IEEE P1003.2a Draft 8 – December 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

5.25 `split` — Split files into pieces

5.25.1 Synopsis

`split [-l` *line_count* `] [-a` *suffix_length* `] [`*file* `[`*name* `]]`

`split -b` *n*`[k|m] [-a` *suffix_length* `] [`*file* `[`*name* `]]`

Obsolescent Version :

`split [-`*line_count* `] [-a` *suffix_length* `] [`*file* `[`*name* `]]`

5.25.2 Description

The `split` utility shall read an input file and write one or more output files. The default size of each output file shall be 1000 lines. The size of the output files can be modified by specification of the `-b` or `-l` options. Each output file shall be created with a unique suffix. The suffix shall consist of exactly *suffix_length* lowercase letters from the POSIX Locale. The letters of the suffix shall be used as if they were a base-26 digit system, with the first suffix to be created consisting of all '`a`'characters, the second with a '`b`'replacing the last '`a`', etc., until a name of all '`z`'s is created. By default, the names of the output files shall be '`x`', followed by a two-character suffix from the character set as described above, starting with `aa`, `ab`, `ac`, etc., and continuing until the suffix `zz`, for a maximum of 676 files.

If the number of files required exceeds the maximum allowed by the suffix length provided, such that the last allowable file would be larger than the requested size, the `split` utility shall fail after creating the last file with a valid suffix; `split` shall not delete the files it created with valid suffixes. If the file limit is not exceeded, the last file created shall contain the remainder of the input file, and may be smaller than the requested size.

5.25.3 Options

The `split` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that the obsolescent version allows a multidigit option, *-line_count* .

The following options shall be supported by the implementation:

`-a` *suffix_length*
: Use *suffix_length* letters to form the suffix portion of the filenames of the split file. If `-a` is not specified, the default suffix length shall be two. If the sum of the *name* operand and the *suffix_length* option-argument would create a filename exceeding {NAME _MAX} bytes, an error shall result; `split` shall exit with a diagnostic message and no files shall be created.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

`-b` *n* Split a file into pieces *n* bytes in size.

`-b` *n*`k` Split a file into pieces *n*∗1024 bytes in size.

`-b` *n*`m` Split a file into pieces *n*∗1 048 576 bytes in size.

`-l` *line_count*
-line_count (Obsolescent.)
Specify the number of lines in each resulting file piece. The *line_count* argument is an unsigned decimal integer. The default is 1000. If the input does not end with a <newline>, the partial line shall be included in the last output file.

5.25.4 Operands

The following operands shall be supported by the implementation:

file The pathname of the ordinary file to be split. If no input file is given or *file* is -, the standard input shall be used.

name The prefix to be used for each of the files resulting from the split operation. If no *name* argument is given, 'x'' shall be used as the prefix of the output files. The combined length of the basename of *prefix* and *suffix_length* cannot exceed {NAME_MAX} bytes; see 5.25.3.

5.25.5 External Influences

5.25.5.1 Standard Input

See Input Files.

5.25.5.2 Input Files

Any file can be used as input.

5.25.5.3 Environment Variables

The following environment variables shall affect the execution of `split`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

LC_CTYPE This variable shall determine the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files).

LC_MESSAGES This variable shall determine the language in which messages should be written.

5.25.5.4 Asynchronous Events

Default.

5.25.6 External Effects

5.25.6.1 Standard Output

None.

5.25.6.2 Standard Error

Used only for diagnostic messages.

5.25.6.3 Output Files

The output files contain portions of the original input file, otherwise unchanged.

5.25.7 Extended Description

None.

5.25.8 Exit Status

The `split` utility shall exit with one of the following values:

0 Successful completion.

>0 An error occurred.

5.25.9 Consequences of Errors

Default.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.25.10 Rationale. *(This subclause is not a part of P1003.2a)*

Usage, Examples

In the following examples `foo` is a text file that contains 5000 lines.

This example creates five files, `xaa`, `xab`, `xac`, `xad`, and `xae`:

```
split foo
```

This example also creates five files, but the suffixed portion of the created files consists of three letters, `xaaa`, `xaab`, `xaac`, `xaad`, and `xaae`:

```
split -a 3 foo
```

This example creates three files with four-letter suffixes and a supplied prefix, `bar_aaaa`, `bar_aaab`, and `bar_aaac`:

```
split -a 4 -l 2000 foo bar_
```

This example creates as many files as are necessary to contain at most 20∗1024 bytes, each with the default prefix of ‘`x`’’and a five-letter suffix:

```
split -a 5 -b 20k foo
```

History of Decisions Made

The −`b` option was added to provide a mechanism for splitting files other than by lines. While most uses of the −`b` option will be for transmitting files over networks, some felt it would have additional uses.

The −`a` option was added to overcome the limitation of being able to create only 676 files.

The −*line_count* option was declared obsolescent and replaced with the −`l` option. This allows future standards to have a `split` utility that fully conforms to the syntax guidelines.

Consideration was given to deleting this utility using the rationale that the function provided by this utility is available via the `csplit` utility (see 5.6). Upon reconsideration of the purpose of the User Portability Extension, it was decided to retain both this utility and the `csplit` utility because users use both utilities and have historical expectations of their behavior. Furthermore, the splitting on byte boundaries in `split` cannot be duplicated with the historical `csplit`.

The text ‘`split` shall not delete the files it created with valid suffixes’’ would normally be assumed, but since the related utility, `csplit`, does delete files under some circumstances, the historical behavior of `split` is made explicit to avoid misinterpretation. 7

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

5.26 `strings` — Find printable strings in files

5.26.1 Synopsis

`strings [-a][-t` *format* `][-n` *number* `][`*file* `...]`

Obsolescent Version :

`strings [-][-t` *format* `][-`*number* `][`*file* `...]`

5.26.2 Description

The `strings` utility shall look for printable strings in regular files and write those strings to standard output. A printable string is any sequence of four (by default) or more printable characters terminated by a `<newline>` or NUL character. Additional implementation-defined strings may be written. (See `localedef` in 4.35.)

5.26.3 Options

The `strings` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except that the obsolescent version uses − in a nonstandard way and allows a multidigit option, −*number* .

The following options shall be supported by the implementation:

`-a`
`-` (Obsolescent.)
: Scan files in their entirety. If `-a` is not specified, it is implementation defined what portion of each file is scanned for strings.

`-n` *number*
−*number* (Obsolescent.)
: Specify the minimum string length, where the *number* argument is a positive decimal integer. The default shall be 4.

`-t` *format*
: Write each string preceded by its byte offset from the start of the file. The format shall be dependent on the single character used as the *format* option-argument:

  `d` The offset shall be written in decimal.

  `o` The offset shall be written in octal.

  `x` The offset shall be written in hexadecimal.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.