NAME

csplit — context-sensitive text splitting

SYNOPSIS

csplit [**-Szsk**] [**-f** *prefix*] [**-b** *suffix-format*|**-n** *suffix-digits*] *file*
{*lineno*|**/***regex***/**[±*offset*]|%*regex*%[±*offset*] [**{***repeat*|*****}**]} ...

DESCRIPTION

Splits consecutive lines of *file* (standard input stream if "**-**") based on the specified **Expressions** into output files created a=rw − *umask* (or truncated if they already exist).

Expressions

Lines start being copied to output file **0**. When the first expression matches a line, it and the lines following it are copied to file **1**, from the second — to file **2**, &c. All non-**{***}**ed expressions must match.

lineno	Matches the *lineno*th line.
/*regex***/**[±*offset*]	Matches the line that matches *regex* (which is a basic regular expression, cf. regex(7)), plus the optionally-signed *offset*. For example, in this document, /EXIT/−1 would match the "The file-name ..." line.
%*regex*%[±*offset*]	As above, but discard lines (and file) leading up to the match.

These may be optionally followed by a recurrence expression in the form

{*repeat***}**	which repeats the previous expression *repeat* times (*lineno*-style expressions grow by *lineno* each time), or
{*}**	which repeats the expression as many times as it'll fit.

OPTIONS

-S, **--suppress-matched**	Remove the lines that matched the expressions. This is usually equivalent to removing the first line of all but the zeroeth output file.
-z, **--elide-empty-files**	If an output file is empty, re-use it for the next expression.
-s, **--quiet**, **--silent**	By default, as each file is closed, its size is written to the standard output stream. This suppresses that.
-k, **--keep-files**	Don't delete output files on error.
-f, **--prefix**=*prefix*	Generate output file-names starting with *prefix*. Default: **xx**.
-b, **--suffix-format**=*format*	Format the number in the file-name according to *format*, which must contain exactly one printf(1) integer conversion.
-n, **--digits**=*suffix-digits*	Equivalent to **-b %0***suffix-digits***d**. Default: **2**.

The file-name for a file *n* is as-if **printf** "%s*suffix-format*" *prefix n*.

EXIT STATUS

1 if *file* or an output file couldn't be opened or written. **2** if an expression went unused. In either case, without **-k**, all output files are removed.

SIGNALS

All but SIGCONT, SIGSTOP, SIGTSTP without **-k**, all output files are removed before re-raising.

EXAMPLES

Extract each version from a Debian package changelog to octally-numbered files under /tmp, observing the size being, expectedly, the largest for the ...−1 versions:

```
$ csplit -Sf /tmp/ -b %#3o.log debian/changelog '/^ --/+1' {*} | paste -s
156     185     289     418     467     174     170
$ grep -m1 ^ /tmp/0*
/tmp/000.log:tzpfms (0.3.2-1~bpo11) bullseye; urgency=medium
/tmp/001.log:tzpfms (0.3.2-1) unstable; urgency=medium
/tmp/002.log:tzpfms (0.3.1-1) unstable; urgency=medium
```

```
/tmp/003.log:tzpfms (0.3.0-1) unstable; urgency=medium
/tmp/004.log:tzpfms (0.2.0-1) unstable; urgency=medium
/tmp/005.log:tzpfms (0.1.0-2) unstable; urgency=medium
/tmp/006.log:tzpfms (0.1.0-1) unstable; urgency=medium
```

SEE ALSO

`split`(1), which chunks the input based on the chunk size, or its **−p** extension which acts like **/***regex***/**.
The original can be re-combined by pasting the bits together (cf. `cat`(1)).

`printf`(1), `deb-changelog`(5), `regex`(7)

STANDARDS

Conforms to IEEE Std 1003.1-2024 (“POSIX.1”); the standard washes its hands of what happens if a *regex* expression matches the same line multiple times. This implementation matches the GNU system (in principle, anyway) in *always* making forward progress — for each line read, it may match the current expression, whereafter it’s either ejected or discarded. This means that, for a file containing consecutive non-empty lines, **/./***−1* **{***2***}** will make two one-line files, then the rest of the file in the third. Conversely, most other implementations, produce two empty files if even just one line matches, since the same expression matches multiple times with look-behind. This is usually less useful, since it makes repeating negative-offset *regex* expressions meaningless. In general, therefore, avoid repeating *regex*es with look-behind in portable applications.

