-9]+\)`	`[\&]`	In Emacs replacement, \& means the whole match.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
order=7
order=42
```

After replacement · Perl, sed and Emacs

```
[order=7]
[order=42]
other=3
```

Change one assumption: Emacs: insert literal \1 · see the comparison-check appendix.

Run all variants for this issue

RGX-13-001

Limit greedy and minimal matches

Greedy .* reaches the final closing tag on a line. Perl and Emacs can use minimal .*?; traditional BRE/ERE instead use a negated character class for this flat input. The changed Emacs example merges two adjacent tags into one replacement. These patterns are not an HTML parser.

Shared input · data/greedy.txt

```
<b>one</b><b>two</b>
<b>three</b>
plain
```

All variations for RGX-13-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`<b>[^<]*</b>`	Search only	Use a negated class for this flat input; BRE has no minimal quantifier.
grep -E / ERE	`<b>[^<]*</b>`	Search only	Do not treat *? as a minimal ERE quantifier.
Perl	`<b>.*?</b>`	`[item]`	Perl supports minimal .*? directly.
sed / BRE	`<b>[^<]*</b>`	`[item]`	Use the restricted negated-class equivalent in BRE.
Emacs	`<b>.*?</b>`	`[item]`	Emacs supports minimal .*? directly.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
<b>one</b>
<b>two</b>
<b>three</b>
```

After replacement · Perl, sed and Emacs

```
[item][item]
[item]
plain
```

Change one assumption: Emacs: compare greedy matching · see the comparison-check appendix.

Run all variants for this issue

RGX-14-001

Replace every occurrence with capture groups

There can be several matches on one line. Perl and sed need g for every occurrence; Emacs replace-regexp replaces from point onward, so start at the beginning. grep -o prints each complete match rather than editing the input.

Shared input · data/global.txt

```
a1 a2 a3
b4 a5
none
```

All variations for RGX-14-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`\(a\)\([0-9]\{1,\}\)`	Search only	BRE groups use escaped parentheses and digit counts.
grep -E / ERE	`(a)([0-9]+)`	Search only	ERE uses ordinary parentheses and +.
Perl	`(a)([0-9]+)`	`$2$1`	The g flag replaces every match on each line.
sed / BRE	`\(a\)\([0-9]\{1,\}\)`	`\2\1`	The g flag also replaces every match on each line.
Emacs	`\(a\)\([0-9]+\)`	`\2\1`	No g flag: replace-regexp works from point onward.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
a1
a2
a3
a5
```

After replacement · Perl, sed and Emacs

```
1a 2a 3a
b4 5a
none
```

Change one assumption: Replace only the first match per line · see the comparison-check appendix.

Run all variants for this issue

RGX-15-001

Screen an ASCII email address

Use a deliberately bounded ASCII email policy: nonempty local segments made from letters, digits, _, +, % and -, separated by single dots; DNS-style domain labels; a final alphabetic label of at least two characters. It rejects consecutive local dots and domain-edge hyphens. Quoted or internationalized mailboxes and other valid local symbols are outside this policy. A match does not prove mailbox existence or delivery.

Shared input · data/email.txt

```
ada@example.org
ada.lovelace+notes@research.example.org
team@lab-2.example.org
bad..dots@example.org
name@-example.org
name@example
missing-at.example.org
"quoted local"@example.org
```

All variations for RGX-15-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([A-Za-z0-9_+%-]\{1,\}\([.][A-Za-z0-9_+%-]\{1,\}\)*\)@\(\([A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)\{0,1\}[.]\)\{1,\}[A-Za-z]\{2,\}\)$`	Search only	BRE escapes the groups, + and ? repetitions, and count braces.
grep -E / ERE	`^([A-Za-z0-9_+%-]+([.][A-Za-z0-9_+%-]+)*)@(([A-Za-z0-9]([A-Za-z0-9-]*[A-Za-z0-9])?[.])+[A-Za-z]{2,})$`	Search only	Group 1 is the local part; group 3 is the full domain.
Perl	`^([A-Za-z0-9_+%-]+([.][A-Za-z0-9_+%-]+)*)@(([A-Za-z0-9]([A-Za-z0-9-]*[A-Za-z0-9])?[.])+[A-Za-z]{2,})$`	`local=$1` `domain=$3`	Extract $1 and $3; leave unmatched records unchanged.
sed / BRE	`^\([A-Za-z0-9_+%-]\{1,\}\([.][A-Za-z0-9_+%-]\{1,\}\)*\)@\(\([A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)\{0,1\}[.]\)\{1,\}[A-Za-z]\{2,\}\)$`	`local=\1` `domain=\3`	Default sed uses BRE groups and \1 / \3 in replacement.
Emacs	`^\([A-Za-z0-9_+%-]+\([.][A-Za-z0-9_+%-]+\)*\)@\(\([A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)?[.]\)+[A-Za-z]\{2,\}\)$`	`local=\1` `domain=\3`	Enter \1 and \3 in the Emacs replacement field.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Format matches · BRE and grep -E

```
ada@example.org
ada.lovelace+notes@research.example.org
team@lab-2.example.org
```

Extracted fields · Perl, sed and Emacs

