NAME

split — size- or line-wise data splitting

SYNOPSIS

split [**-vudx**] [**-a** *digits*] [**-l** *lines*|**-p** *expression*|**-C** *line-bytes*|**-b** *bytes*|
-n [**l/**|**r/**][*single***/**]*chunks* [**-e**]] [**-t** *linesep*]
[**--additional-suffix**=*suffix*] [**--filter**=*program*] [*file* [*prefix*]]

DESCRIPTION

Splits consecutive lines (by default, with **-l**, or with **-C**) or bytes (with **-b** or **-n**) of *file* (standard input stream if "**-**", the default) into files starting with *prefix* ("**x**" by default), followed by a consecutive number in the form:

by default, **aa**, **ab**, ..., **zz**,
with **-d**, **00**, **01**, ..., **99**, or
with **-x**, **00**, **01**, ..., **ff**,

and *suffix* (empty by default), created `a=rw` – *umask*, or runs *program*, or copies a *single* chunk to the standard output stream.

The default width is **2** and expands by two characters to fit more output files while preserving their names' lexicographical ordering: **aa**, ..., **yz**, **zaaa**, **zaab**, ..., **zyzz**, **zzaaaa**, **zaaaab**. Thus **cat** *prefix** can always be used to reconstruct *file*, except in **r/** mode. *digits* can be specified to just error out after too many files.

lines, *line-bytes*, *bytes*, *single*, and *chunks* are in the case-insensitive format:
[*base*][**KMGTPEZY**][**B**] (with at least one of {*base*, **KMGTPEZY**, **B**})
Where *base* is an optionally-floating-point number of bytes, defaulting to **1**, which is then optionally multiplied by the relevant unit. **B** sets the unit multiplier to **1000** (from **1024**). The argument is equal to $base \cdot unit^{mult}$, if any, or *base*.

OPTIONS

-l, **--lines**=*lines*
Split into at most *lines* lines per file. Defaults to **1000**.

-p *expression*
Split just before lines matching *expression* (which is an extended regular expression, cf. `regex`(7)). The first line is never split before.

-C, **--line-bytes**=*line-bytes*
Split into at most *line-bytes* bytes per file, but don't break lines (except if longer than *line-bytes*).

-b, **--bytes**=*bytes*
Split into at most *bytes* bytes per file.

-n, **--number**=*chunks*
Divide into *chunks* evenly-sized-rounded-down parts of at least **1** byte. The last output file accrues the entire remainder of *file*, including additional growth and rounding errors. If *file* runs out before *chunks* parts were yielded (because it shrunk), empty files are created.

-n, **--number=l/***chunks*
Likewise, but do not break lines. If a line runs over the edge of the chunk, the next chunk is smaller. If it spans multiple chunks, those chunks are empty.

-e, **--elide-empty-files**
Only create as many files as required. With **l/**, don't create empty files for run-over chunks either.

-n, **--number**=[**l/**]*single***/***chunks*
Like with just [**l/**]*chunks*, but write the *single*th chunk to the standard output stream and discard all others. Creates no files, excludes **--filter**, not affected by **-e**.

-n, **--number=r/***chunks*
Copy consecutive lines into consecutive files; return to the first file after every *chunks*th line.

-n, **--number=r/***first***/***chunks*
Copy every *chunks*th line to the standard output stream, starting with *first*.

-a, **--suffix-length**=*digits*	Use *digits* bytes for the file number; don't expand.
-t, **--separator**=*linesep*	Lines end in *linesep*, which must be a single byte or the literal "**\0**" for NUL (**0**), instead of a new-line (**0xA**).
-v, **--verbose**	Log output file names to the standard output stream.
-u, **--unbuffered**	Disable buffering on the standard output stream and all output files.
-d, **--numeric-suffixes**	Use characters from the `0123456789` alphabet to number files.
-d, **--numeric-suffixes**=*first*	Start at *first* instead of **0**.
-x, **--hex-suffixes**	Use characters from the `0123456789abcdef` alphabet to number files.
-x, **--hex-suffixes**=*first*	Start at *first* instead of **0**.
--filter=*program*	Instead of creating output files, run shell program *program* with the name in the `FILE` environment variable and the chunk fed to its standard input stream.