POSIX is vague on the precise list of signals to be caught and cleaned up without **−k** — rather, it just specifies, in the **ASYNCHRONOUS EVENTS** sexion that they should be. This is taken here to mean "catch all signals" (sans the job control ones, natch); this list may be narrower in other implementations.

Nominally, *regex*es must match the BRE definition literally, and, hence, **$** must *not* match the new-line – i.e. **/***5$***/** has no matches. This is supremely annoying, given that each line, by definition, ends with one, and it’d need to be entered literally as part of the *regex*; therefore, as an extension, we compile the expressions with `REG_NEWLINE`, which means that **$** also matches the new-line – i.e. **/***5$***/** matches a line that ends with a "5". This is probably what you expect, and mimicks the GNU system and the illumos gate (though they also reject the required portable spelling of **/***5*
/, so.).

Only **−skfn** are standard: **−−suppress−matched**, **−zb** are extensions, originating from the GNU system (though that **−b** rejects some valid input and unconditionally overrides **−n**), and the **−S** spelling is an extension. **{*}** likewise originates from the GNU system, and works similarly thereto.

HISTORY

Appears in the Programmer’s Workbench (PWB/UNIX) as `csplit`(I) ("context split") with a **SYNOPSIS** of

    csplit [**−s**] [**−f** prefix] file [RE01 RE02 **...** REn]

with a hard limit of *n* < **100**, for a total of as many files, and *REn* equivalent to present-day **/***REn***/**. If no expressions are specified, it just degrades to **cp**. **−sf** are fully-formed, but the size of *file* is also noted by default; this is because **csplit** generates a `bfs`(I) (the "big file scanner", a generalised `ed`(1)-style read-only file processor) program, and **−s** corresponds to its **−**. As another side-effect, the expressions match with wrap-around, and, quite naturally, **$** matches the new-line.

The example usage is for COBOL, to edit a program separately in four sections:

    csplit −f zz file "procedure division" par5. par16.

AT&T System III UNIX sees a **SYNOPSIS** of

    csplit [**−s**] [**−k**] [**−f** prefix] file arg1 [... argn]

and can safely be considered fully-formed, with *arg*s of */rexp/*, *%rexp%* (both taking an optional offset), *lnno*, and {*num*} as present-day. *prefix*es longer than **12** bytes are rejected as too long, to match `DIRSIZ` (p.a. to modern `NAME_MAX`) of **14**.

Oddly, "Regular expressions may not contain embedded new-lines.", but they *must* do to match a new-line — **$** handling is as present-day. The **100**-file limit, noted likewise in PWB/UNIX, is actually enforced and causes an error, rather than being implicit through wrap-around. **−k** disables clean-up "if an error occurs"; the only caught signal is `SIGINT` (no signal handling is documented), and unmatched expressions become errors.

The *rexp*ression offset doesn't require a sign, but the wording of the manual implies it does. In general, the editorial choices are quite awful, starting at removing a possessive apostrophe, through putting *every* code sample at a different depth, and ending at describing how expressions cut up the file in terms of absolute motion ("A file is to be created for the section from the current line up to (but not including) the line containing the regular expression *rexp*. The current line becomes the line containing *rexp*. ...", &c.), which has no chance of producing the worst normative text you've ever read.

The **EXAMPLES** grow
```
    csplit −k file 100 {99}
```
to "split the file at every 100 lines, up to 10,000 lines." (and keep the chunks if the file's shorter) and
```
    csplit −k prog.c '%main(%' '/^}/+1' {20}
```
which, "Assuming that **prog.c** follows the normal **C** coding convention of ending routines with a } at the beginning of the line, this example will create a file containing each separate **C** routine (up to 21) in **prog.c**.", and is not the most bat-shit paragraph you've ever seen in a standards document. The prefix in the COBOL example becomes **cobol**, though.