```
local=ada domain=example.org
local=ada.lovelace+notes domain=research.example.org
local=team domain=lab-2.example.org
bad..dots@example.org
name@-example.org
name@example
missing-at.example.org
"quoted local"@example.org
```

Change one assumption: Try an over-permissive email shortcut · see the comparison-check appendix.

Run all variants for this issue

RGX-16-001

Screen HTTP(S) URLs and capture the host

Screen a lowercase http/https profile with a dotted ASCII host, an optional 1–5 digit port and a non-space path/query/fragment. Userinfo, IPv6 literals, local hostnames and internationalized hostnames are outside this profile. The five regex views check shape only: 99999 still fits the port spelling. The additional check enforces the port range; full URI parsing, percent-encoding and reachability are separate questions.

Shared input · data/urls.txt

```
https://example.org
http://data.example.org:8080/run?q=regex#part
https://example.org?q=one
https://example.org/a%20b
https://example.org:99999/path
ftp://example.org/file
https://bad..example.org/path
https://example.org/a b
```

All variations for RGX-16-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\(https\{0,1\}\)://\([A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)\{0,1\}\([.][A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)\{0,1\}\)\{1,\}\)\(:\([0-9]\{1,5\}\)\)\{0,1\}\([/?#][^[:space:]]*\)\{0,1\}$`	Search only	The same structure needs escaped BRE groups and bounded repeats.
grep -E / ERE	`^(https?)://([A-Za-z0-9]([A-Za-z0-9-]*[A-Za-z0-9])?([.][A-Za-z0-9]([A-Za-z0-9-]*[A-Za-z0-9])?)+)(:([0-9]{1,5}))?([/?#][^[:space:]]*)?$`	Search only	Scheme is group 1, full host group 2 and port digits group 7.
Perl	`^(https?)://([A-Za-z0-9]([A-Za-z0-9-]*[A-Za-z0-9])?([.][A-Za-z0-9]([A-Za-z0-9-]*[A-Za-z0-9])?)+)(:([0-9]{1,5}))?([/?#][^[:space:]]*)?$`	`scheme=$1 host=$2 port=$7`	The optional $7 inserts empty text when no port was supplied.
sed / BRE	`^\(https\{0,1\}\)://\([A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)\{0,1\}\([.][A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)\{0,1\}\)\{1,\}\)\(:\([0-9]\{1,5\}\)\)\{0,1\}\([/?#][^[:space:]]*\)\{0,1\}$`	`scheme=\1 host=\2 port=\7`	Default BRE keeps the same group numbers: \1, \2 and \7.
Emacs	`^\(https?\)://\([A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)?\([.][A-Za-z0-9]\([A-Za-z0-9-]*[A-Za-z0-9]\)?\)+\)\(:\([0-9]\{1,5\}\)\)?\([/?#][^[:space:]]*\)?$`	`scheme=\1 host=\2 port=\7`	Emacs supports ? and +, but escapes braces and groups.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Format matches · BRE and grep -E

```
https://example.org
http://data.example.org:8080/run?q=regex#part
https://example.org?q=one
https://example.org/a%20b
https://example.org:99999/path
```

Extracted fields · Perl, sed and Emacs

```
scheme=https host=example.org port=
scheme=http host=data.example.org port=8080
scheme=https host=example.org port=
scheme=https host=example.org port=
scheme=https host=example.org port=99999
ftp://example.org/file
https://bad..example.org/path
https://example.org/a b
```

Additional check: Check the numeric port range · see the comparison-check appendix.

Run all variants for this issue

RGX-17-001

Screen ORCID iDs and verify their check digit

Accept a bare, hyphenated ORCID-shaped identifier or its exact https://orcid.org/ form. Keep all 16 characters, including leading zeros; the final character can be an uppercase X. The replacement builds the canonical URI. This format screen also accepts an incorrect check digit. The additional ISO 7064 MOD 11-2 calculation distinguishes it; a correct checksum is still not proof that the registry issued the identifier.

Shared input · data/orcid.txt

```
0000-0002-1825-0097
0000-0002-1694-233X
https://orcid.org/0000-0002-1825-0097
0000-0002-1825-0098
0000-0002-1694-233x
0000 0002 1825 0097
0000-0002-1825-097
https://example.org/0000-0002-1825-0097
```

All variations for RGX-17-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\(https://orcid[.]org/\)\{0,1\}\([0-9]\{4\}\)-\([0-9]\{4\}\)-\([0-9]\{4\}\)-\([0-9]\{3\}[0-9X]\)$`	Search only	Groups and {4}/{3} counts need BRE backslashes.
grep -E / ERE	`^(https://orcid[.]org/)?([0-9]{4})-([0-9]{4})-([0-9]{4})-([0-9]{3}[0-9X])$`	Search only	The optional URI prefix is group 1; identifier chunks are 2–5.
Perl	`^(https://orcid[.]org/)?([0-9]{4})-([0-9]{4})-([0-9]{4})-([0-9]{3}[0-9X])$`	`https://orcid.org/$2-$3-$4-$5`	Build the URI with $2-$3-$4-$5; preserve leading zeros.
sed / BRE	`^\(https://orcid[.]org/\)\{0,1\}\([0-9]\{4\}\)-\([0-9]\{4\}\)-\([0-9]\{4\}\)-\([0-9]\{3\}[0-9X]\)$`	`https://orcid.org/\2-\3-\4-\5`	Build the same URI with \2-\3-\4-\5.
Emacs	`^\(https://orcid[.]org/\)?\([0-9]\{4\}\)-\([0-9]\{4\}\)-\([0-9]\{4\}\)-\([0-9]\{3\}[0-9X]\)$`	`https://orcid.org/\2-\3-\4-\5`	Use an uppercase X; the tour binds case-fold-search to nil.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Format matches · BRE and grep -E