ENVIRONMENT

`SHELL` Pipe to `SHELL` **-c** *program* (defaults to `/bin/`**sh**).
`FILE` Set to the would-be output file-name for *program*.

EXIT STATUS

125	If *file*, output file, or the standard output stream couldn't be opened or written, **-n** and the size of *file* couldn't be determined, or **-a** and too many files.
128+*signal* (except `SIGPIPE`)	If *program* dies to *signal*. `SIGPIPE` is treated as a successful completion.
126	`SHELL` exists, but couldn't be executed for a different reason.
127	`SHELL` wasn't found.
All others	Bubbled if *program* exits non-zero.

SIGNALS

`SIGPIPE` If *program*: ignored (does not propagate to *program*); `EPIPE` (*program* exiting early) is ignored as well. Otherwise default.

EXAMPLES

Generate **512**-byte hexadecimal dumps:

```
# split -b 512 --filter 'od -A x -t x1z > "$FILE"' /dev/sda ~/sda-
# head -n 3 ~/sda-* | head -n 20
==> /root/sda-aa <==
000000 eb 63 90 00 00 00 00 00 00 00 00 00 00 00 00 00  >.c..............<
000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
*

==> /root/sda-ab <==
000000 45 46 49 20 50 41 52 54 00 00 01 00 5c 00 00 00  >EFI PART....\...<
000010 b7 fa 14 75 00 00 00 00 01 00 00 00 00 00 00 00  >...u............<
000020 af 0a 74 07 00 00 00 00 00 08 00 00 00 00 00 00  >..t.............<

==> /root/sda-ac <==
000000 48 61 68 21 49 64 6f 6e 74 4e 65 65 64 45 46 49  >Hah!IdontNeedEFI<
000010 c5 23 1b 81 d1 67 49 55 8b a6 90 ae d5 95 d5 ce  >.#...gIU........<
000020 00 08 00 00 00 00 00 00 ff 0f 00 00 00 00 00 00  >................<

==> /root/sda-ad <==
000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  >................<
```

```
*
000200
```

Demonstrate **r/** mode:

```
$ seq 10 | split -n r/4
$ paste -s x*
1       5       9       # xaa
2       6       10      # xab
3       7               # xac
4       8               # xad
```

SEE ALSO

`csplit`(1), which provides more complicated **-p**-style and line-number splitting. `head`(1), `tail`(1).

STANDARDS

Violates IEEE Std 1003.1-2024 ("POSIX.1") to provide the number-expansion behaviour, which matches the GNU system and NetBSD current — specify **-a** *2* explicitly to get standard behaviour. Unlike the aforementioned, this implementation stops expanding the width when it runs into `NAME_MAX` or `PATH_MAX`. Other implementations exit **1** for a generic error. In line-based modes, if the file doesn't end in a new-line (**0xA**), that tail is treated as a line and may start a new output file — this violates the standard for compatibility with historical implementations, NetBSD, the GNU system, and the illumos gate.

Only **-lba** and the "**k**" and "**m**" suffixes and only to *bytes* are standard. **-p** is an extension, originating from OpenBSD; **-dp** are also available on FreeBSD. Other flags are extensions, originating from the GNU system. **-n** *chunks* is also present on NetBSD and FreeBSD in a similar form except when

	there	**here**
dealing with pipes	size is somehow the size of the data in the first consumed buffer(?)	rejected,
dealing with devices	as-in-`stat`(2) (**0**)	underlying size used if defined, else error,
too-many *chunks* to get one byte per	refused	rounded up to a byte and filled out with empty files (matches the GNU system).

The **-v** spelling is an extension. The GNU system's **--numeric-suffixes**=*first* and **--hex-suffixes**=*first* suppress autoexpansion (as-if **-a** was specified) and don't auto-expand if *first* needs more than **2** bytes. Its **-u** provides softer no-buffer guarantees and *suffix*es with '**/**'es are rejected. It only allows unscaled *chunks* and disallows *lines*, *line-bytes*, and *bytes* with **B** but without a multiplier, as well as lower-case **B**, and only supports integer *base*s.

A heretofore-unnoted legacy **-***lines* argument format, equivalent to **-l** *lines*, is also accepted, for compatibility with Version 5 AT&T UNIX. Avoid it.