AT&T System V Release 2 UNIX adds the "**-**"-as-standard-input-stream behaviour (by copying it entirely to a `tmpfile`(3), then processing that normally), checks only the final component of the *`prefix`* for length (such that you can do **`−f`** `/tmp/cicada/cobol`), and strips the final new-line from each processed line. Oddly, through the magic (weirdness) of `<regexp.h>`, this makes **/***5*
/ and **/***5$***/** equivalent.

AT&T System V Release 4 UNIX is ported to `<regexpr.h>`, which with it brings non-single-byte encoding support but removes the newline/end-of-string equivalence, so **/***5*
/ can never match, but **/***5$***/** matches a "5" at end-of-line.

Of all of these, only the gain of **-** is noted. AT&T System V Release 4 UNIX also adds a gratuitous "See `ed`(1) for a description of how to specify a regular expression." callback in the */rexp/* sexion (this seems like a back-solve from a similarly-placed similar call-back — "(Regular expressions as in ED(BU_CMD) are accepted.)" — in System V Interface Definition Issue 2 ("SVID2"), otherwise copying AT&T System V Release 2 UNIX `csplit`(1) verbatim (possibly also the System V Interface Definition ("SVID"), corresponding to AT&T System V Release 1 UNIX, but that doesn't appear to be extant)). By accident, the unchanging stanza of new-lines in regexes is the closest to making sense there as well.

The only innovation in the SVID is renaming *`lnno`* to *`line_no`* (and even that only partially). X/Open Portability Guide Issue 2 ("XPG2") additionally renames *`rexp`* to *`re`* and upper-cases '/PROCEDURE DIVISION/', and thus end the non-editorial differences; the **`ed`** reference is replaced with "Simple regular expression syntax is accepted.".

IEEE Std 1003.2a-1992 ("POSIX.2") largely formalises the SVID text, incl. the same unsung implementation details — **`−f`** is to early-exit if any part of the path would exceed `NAME_MAX`, as present-day, and removes the new-line-in-regex note. **−k** is for the first time described as influencing signal behaviour (as present-day, cf. **STANDARDS**), and present-day **−n** is invented (elsewhere rationalised to remove the hundred-file restrixion, though that's only ever implied in the standard).

The main trouble is *`rexp`*ressions, which are, similarly, formalised. Unfortunately, this is done by referencing them to sexion 2.8.3 ("Basic Regular Expressions", IEEE Std 1003.2-1992 ("POSIX.2")), which strictly states that the **$** anchor matches the end of the string. This is in stark contrast to SVID's **`ed`** reference (where it matches the new-line) as well as universal practice (with a small lapse from AT&T System III UNIX until AT&T System V Release 2 UNIX, and even then it was an obvious bug). Interestingly, for *`offset`*, "The integer value shall be preceded by + or −." — which is another unprecedented invention by an experience-free editor over-interpreting the original "This argument may be followed by an optional + or − some number of lines (e.g., `/page/−5`).".

A disclaimer of

> The use of repeated regular expressions with negative offsets produces bizarre results in historical implementations.

is added to match the as-present-day cop-out:

> If the selection of lines from an offset expression of this type would create a file with zero lines, or one with greater than the number of lines left in the input file, the results are unspecified.

X/Open Portability Guide Issue 4 ("XPG4") imports this essentially verbatim (dropping the "bizarre" note). Version 2 of the Single UNIX Specification ("SUSv2") adds **FUTURE DIRECTIONS** noting concerns, forwarded to be included from IEEE Std 1003.2b ("POSIX.2b": Shell and Utilities — Amendment). This adds escaping of **/** and **%** as **\/** and **\%** in *rexp*ressions to match "historical practice" (all implementations), and makes the *offset* sign optional, likewise (though through a contortionist sentence). With the same reasoning, rather than always matching *rexp* from the current line,

> If the current line is the first line in the file and an RE operation has not yet been performed, the pattern match of *rexp* shall be applied from the current line to the end of the file. Otherwise, the pattern match of *rexp* shall be applied from the line following the current line to the end of the file.

which makes **/***5***/ {***2***}** not eject two empty files. Version 3 of the Single UNIX Specification ("SUSv3") includes these changes, in addition to moving **csplit** to the User Portability Utilities feature group.

IEEE Std 1003.1-2008 ("POSIX.1") moves **csplit** back to the base spec, since its User Portability Utilities are exclusively interactive.