```
0000-0002-1825-0097
0000-0002-1694-233X
https://orcid.org/0000-0002-1825-0097
0000-0002-1825-0098
```

Canonical form · Perl, sed and Emacs

```
https://orcid.org/0000-0002-1825-0097
https://orcid.org/0000-0002-1694-233X
https://orcid.org/0000-0002-1825-0097
https://orcid.org/0000-0002-1825-0098
0000-0002-1694-233x
0000 0002 1825 0097
0000-0002-1825-097
https://example.org/0000-0002-1825-0097
```

Additional check: Verify the ORCID checksum · see the comparison-check appendix.

Run all variants for this issue

RGX-18-001

Screen dotted IPv4 and check the octets

Use four captured runs of one to three ASCII digits, separated by literal periods. This is a shape screen, so 256 and a leading-zero octet still match. The additional check enforces decimal octets in 0–255 without leading zeros. address-syntax-ok says nothing about whether an address is public, assigned or reachable.

Shared input · data/ipv4.txt

```
192.0.2.1
198.51.100.42
0.0.0.0
255.255.255.255
256.1.2.3
192.168.001.1
1.2.3
192.0.2.1:80
```

All variations for RGX-18-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)$`	Search only	BRE escapes the group delimiters and \{1,3\} bounds.
grep -E / ERE	`^([0-9]{1,3})[.]([0-9]{1,3})[.]([0-9]{1,3})[.]([0-9]{1,3})$`	Search only	[.] is a literal separator; {1,3} only limits digit count.
Perl	`^([0-9]{1,3})[.]([0-9]{1,3})[.]([0-9]{1,3})[.]([0-9]{1,3})$`	`octets=$1/$2/$3/$4`	Groups $1–$4 hold the four octet strings.
sed / BRE	`^\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)$`	`octets=\1/\2/\3/\4`	Default sed uses \1–\4 in the replacement.
Emacs	`^\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)[.]\([0-9]\{1,3\}\)$`	`octets=\1/\2/\3/\4`	Emacs escapes the groups and count braces; use \1–\4.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Format matches · BRE and grep -E

```
192.0.2.1
198.51.100.42
0.0.0.0
255.255.255.255
256.1.2.3
192.168.001.1
```

Extracted fields · Perl, sed and Emacs

```
octets=192/0/2/1
octets=198/51/100/42
octets=0/0/0/0
octets=255/255/255/255
octets=256/1/2/3
octets=192/168/001/1
1.2.3
192.0.2.1:80
```

Additional check: Check decimal octets and leading zeros · see the comparison-check appendix.

Run all variants for this issue

RGX-19-001

Extract fields from structured logs

Match one complete record: a UTC-shaped timestamp, a status from 100 to 599, a token request ID and a non-space path. Groups 1, 2 and 3 capture time, status and ID; path text is deliberately not captured. The token - means no request ID. This reads a stated log format, not arbitrary server logs or calendar-valid timestamps.

Shared input · data/logs.txt

```
2026-11-01T09:15:00Z status=200 request=req-17 path=/guide
2026-11-01T09:15:01Z status=404 request=req-18 path=/missing
2026-11-01T09:15:02Z status=503 request=req-19 path=/api?q=1
2026-11-01T09:15:03Z status=200 request=- path=/health
2026-11-01T09:15:04Z status=20 request=req-20 path=/bad
unstructured log message
```

All variations for RGX-19-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([0-9]\{4\}-[0-9]\{2\}-[0-9]\{2\}T[0-9]\{2\}:[0-9]\{2\}:[0-9]\{2\}Z\) status=\([1-5][0-9]\{2\}\) request=\([A-Za-z0-9-]\{1,\}\) path=[^[:space:]]\{1,\}$`	Search only	Escape groups, + and repeat counts in BRE. Anchors cover the full log record.
grep -E / ERE	`^([0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}Z) status=([1-5][0-9]{2}) request=([A-Za-z0-9-]+) path=[^[:space:]]+$`	Search only	Groups 1/2/3 hold timestamp/status/request ID; the 20 status fails.
Perl	`^([0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}Z) status=([1-5][0-9]{2}) request=([A-Za-z0-9-]+) path=[^[:space:]]+$`	`time=$1 code=$2 id=$3`	Keep $1, $2 and $3 while discarding the path field.
sed / BRE	`^\([0-9]\{4\}-[0-9]\{2\}-[0-9]\{2\}T[0-9]\{2\}:[0-9]\{2\}:[0-9]\{2\}Z\) status=\([1-5][0-9]\{2\}\) request=\([A-Za-z0-9-]\{1,\}\) path=[^[:space:]]\{1,\}$`	`time=\1 code=\2 id=\3`	Use the same capture numbers with \1, \2 and \3.
Emacs	`^\([0-9]\{4\}-[0-9]\{2\}-[0-9]\{2\}T[0-9]\{2\}:[0-9]\{2\}:[0-9]\{2\}Z\) status=\([1-5][0-9]\{2\}\) request=\([A-Za-z0-9-]+\) path=[^[:space:]]+$`	`time=\1 code=\2 id=\3`	Enter literal regexp backslashes; use the shared case-sensitive setup.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
2026-11-01T09:15:00Z status=200 request=req-17 path=/guide
2026-11-01T09:15:01Z status=404 request=req-18 path=/missing
2026-11-01T09:15:02Z status=503 request=req-19 path=/api?q=1
2026-11-01T09:15:03Z status=200 request=- path=/health
```

