sd Simple find and replace

START HERE

`sd <find> <replace> <file>`
Replace in a file, in place

`sd -p <find> <replace> <file>`
Preview, change nothing

`<cmd> | sd <find> <replace>`
Filter a stream

`sd -F <text> <replace>`
Literal, no regex

`sd '(\w+)@(\w+)' '$2 at $1'`
Capture groups

`sd -A '\n' ' '`
Match across lines

`sd -n 1 <find> <replace>`
Only first match per file

`sd -f i <find> <replace>`
Ignore case

Input files or stdin → `line by line` → **Regex find** Rust syntax → `$1, $name` → **Replacement** $$ = literal $ → `in place / stdout` → **Output** file or stdout

CONCEPTS

Files are rewritten in place with no undo: run -p first, or work in a git repo.

Unlike sed, the pattern is a regex by default and every match is replaced: no s///g, no delimiters to escape.

Default mode reads one line at a time, so a pattern cannot contain a newline until you add -A.

BASICS

`sd <find> <replace> <file>...`
Replace all matches, editing files in place

`<cmd> | sd <find> <replace>`
Read stdin, write the result to stdout

`sd <find> <replace> < <file>`
Print the result without touching the file

`sd -p <find> <replace> <file>`
Preview the changes in a reviewable form

`sd -F <text> <replace>`
Treat find and replace as plain strings

`sd -n <count> <find> <replace>`
Limit replacements per file (0 = no limit)

`sd -A <find> <replace>`
Process input as a whole, allow multi-line matches
Uses more memory and stops streaming, so avoid it on huge inputs.

`sd -- <find> <replace>`
End options so a leading - is not a flag
Example: sd -- '--foo' '-w'

BASICS (cont.)

`sd --version`
Print the installed version

`sd --help`
Print the option summary

PATTERNS

`sd '\s+$' ''`
Trim trailing whitespace

`sd '^\s+' ''`
Trim leading whitespace

`sd '\d+' '<n>'`
Replace every run of digits

`sd '\bfoo\b' bar`
Whole word with word boundaries

`sd '^#.*\n' '' -A`
Drop comment lines (multi-line)

`sd 'a|b' c`
Alternation: either a or b

`sd '(?i)error' warn`
Inline case-insensitive group

`sd '\.' ','`
Escape regex metacharacters with a backslash

`sd '^$' '<text>'`
Fill empty lines

`sd '(?s)<a>.*</a>' '' -A`
Dot matches newlines with (?s)

REPLACEMENT

`sd '(\w+) (\w+)' '$2 $1'`
Swap two words with numbered groups

`sd '(?P<k>\w+)=(?P<v>\w+)' '$v=$k'`
Named groups, referenced as $name

`sd '(\w+)' '${1}_x'`
Braces end the group name: ${1}_x
$1_x would look for a group called 1_x and give empty output.

REPLACEMENT (cont.)

`sd foo '$$bar'`
Write a literal $ as $$

`sd '(a)(b)' '$0'`
$0 is the whole match

`sd -F '$' '<text>'`
With -F, $ in replacement stays literal

REGEX FLAGS

`sd -f i <find> <replace>`
Case-insensitive

`sd -f c <find> <replace>`
Case-sensitive (the default)

`sd -f w <find> <replace>`
Match whole words only

`sd -f m <find> <replace>`
Multi-line mode for ^ and $

`sd -f e <find> <replace>`
Disable multi-line matching

`sd -f s <find> <replace>`
Let . match newlines

`sd -f im <find> <replace>`
Combine flags in one value

FILES AND PIPELINES

`sd <find> <replace> <f1> <f2>`
Several files in one call

`fd -t f -x sd <find> <replace>`
Replace across a project with fd
fd runs sd once per file. Add -e <ext> to narrow the files.

`rg -l <find> | xargs sd <find> <replace>`
Touch only files that match

`git ls-files | xargs sd <find> <replace>`
Limit to tracked files

`echo <text> | sd <find> <replace>`
Test a pattern on a sample line

FROM SED

sed to sd

sed	sd
`sed 's/a/b/g'`	`sd a b`
`sed -i 's/a/b/g' f`	`sd a b f`
`sed -E 's/(a)(b)/\2\1/'`	`sd '(a)(b)' '$2$1'`
`sed 's/a/b/'`	`sd -n 1 a b`
`sed 's/a/b/I'`	`sd -f i a b`
`sed 'N;s/\n/,/'`	`sd -A '\n' ','`
`sed -n '/x/p'`	not supported: use rg or grep

sd only does find and replace: no line addressing, deletion by line number, or scripts. Reach for sed or awk for those.

-n 1 in sd limits per file, while sed s/a/b/ replaces the first match on every line.

RECIPES

Safe project-wide rename

```
git status --short                    # clean tree first
fd -t f -e js -x sd -p 'oldName' 'newName'
fd -t f -e js -x sd 'oldName' 'newName'
git diff
```

Reformat dates

```
echo '2026-10-21' | sd '(\d+)-(\d+)-(\d+)' '$3.$2.$1'
# 21.10.2026
```

Join lines into a CSV row

```
printf 'a\nb\nc\n' | sd -A '\n' ','
# a,b,c,
```

Replace literal text with specials

```
sd -F 'foo((([]))' 'bar' notes.txt
```