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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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

transfer between architectures is generally character data.

5.34 uuencode — Encode a binary file

5.34.1 Synopsis

`uuencode` [*file*] *decode_pathname*

5.34.2 Description

The `uuencode` utility shall write an encoded version of the named input file, or standard input if no *file* is specified, to standard output. The output shall be encoded using the algorithm described in 5.34.6.1, and shall include the file access permission bits (in `chmod` octal or symbolic notation; see 4.7) of the input file and the *decode_pathname* , for recreation of the file on another system that conforms to this standard.

5.34.3 Options

None.

5.34.4 Operands

The following operands shall be supported by the implementation:

decode_pathname
: The pathname of the file into which the `uudecode` utility (see 5.33) shall place the decoded file. If there are characters in *decode_pathname* that are not in the portable filename character set (see 2.2.2.111 in POSIX .1 {8}),the results are unspecified.

file
: A pathname of the file to be encoded.

5.34.5 External Influences

5.34.5.1 Standard Input

See Input Files.

5.34.5.2 Input Files

Input files can be files of any type.

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

6138 #### 5.34.5.3 Environment Variables

6139 The following environment variables shall affect the execution of `uuencode`:

6140 **LANG** This variable shall determine the locale to use for the
6141 locale categories when both **LC_ALL** and the correspond-
6142 ing environment variable (beginning with **LC_**) do not
6143 specify a locale. See 2.6.

6144 **LC_ALL** This variable shall determine the locale to be used to over-
6145 ride any values for locale categories specified by the set-
6146 tings of **LANG** or any environment variables beginning
6147 with **LC_**.

6148 **LC_CTYPE** This variable shall determine the locale for the interpreta-
6149 tion of sequences of bytes of text data as characters (e.g.,
6150 single- versus multibyte characters in arguments and
6151 data).

6152 **LC_MESSAGES** This variable shall determine the language in which mes-
6153 sages should be written.

6154 #### 5.34.5.4 Asynchronous Events

6155 Default.

6156 ### 5.34.6 External Effects

6157 #### 5.34.6.1 Standard Output

6158 The standard output shall be a text file (encoded in the character set of the 7
6159 current locale) that begins with the line: 7

6160 `"begin␣%s␣%s\n"`, *<mode>*, *decode_pathname*

6161 and ends with the line:

6162 `"end\n"`

6163 In both cases, the lines shall have no preceding or trailing `<blank>`s. 7

6164 The algorithm that shall be used for lines in between `begin` and `end` takes three
6165 octets as input and writes four characters of output by splitting the input at six- 7
6166 bit intervals into four octets, containing data in the lower six bits only. These 7
6167 octets shall be converted to characters by adding a value of 0x20 to each octet, so 7
6168 that each octet is in the range 0x20–0x5f, and then it shall be assumed to 7
6169 represent a printable character in the ISO/IEC 646 {1} encoded character set. It 7
6170 then shall be translated into the corresponding character codes for the code set in 7
6171 use in the current locale. (For example, the octet 0x41, representing `A`, would be 7
6172 translated to `A` in the current code set, such as 0xc1 if it were EBCDIC.) 7

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

Where the bits of two octets are combined, the least significant bits of the first octet shall be shifted left and combined with the most significant bits of the second octet shifted right. Thus the three octets A, B, C are converted into the four octets:

```
0x20 + (( A >> 2                    ) & 0x3F)
0x20 + (((A << 4) | ((B >> 4) & 0xF)) & 0x3F)
0x20 + (((B << 2) | ((C >> 6) & 0x3)) & 0x3F)
0x20 + (( C                         ) & 0x3F)
```

These octets are then translated into the local character set.

Each encoded line shall contain a length character, equal to the number of characters to be decoded plus 0x20 translated to the local character set as described above, followed by the encoded characters. The maximum number of octets to be encoded on each line shall be 45.

5.34.6.2 Standard Error

Used only for diagnostic messages.

5.34.6.3 Output Files

None.

5.34.7 Extended Description

None.

5.34.8 Exit Status

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

0 Successful completion.

>0 An error occurred.

5.34.9 Consequences of Errors

Default.

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

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

Usage, Examples

The file is expanded by 35 percent (each three octets become four, plus control information) causing it to take longer to transmit.

History of Decisions Made

Since this utility is intended to create files to be used for data interchange between systems with possibly different code sets, and to represent binary data as a text file, ISO/IEC 646 {1} was chosen for a midpoint in the algorithm as a known reference point. The output from `uuencode` is a text file on the local system. If the output were in the ISO/IEC 646 {1} code set, it might not be a text file (at least because the `<newline>`s might not match), and the goal of creating a text file would be defeated. If this text file was then carried to another machine with the same code set, it would be perfectly compatible with that system's `uudecode`. If it was transmitted over a mail system or sent to a machine with a different code set, it is assumed that, as for every other text file, some translation mechanism would convert it (by the time it reached a user on the other system) into an appropriate code set. This translation only makes sense from the local code set, not if the file has been put into an ISO/IEC 646 {1} representation first. Similarly, `uuencode`d files can be placed in `pax` archives, intermixed with other text files in the same code set.

The algorithm is described in terms of 8-bit quantities, or octets. Since no byte alignment is implied, it will encode data from machines with any number of bits per byte. However, unless that encoded data is then decoded on a machine with the same number of bits per byte, the output might not be useful.

The term ''source file'' in previous drafts was changed to ''file'' to avoid any implied relationship to computer-language source code files.

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

5.35 vi — Screen-oriented (visual) display editor

This utility shall be provided on systems that both support the User Portability Utilities Option and define the {POSIX 2_CHAR _TERM } symbol. On other systems it is optional.

5.35.1 Synopsis

`vi [−rR][−c` *command* `][−t` *tagstring* `][−w` *size* `][`*file* `...]`

Obsolescent Version :

`vi [−rR][+`*command* `][−t` *tagstring* `][−w` *size* `][`*file* `...]`

5.35.2 Description

The `vi` (visual) utility is a screen-oriented text editor. The user can switch back and forth between `vi` and the line editor `ex` (see 5.10) and execute `ex` commands from within `vi`.

When using `vi`, the terminal screen acts as a window into the editing buffer. Changes made to the editing buffer shall be reflected in the screen display; the position of the cursor on the screen shall indicate the position within the editing buffer.

Certain block-mode terminals do not have all the capabilities necessary to support the complete `vi` definition. When these commands cannot be supported on such terminals, this condition shall not produce an error message such as ‘‘not an editor command’’ or report a syntax error. The implementation may either accept the commands and produce results on the screen that are the result of an unsuccessful attempt to meet the requirements of this standard or report an error describing the terminal-related deficiency.

5.35.3 Options

The `vi` utility shall conform to the utility argument syntax guidelines described in 2.10.2, except for the obsolescent +*command* ‘‘option.’’

The following options shall be supported by the implementation:

`−c` *command*
+*command* (Obsolescent.)
Begin editing by executing the specified `ex` command-mode command(s). As with normal `ex` command-line entries, the *command* option-argument can consist of multiple `ex` commands separated by vertical-line characters (|). The use of commands that enter input mode in this manner produces undefined results.

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