After replacement · Perl, sed and Emacs

```
time=2026-11-01T09:15:00Z code=200 id=req-17
time=2026-11-01T09:15:01Z code=404 id=req-18
time=2026-11-01T09:15:02Z code=503 id=req-19
time=2026-11-01T09:15:03Z code=200 id=-
2026-11-01T09:15:04Z status=20 request=req-20 path=/bad
unstructured log message
```

Change one assumption: Select only client and server errors · see the comparison-check appendix.

Run all variants for this issue

RGX-20-001

Normalize DOI fields in literature data

Accept a bare DOI candidate or a lowercase doi: prefix with optional horizontal blanks. Group 1 is the display prefix; group 2 is the identifier. The bounded ASCII suffix policy includes parentheses and punctuation; it is not a universal DOI validator. These fictional examples require no lookup. Prefix normalization preserves identifier case. For surname/given-name conversion, compare RGX-06-001.

Shared input · data/dois.txt

```
doi:10.1234/ALPHA.7
10.56789/notes-2
doi: 10.2468/data_set(3)
doi:10.123/short
doi:10.1234/
See 10.1234/ALPHA.7.
https://doi.org/10.56789/notes-2
```

All variations for RGX-20-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\(doi:[[:blank:]]*\)\{0,1\}\(10[.][0-9]\{4,9\}/[-A-Za-z0-9._;()/:]\{1,\}\)$`	Search only	[.] means a literal dot; the registrant part has 4–9 ASCII digits.
grep -E / ERE	`^(doi:[[:blank:]]*)?(10[.][0-9]{4,9}/[-A-Za-z0-9._;()/:]+)$`	Search only	The DOI is group 2; parentheses inside its character class are literal.
Perl	`^(doi:[[:blank:]]*)?(10[.][0-9]{4,9}/[-A-Za-z0-9._;()/:]+)$`	`https://doi.org/$2`	Discard the display prefix, then prepend https://doi.org/ to $2.
sed / BRE	`^\(doi:[[:blank:]]*\)\{0,1\}\(10[.][0-9]\{4,9\}/[-A-Za-z0-9._;()/:]\{1,\}\)$`	`https://doi.org/\2`	Default BRE captures the same identifier; replacement uses \2.
Emacs	`^\(doi:[[:blank:]]*\)?\(10[.][0-9]\{4,9\}/[-A-Za-z0-9._;()/:]+\)$`	`https://doi.org/\2`	Use the exact Emacs field; preserve case with case-replace set to nil.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
doi:10.1234/ALPHA.7
10.56789/notes-2
doi: 10.2468/data_set(3)
```

DOI URI fields · Perl, sed and Emacs

```
https://doi.org/10.1234/ALPHA.7
https://doi.org/10.56789/notes-2
https://doi.org/10.2468/data_set(3)
doi:10.123/short
doi:10.1234/
See 10.1234/ALPHA.7.
https://doi.org/10.56789/notes-2
```

Change one assumption: Try unanchored replacement inside prose · see the comparison-check appendix.

Run all variants for this issue

RGX-21-001

Reassemble project filenames with captures

The contract is project_YYYYMMDD_vNN.ext: a project token may contain letters, digits and hyphens, the version has exactly two digits and the extension has three lowercase letters. Groups 1–6 are project/year/month/day/version/extension. Put the date first without dropping zero padding. The input is a text list; no files are renamed. Date spelling alone does not validate a calendar day; see RGX-23-001.

Shared input · data/filenames.txt

```
atlas_20261101_v02.csv
atlas_20261031_v12.tsv
rna-seq_20260229_v01.csv
atlas_20261101_v2.csv
my_project_20261101_v02.csv
notes.txt
```

All variations for RGX-21-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([A-Za-z][A-Za-z0-9-]*\)_\([0-9]\{4\}\)\([0-9]\{2\}\)\([0-9]\{2\}\)_v\([0-9]\{2\}\)[.]\([a-z]\{3\}\)$`	Search only	[.] separates the extension literally; fixed repeats enforce padding.
grep -E / ERE	`^([A-Za-z][A-Za-z0-9-]*)_([0-9]{4})([0-9]{2})([0-9]{2})_v([0-9]{2})[.]([a-z]{3})$`	Search only	Six captures define the handover; group 5 keeps the original version text.
Perl	`^([A-Za-z][A-Za-z0-9-]*)_([0-9]{4})([0-9]{2})([0-9]{2})_v([0-9]{2})[.]([a-z]{3})$`	`${2}-${3}-${4}_${1}_v${5}.${6}`	Braced numbered references make adjacency to underscores explicit.
sed / BRE	`^\([A-Za-z][A-Za-z0-9-]*\)_\([0-9]\{4\}\)\([0-9]\{2\}\)\([0-9]\{2\}\)_v\([0-9]\{2\}\)[.]\([a-z]\{3\}\)$`	`\2-\3-\4_\1_v\5.\6`	The same six captures can be reordered with \2, \3, \4, \1, \5, \6.
Emacs	`^\([A-Za-z][A-Za-z0-9-]*\)_\([0-9]\{4\}\)\([0-9]\{2\}\)\([0-9]\{2\}\)_v\([0-9]\{2\}\)[.]\([a-z]\{3\}\)$`	`\2-\3-\4_\1_v\5.\6`	Use the displayed replacement directly; it is not a Lisp string.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
atlas_20261101_v02.csv
atlas_20261031_v12.tsv
rna-seq_20260229_v01.csv
```

