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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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]

5.26 `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.

5.26.4 Operands

The following operand shall be supported by the implementation:

file A pathname of a regular file to be used as input. If no *file* operand is specified, the `strings` utility shall read from the standard input.

5.26.5 External Influences

5.26.5.1 Standard Input

See Input Files.

5.26.5.2 Input Files

The input files named by the utility arguments or the standard input shall be regular files of any format.

5.26.5.3 Environment Variables

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

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_**.

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), and to identify printable strings.

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

5.26.5.4 Asynchronous Events

Default.

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

5.26.6 External Effects

5.26.6.1 Standard Output

Strings found shall be written to the standard output, one per line.

When the `-t` option is not specified, the format of the output shall be:

"`%s`", <*string*>

With the `-t o` option, the format of the output shall be:

"`%o %s`", <*byte offset*>, <*string*>

With the `-t x` option, the format of the output shall be:

"`%x %s`", <*byte offset*>, <*string*>

With the `-t d` option, the format of the output shall be:

"`%d %s`", <*byte offset*>, <*string*>

5.26.6.2 Standard Error

Used only for diagnostic messages.

5.26.6.3 Output Files

None.

5.26.7 Extended Description

None.

5.26.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.26.9 Consequences of Errors

Default.

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

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

Usage, Examples

The `strings` utility is useful for identifying random object files and many other things.

History of Decisions Made

Apart from rationalizing the option syntax and slight difficulties with object and executable binary files, `strings` is specified to closely match existing practice. The `-a` and `-n` options were introduced to replace the nonconforming `-` and *-number* options.

Investigation revealed that `-o` historically means different things on different implementations. Some maintain that it means "*offset* in decimal," while others maintain that it means "offset in *octal*." Instead of trying to decide which way would be least objectionable, the `-t` option was added. It was originally named `-O` to mean "offset,"but was changed to `-t` to be consistent with `od`.

By default, the data area (as opposed to the text, "bss," or header areas) of a binary executable file is scanned in historical implementations. Implementations are required to document which areas are scanned.

Some historical implementations do not require NUL or `<newline>` terminators for strings to permit those languages that do not use NUL as a string terminator to have their strings written.

The C Standard {7} function *isprint* () is restricted to a domain of *unsigned char*. This standard requires implementations to write strings as defined by the current locale.

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

5.27 `tabs` — Set terminal tabs

5.27.1 Synopsis

`tabs` [*−n*] [`−T` *type*]

`tabs` [`−T` *type*] *n1* [`,`*n2*`,...`]

5.27.2 Description

The `tabs` utility shall display a series of characters that first clears the hardware terminal tab settings and then initializes the tab stops at the specified positions.

The phrase ''tab-stop position *N*'' shall be taken to mean that, from the start of a line of output, tabbing to position *N* shall cause the next character output to be in the (*N*+1)th column position on that line. The maximum number of tab stops allowed is terminal dependent.

It need not be possible to implement `tabs` on certain terminals. If the terminal type obtained from the **TERM** environment variable or `−T` option represents such a terminal, an appropriate diagnostic message shall be written to standard error and `tabs` shall exit with a status greater than zero.

5.27.3 Options

The `tabs` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation:

−n Specify repetitive tab stops separated by a uniform number of column positions, *n*, where *n* is a single-digit decimal number. The default usage of `tabs` with no arguments shall be equivalent to `tabs −8`. When `−0` is used, the tab stops shall be cleared and no new ones set.

`−T` *type* Indicate the type of terminal. If this option is not supplied and the **TERM** variable is unset or null, an unspecified default terminal type shall be used. The setting of *type* shall take precedence over the value in **TERM**. 8 8

5.27.4 Operands

The following operand shall be supported by the implementation:

n1 [`,`*n2*`,...`]
A single command-line argument that consists of tab-stop values separated using either commas or `<blank>`s. The tab-stop values shall be positive decimal integers in strictly ascending order. If

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