HISTORY

Appears in Version 3 AT&T UNIX as `split` (I):

```
NAME            split -- split a file into pieces

SYNOPSIS        split [ [ file1 ] file2 ]

FILES           -

SEE ALSO        --

DIAGNOSTICS     yes

BUGS            Watch out for 8-character file names.
```

file1 ("**-**"), *file2* ("**x**"), and **-l** *1000*-equivalent defaults match present-day. However, the suffix to *file2* consists of just '**a**', incremented ad infinitum (**a**, ..., **z**, **{**, **|**, ...).

Naturally, since file-names are at most **8** bytes, that's a very easy limit to over-run.

Version 4 AT&T UNIX grows directory entries to contain **14**-byte names, updating **BUGS**, and adds a **BUGS** of "The number of lines per file should be an argument.".

Version 5 AT&T UNIX sees a **SYNOPSIS** of

split **-***n* [file [name]]

with empty **BUGS**, **-***n* as present-day (though with **-***0* looping forever instead of being refused), and a two-byte initially-**aa** suffix, with present-day-but-unlimited **aa**, **ab**, ..., **zz**, **{a**, **{b**, ... progression. The buffer for the output file-name is **100** bytes and unchecked; this is largely inconsequential since it corresponds to over seven full-file-name directory levels under this system.

Version 7 AT&T UNIX segfaults for **-***0*.

AT&T System V Release 1 UNIX errors instead of creating the file following **zz**, documenting the hard **676**-file limit, and refuses *name*s whose basenames are longer than **12** bytes, thus ensuring the resulting output files fit within the unchanged **14**-byte NAME_MAX (both as present-day POSIX). This is misdocumented as just *name* exceeding that limit.

AT&T System V Release 4 UNIX refuses **-***0* and uses statvfs(2)'s *f_namemax* field for the output base-name limit.

4.3BSD–Tahoe sees a **SYNOPSIS** of

split [-**n**] [-**b** byte_cnt] [file [name]]

accepting, uncharacteristically for the time, both **-b** *10* and **-b***10*. **0** sizes are refused, the file-name buffer size is MAXPATHLEN (what we'd now call PATH_MAX), still unchecked.

If *name* isn't specified the default is, effectively, empty *prefix* and **-a** *3*. This persists in OpenBSD.

-b is as present-day but with just a plain number. The **EXIT STATUS**es for errors are wild — there's an ERREXIT macro which is **0** and only used sometimes; some other cases use ERR (**-1**(!)), which is for functions, not the program.

4.3BSD–Reno catches write errors and all short writes (as errors) instead of ignoring them, exits **1** for errors, and terminates on read errors in line mode (instead of noting the error and continuing).

4.4BSD sees a **SYNOPSIS** of

split [**-b** *byte_count*[*k*|*m*]] [**-l** *line_count*] [*file* [*name*]]

with **-***line_count* accepted as an "Undocumented kludge" and "**-**"-as-standard-input-stream accepted as "Undocumented: historic stdin flag.". Thus, splitting the standard input stream with a non-default *name* is for some reason undocumented? *byte_count* suffixes are as in POSIX.

X/Open Portability Guide Issue 2 ("XPG2") includes AT&T System V Release 1 UNIX **split** but with "**12**" re-spelled as "{NAME_MAX}-2".

IEEE Std 1003.2a-1992 ("POSIX.2") invents effectively-present-day **split** with a **Synopsis** of

split [-l *line_count*] [-a *suffix_length*] [*file* [*file*]]
split -b *n*[k|m] [-a *suffix_length*] [*file* [*file*]]
Obsolescent Version:
split [-*line_count*] [-a *suffix_length*] [*file* [*file*]]

-alb are as present-day (oddly, 4.4BSD does not mention compatibility with any standard). The *basename* of the output files being validated against NAME_MAX is finally correctly specified.

X/Open Portability Guide Issue 4 ("XPG4") imports IEEE Std 1003.2a-1992 ("POSIX.2") **split** verbatim,

IEEE Std 1003.1-2001 ("POSIX.1") moves **split** to the User Portability Utilities feature group and removes the obsolescent spelling.

IEEE Std 1003.1-2008 (“POSIX.1”) moves **split** back to the base spec and fixes a wording mishap requiring empty *file*s yielding one empty output file.