After replacement · Perl, sed and Emacs

```
2026-11-01_atlas_v02.csv
2026-10-31_atlas_v12.tsv
2026-02-29_rna-seq_v01.csv
atlas_20261101_v2.csv
my_project_20261101_v02.csv
notes.txt
```

Change one assumption: Allow versions with one to three digits · see the comparison-check appendix.

Run all variants for this issue

RGX-22-001

Normalize horizontal whitespace and repeated words

Group 1 captures an ASCII word; [[:blank:]]+ consumes its following horizontal whitespace. Replace the whole match by that word and one space. The Use line contains two real tabs. Line breaks and case stay unchanged. Matched-text results retain trailing whitespace, which can be hard to see; the replacement result makes the cleanup visible. The exercise then adds a distinct repeated-word pass.

Shared input · data/cleanup.txt

```
Keep   spacing   readable.
Use		one separator.
the the result
go go go
The the result
line break stays
```

All variations for RGX-22-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`\([A-Za-z]\{1,\}\)[[:blank:]]\{1,\}`	Search only	A word capture followed by one or more blanks; neither class matches LF.
grep -E / ERE	`([A-Za-z]+)[[:blank:]]+`	Search only	Use [[:blank:]], not a portable-looking but ambiguous backslash shortcut.
Perl	`([A-Za-z]+)[[:blank:]]+`	`$1`	Replace with $1 and one trailing space; leave punctuation and case intact.
sed / BRE	`\([A-Za-z]\{1,\}\)[[:blank:]]\{1,\}`	`\1`	Keep \1 plus one space. Do not trim the replacement field.
Emacs	`\([A-Za-z]+\)[[:blank:]]+`	`\1`	Copy the trailing space too. Leave lax-whitespace nil for exact fields.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
Keep
spacing
Use
one
the
the
go
go
The
the
line
break
```

After replacement · Perl, sed and Emacs

```
Keep spacing readable.
Use one separator.
the the result
go go go
The the result
line break stays
```

Change one assumption: Add a second pass for repeated whole words · see the comparison-check appendix.

Run all variants for this issue

RGX-23-001

Reorder date fields and check calendar rules

Capture exactly DD.MM.YYYY as day/month/year groups 1/2/3 and replace it by YYYY-MM-DD. The regex checks digit counts and literal separators, so impossible dates also transform. The additional offline check applies Gregorian month lengths and leap-year rules to years 0001–9999: 1900 is not a leap year, while 2000 is. Input already in ISO order stays unchanged.

Shared input · data/dates.txt

```
01.11.2026
29.02.2024
29.02.2026
31.04.2026
29.02.1900
29.02.2000
00.13.2026
2026-11-01
```

All variations for RGX-23-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^\([0-9]\{2\}\)[.]\([0-9]\{2\}\)[.]\([0-9]\{4\}\)$`	Search only	Escaped count braces and groups; [.] keeps the separators literal.
grep -E / ERE	`^([0-9]{2})[.]([0-9]{2})[.]([0-9]{4})$`	Search only	Anchors reject the existing ISO form. Digit counts do not validate dates.
Perl	`^([0-9]{2})[.]([0-9]{2})[.]([0-9]{4})$`	`$3-$2-$1`	Use $3-$2-$1; capture values preserve the leading zeros.
sed / BRE	`^\([0-9]\{2\}\)[.]\([0-9]\{2\}\)[.]\([0-9]\{4\}\)$`	`\3-\2-\1`	Use \3-\2-\1 in replacement; no date utility is invoked.
Emacs	`^\([0-9]\{2\}\)[.]\([0-9]\{2\}\)[.]\([0-9]\{4\}\)$`	`\3-\2-\1`	The actual minibuffer fields use literal backslashes, without Lisp quotes.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
01.11.2026
29.02.2024
29.02.2026
31.04.2026
29.02.1900
29.02.2000
00.13.2026
```

After replacement · Perl, sed and Emacs

```
2026-11-01
2024-02-29
2026-02-29
2026-04-31
1900-02-29
2000-02-29
2026-13-00
2026-11-01
```

Additional check: Check month lengths and century leap years · see the comparison-check appendix.

Run all variants for this issue

RGX-24-001

Migrate URL paths while preserving query and fragment

Match only the exact old.example.org host and the /docs/slug/YYYY path. Groups 1/2/3 hold slug/year/optional query or fragment. Build the new host and /guides/YYYY/slug, then append group 3 unchanged. The host dots are literal and the path is anchored before its optional tail. Similar hostnames, extra path segments and other sites stay unchanged. These are text rewrites, not HTTP requests or server redirects.

Shared input · data/migration.txt

```
https://old.example.org/docs/regex/2026
https://old.example.org/docs/data-tools/2025?view=all#top
https://old.example.org/docs/regex/2026?next=/docs/awk/2024
https://old.example.org.evil.test/docs/regex/2026
https://old.example.org/docs/regex/2026/extra
https://elsewhere.example.org/docs/regex/2026
```

All variations for RGX-24-001

Variant	Search regexp	Replacement	Difference to notice
Plain / BRE (grep)	`^https://old[.]example[.]org/docs/\([A-Za-z0-9-]\{1,\}\)/\([0-9]\{4\}\)\([?#][^[:space:]]*\)\{0,1\}$`	Search only	BRE groups and optional tail need escapes; host dots stay literal.
grep -E / ERE	`^https://old[.]example[.]org/docs/([A-Za-z0-9-]+)/([0-9]{4})([?#][^[:space:]]*)?$`	Search only	Anchors constrain host and path; the optional query/fragment is group 3.
Perl	`^https://old[.]example[.]org/docs/([A-Za-z0-9-]+)/([0-9]{4})([?#][^[:space:]]*)?$`	`https://new.example.org/guides/$2/$1$3`	Swap $2/$1, preserve $3 and change only the declared host.
sed / BRE	`^https://old[.]example[.]org/docs/\([A-Za-z0-9-]\{1,\}\)/\([0-9]\{4\}\)\([?#][^[:space:]]*\)\{0,1\}$`	`https://new.example.org/guides/\2/\1\3`	Use \2/\1\3; an unmatched tail inserts empty text.
Emacs	`^https://old[.]example[.]org/docs/\([A-Za-z0-9-]+\)/\([0-9]\{4\}\)\([?#][^[:space:]]*\)?$`	`https://new.example.org/guides/\2/\1\3`	The same tail capture preserves query/fragment text, including /docs/ inside it.

Emacs: enter the two fields with `M-x replace-regexp`. Multiline BRE entries are separate `grep -e` patterns. Commands are available below.

Matched text · BRE and grep -E

```
https://old.example.org/docs/regex/2026
https://old.example.org/docs/data-tools/2025?view=all#top
https://old.example.org/docs/regex/2026?next=/docs/awk/2024
```

After replacement · Perl, sed and Emacs

```
https://new.example.org/guides/2026/regex
https://new.example.org/guides/2025/data-tools?view=all#top
https://new.example.org/guides/2026/regex?next=/docs/awk/2024
https://old.example.org.evil.test/docs/regex/2026
https://old.example.org/docs/regex/2026/extra
https://elsewhere.example.org/docs/regex/2026
```

Change one assumption: Try a global path-only rewrite · see the comparison-check appendix.

Run all variants for this issue

Comparison checks · assumptions and validation

Each check belongs to the issue ID shown. The HTML keeps these results inside their issue; the print edition collects them here.

RGX-01-001 · Let a wildcard match a period

Remove the literal-dot character class. The wildcard also changes reportXcsv, which the literal pattern left alone.

Changed regexp

```
report.csv
```

Replacement

```
report.tsv
```

Changed result

```
report.tsv
report.tsv
report.tsv.bak
README
REPORT.csv
```

RGX-02-001 · Search without whole-record anchors

Remove the anchors. The expression now extracts identifiers embedded in longer records.

Changed regexp

```
[A-Z]{2}-[0-9]{3}
```

Changed result

```
AB-123
AB-123
ZZ-999
AB-123
```

RGX-03-001 · Try the POSIX digit class

Use the POSIX digit class instead of an ASCII range. On these ASCII inputs the result is identical; this does not make Perl-style backslash-d portable.

Changed regexp

```
^t=[[:digit:]]+$
```

Changed result

```
t=15
t=7
t=22
```

RGX-04-001 · Narrow the repetition bound

Require exactly two digits instead of two or three. Records with three digits are excluded.

Changed regexp

```
^colou?r=[0-9]{2}$
```

Changed result

```
color=12
colour=99
```

RGX-05-001 · Remove the alternation group

Remove the alternation group. The beginning anchor applies only to cat, and the ending anchor only to dogs?.

Changed regexp

```
^cat|dogs?$
```

Changed result

```
cat
dog
cat
dogs
cat
dog
cat
```

RGX-06-001 · Reuse a capture twice

Reuse a captured value twice in replacement text: the surname appears after the given name and again in parentheses.

Changed regexp

```
^([A-Za-z]+), ([A-Za-z]+)$
```

Replacement

```
$2 $1 ($1)
```

Changed result

```
Jane Doe (Doe)
Sam Smith (Smith)
No comma
Ada Li (Li)
```

RGX-07-001 · Choose the wrong nested group numbers

Use the wrong nested group numbers. Read the incorrect date output to identify which opening parenthesis each number names.

Changed regexp

```
^(([0-9]{4})-([0-9]{2}))-([0-9]{2})$
```

Replacement

```
$3/$2/$1
```

Changed result

```
11/2026/2026-11
02/2025/2025-02
2026/11/01
date=2026-11-01
```

RGX-08-001 · Capture words without requiring equality

Capture two independent words instead of using a pattern backreference. Different words now match too.

Changed regexp

```
^([a-z]+) ([a-z]+)$
```

Changed result

```
red|red
blue|blue
red|blue
```

RGX-09-001 · Make the optional title mandatory

Make the title mandatory. The record without a title no longer matches.

Changed regexp

```
^([A-Za-z]+)(:(Dr|Prof))$
```

Changed result

```
Ada|Dr
Clara|Prof
```

RGX-10-001 · Add an outer named capture

Add an outer named group. Named replacement references still identify prefix and number even though numeric positions change.

Changed regexp

```
^(?<whole>(?<prefix>[A-Z]+)-(?<number>[0-9]+))$
```

Replacement

```
$+{number}/$+{prefix}
```

Changed result

```
17/AB
003/XY
bad-17
AB-x
```

RGX-11-001 · Let a structural group shift the numbers

Make the structural color group capturing but keep the old replacement number. The output contains the color instead of the number.

Changed regexp

```
^(red|blue):([0-9]+)$
```

Replacement

```
n=$1
```

Changed result

```
n=red
n=blue
green:9
red:x
n=blue
```

RGX-12-001 · Emacs: insert literal \1

Insert the literal characters backslash and 1. In the Emacs Replacement field, type \\1 rather than \1.

Changed regexp

```
order=\([0-9]+\)
```

Replacement

```
\\1
```

Changed result

```
\1
\1
other=3
```

RGX-13-001 · Emacs: compare greedy matching

Use greedy .* in Emacs instead of minimal .*?. The two adjacent tags on the first line become one match and one replacement.

Changed regexp

```
<b>.*</b>
```

Replacement

```
[item]
```

Changed result

```
[item]
[item]
plain
```

RGX-14-001 · Replace only the first match per line

Remove the Perl g flag. Only the first match on each input line is replaced.

Changed regexp

```
(a)([0-9]+)
```

Replacement

```
$2$1
```

Changed result

```
1a a2 a3
b4 5a
none
```

RGX-15-001 · Try an over-permissive email shortcut

Replacing the local/domain rules by non-space runs admits consecutive dots, a leading domain hyphen and a domain without a final dot. These are outside the declared email policy.

Changed regexp

```
^([^[:space:]@]+)@([^[:space:]@]+)$
```

Replacement

```
local=$1 domain=$2
```

Changed result

```
local=ada domain=example.org
local=ada.lovelace+notes domain=research.example.org
local=team domain=lab-2.example.org
local=bad..dots domain=example.org
local=name domain=-example.org
local=name domain=example
missing-at.example.org
"quoted local"@example.org
```

RGX-16-001 · Check the numeric port range

After the same format match, read group 7 as a decimal number and require 0–65535. shape+port-ok reports just these checks; it does not certify a complete or reachable URL.

Format regexp

```
^(https?://([A-Za-z0-9]([A-Za-z0-9-]*[A-Za-z0-9])?([.][A-Za-z0-9]([A-Za-z0-9-]*[A-Za-z0-9])?)+)(:
([0-9]{1,5}))?([/?#][^[:space:]]*)?$
```

Additional check result

```
shape+port-ok   https://example.org
shape+port-ok   http://data.example.org:8080/run?q=regex#part
shape+port-ok   https://example.org?q=one
shape+port-ok   https://example.org/a%20b
port-out-of-range       https://example.org:99999/path
outside-profile ftp://example.org/file
outside-profile https://bad..example.org/path
outside-profile https://example.org/a b
```

RGX-17-001 · Verify the ORCID checksum

Apply MOD 11-2 to the first 15 digits: repeatedly double the running total after adding a digit. The resulting check character is compared with the last character, with 10 represented by X. The supplied 0098 ending must fail; this offline check does not query the registry.

Format regexp

```
^(https://orcid[.]org/)?([0-9]{4})-([0-9]{4})-([0-9]{4})-([0-9]{3}[0-9X])$
```

Additional check result

```
checksum-ok     0000-0002-1825-0097
checksum-ok     0000-0002-1694-233X
checksum-ok     0000-0002-1825-0097
bad-checksum(expected=7)        0000-0002-1825-0098
outside-format  0000-0002-1694-233x
outside-format  0000 0002 1825 0097
outside-format  0000-0002-1825-097
outside-format  https://example.org/0000-0002-1825-0097
```

RGX-18-001 · Check decimal octets and leading zeros

Require each captured octet to be at most 255 and reject multi-digit octets beginning with zero. This checks the dotted-decimal address syntax used here, not routing or reserved-address policy.

Format regexp

```
^([0-9]{1,3})[.]([0-9]{1,3})[.]([0-9]{1,3})[.]([0-9]{1,3})$
```

Additional check result

```
address-syntax-ok       192.0.2.1
address-syntax-ok       198.51.100.42
address-syntax-ok       0.0.0.0
address-syntax-ok       255.255.255.255
octet-out-of-range      256.1.2.3
leading-zero    192.168.001.1
outside-format  1.2.3
outside-format  192.0.2.1:80
```

RGX-19-001 · Select only client and server errors

Narrow status group 2 from [1-5][0-9]{2} to [45][0-9]{2}. The Perl check prints only captured 4xx/5xx records; successful 2xx requests disappear from this diagnostic view.

Changed regexp

```
^([0-9]{4}-[0-9]{2}-[0-9]{2}T[0-9]{2}:[0-9]{2}:[0-9]{2}Z) status=([45][0-9]{2}) request=([A-Za-z0-9-]+) path=[^[:space:]]+$
```

Changed result

```
time=2026-11-01T09:15:01Z code=404 id=req-18
time=2026-11-01T09:15:02Z code=503 id=req-19
```

RGX-20-001 · Try unanchored replacement inside prose

Searching inside prose can find more candidates, but a global replacement can retain a sentence-final period and duplicate an existing resolver prefix. These are real counterexamples: extraction and field normalization need different contracts. Crossref discusses punctuation bycatch; a match does not establish registration.

Changed regexp

```
(10[.][0-9]{4,9}/[-A-Za-z0-9._;()/:]+)
```

Replacement

```
https://doi.org/$1
```

Changed result

```
doi:https://doi.org/10.1234/ALPHA.7
https://doi.org/10.56789/notes-2
doi: https://doi.org/10.2468/data_set(3)
doi:10.123/short
doi:10.1234/
See https://doi.org/10.1234/ALPHA.7.
https://doi.org/https://doi.org/10.56789/notes-2
```

RGX-21-001 · Allow versions with one to three digits

Relax only group 5 from two digits to one through three. The previously excluded v2 record now transforms. Keep the supplied spelling: accepting variable width does not silently zero-pad the version.

Changed regexp

```
^([A-Za-z][A-Za-z0-9-]*)_([0-9]{4})([0-9]{2})([0-9]{2})_v([0-9]{1,3})[.]([a-z]{3})$
```

Replacement

```
${2}-${3}-${4}_${1}_v${5}.${6}
```

Changed result

```
2026-11-01_atlas_v02.csv
2026-10-31_atlas_v12.tsv
2026-02-29_rna-seq_v01.csv
2026-11-01_atlas_v2.csv
my_project_20261101_v02.csv
notes.txt
```

RGX-22-001 · Add a second pass for repeated whole words

First apply the whitespace cleanup above, then collapse adjacent identical whole words with a pattern backreference. The repeated group consumes the whole go go go run. Matching remains case-sensitive, so The the survives. The two cleanup decisions remain separate; the complete script shows both passes.

Changed regexp

```
\b([A-Za-z]+)([[:blank:]]+\1)+\b
```

Replacement

```
$1
```

Changed result

```
Keep spacing readable.
Use one separator.
the result
go
The the result
line break stays
```

RGX-23-001 · Check month lengths and century leap years

After the format match, parse captures as decimal values. Require year >= 1, month 1–12 and day within that month. A leap year is divisible by 400, or divisible by 4 but not 100. Emit an ISO date only for valid dates. The 1900/2000 examples test the century rule explicitly.

Format regexp

```
^([0-9]{2})[.]([0-9]{2})[.]([0-9]{4})$
```

Additional check result

```
calendar-ok     2026-11-01
calendar-ok     2024-02-29
invalid-date    29.02.2026
invalid-date    31.04.2026
invalid-date    29.02.1900
calendar-ok     2000-02-29
invalid-date    00.13.2026
outside-format  2026-11-01
```

RGX-24-001 · Try a global path-only rewrite

Remove the host and end constraints. The path-only pattern now rewrites other hosts, longer paths and even the nested /docs/ path inside the query value. Compare with the main result: host selection and tail preservation are part of the migration contract.

Changed regexp

```
/docs/([A-Za-z0-9-]+)/([0-9]{4})
```

Replacement

```
/guides/$2/$1
```

Changed result

```
https://old.example.org/guides/2026/regex
https://old.example.org/guides/2025/data-tools?view=all#top
https://old.example.org/guides/2026/regex?next=/guides/2024/awk
https://old.example.org.evil.test/guides/2026/regex
https://old.example.org/guides/2026/regex/extra
https://elsewhere.example.org/guides/2026/regex
```

Example and command index

- RGX-01-001 · Literal characters and metacharacters
- RGX-02-001 · Anchor the complete record
- RGX-03-001 · Choose a character class
- RGX-04-001 · Make parts optional or bounded
- RGX-05-001 · Group alternatives before anchoring
- RGX-06-001 · Define groups and swap their values
- RGX-07-001 · Number nested capture groups
- RGX-08-001 · Refer back to text already matched
- RGX-09-001 · Handle an unmatched optional capture
- RGX-10-001 · Name captures in Perl and map Emacs numbers
- RGX-11-001 · Group structure without capturing it
- RGX-12-001 · Distinguish the whole match from group 1
- RGX-13-001 · Limit greedy and minimal matches
- RGX-14-001 · Replace every occurrence with capture groups
- RGX-15-001 · Screen an ASCII email address
- RGX-16-001 · Screen HTTP(S) URLs and capture the host
- RGX-17-001 · Screen ORCID iDs and verify their check digit
- RGX-18-001 · Screen dotted IPv4 and check the octets
- RGX-19-001 · Extract fields from structured logs
- RGX-20-001 · Normalize DOI fields in literature data
- RGX-21-001 · Reassemble project filenames with captures
- RGX-22-001 · Normalize horizontal whitespace and repeated words
- RGX-23-001 · Reorder date fields and check calendar rules
- RGX-24-001 · Migrate URL paths while preserving query and fragment

---

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.