Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'uuencode.1p' command

$ man uuencode.1p

UUENCODE(1P) POSIX Programmer's Manual UUENCODE(1P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

uuencode ? encode a binary file

SYNOPSIS

uuencode [-m] [file] decode_pathname

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 one of the algorithms described in the STDOUT section and shall include the file access permission bits (in chmod octal or symbolic notation) of the input file and the de? code_pathname, for re-creation of the file on another system that con? forms to this volume of POSIX.1?2017.

OPTIONS

The uuencode utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines.

The following option shall be supported by the implementation:

-m Encode the output using the MIME Base64 algorithm described

in STDOUT. If -m is not specified, the historical algorithm described in STDOUT shall be used.

OPERANDS

The following operands shall be supported:

decode_pathname

The pathname of the file into which the uudecode utility shall place the decoded file. Specifying a decode_pathname operand of /dev/stdout shall indicate that uudecode is to use standard output. If there are characters in decode_pathname that are not in the portable filename character set the re? sults are unspecified.

file A pathname of the file to be encoded.

STDIN

See the INPUT FILES section.

INPUT FILES

Input files can be files of any type.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of uuen? code:

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

LC_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard er?

ror.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

ASYNCHRONOUS EVENTS

Default.

STDOUT

uuencode Base64 Algorithm

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

"begin-base64 %s %s\n", <mode>, <decode_pathname>

and ends with the line:

"====\n"

In both cases, the lines shall have no preceding or trailing <blank> characters.

The encoding process represents 24-bit groups of input bits as output strings of four encoded characters. Proceeding from left to right, a 24-bit input group shall be formed by concatenating three 8-bit input groups. Each 24-bit input group then shall be treated as four concate? nated 6-bit groups, each of which shall be translated into a single digit in the Base64 alphabet. When encoding a bit stream via the Base64 encoding, the bit stream shall be presumed to be ordered with the most-significant bit first. That is, the first bit in the stream shall be the high-order bit in the first byte, and the eighth bit shall be the low-order bit in the first byte, and so on. Each 6-bit group is used as an index into an array of 64 printable characters, as shown in Table 4-22, uuencode Base64 Values.

Table 4-22: uuencode Base64 Values

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1

? 3 ? D ?? 20 ? U ?? 37 ? l ?? 54 ? 2 ?

? 4 ? E ?? 21 ? V ?? 38 ? m ?? 55 ? 3 ?

? 5 ? F ?? 22 ? W ?? 39 ? n ?? 56 ? 4 ?

? 6 ? G ?? 23 ? X ?? 40 ? o ?? 57 ? 5 ?

? 7 ? H ?? 24 ? Y ?? 41 ? p ?? 58 ? 6 ?

? 8 ? I ?? 25 ? Z ?? 42 ? q ?? 59 ? 7 ?

? 9 ? J ?? 26 ? a ?? 43 ? r ?? 60 ? 8 ?

? 10 ? K ?? 27 ? b ?? 44 ? s ?? 61 ? 9 ?

? 11 ? L ?? 28 ? c ?? 45 ? t ?? 62 ? + ?

? 12 ? M ?? 29 ? d ?? 46 ? u ?? 63 ? / ?

? 13 ? N ?? 30 ? e ?? 47 ? v ?? ? ?

? 14 ? O ?? 31 ? f ?? 48 ? w ??(pad) ? = ?

? 15 ? P ?? 32 ? g ?? 49 ? x ?? ? ?

? 16 ? Q ?? 33 ? h ?? 50 ? y ?? ? ?

??

The character referenced by the index shall be placed in the output string.

The output stream (encoded bytes) shall be represented in lines of no more than 76 characters each. All line breaks or other characters not found in the table shall be ignored by decoding software (see uude? code).

Special processing shall be performed if fewer than 24 bits are avail? able at the end of a message or encapsulated part of a message. A full encoding quantum shall always be completed at the end of a message.

When fewer than 24 input bits are available in an input group, zero bits shall be added (on the right) to form an integral number of 6-bit groups. Output character positions that are not required to represent actual input data shall be set to the character '='. Since all Base64 input is an integral number of octets, only the following cases can arise:

1. The final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output shall be an integral multiple of 4 characters with no '=' padding.

2. The final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output shall be three characters followed by one '=' padding character.

3. The final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output shall be two characters followed by two '=' padding characters.

A terminating "====" evaluates to nothing and denotes the end of the encoded data.

uuencode Historical Algorithm

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

"begin %s %s\n" <mode>, <decode_pathname>

and ends with the line:

"end\n"

In both cases, the lines shall have no preceding or trailing <blank> characters.

The algorithm that shall be used for lines in between begin and end takes three octets as input and writes four characters of output by splitting the input at six-bit intervals into four octets, containing data in the lower six bits only. These octets shall be converted to characters by adding a value of 0x20 to each octet, so that each octet is in the range [0x20,0x5f], and then it shall be assumed to represent a printable character in the ISO/IEC 646:1991 standard encoded charac? ter set. It then shall be translated into the corresponding character codes for the codeset in use in the current locale. (For example, the octet 0x41, representing 'A', would be translated to 'A' in the current codeset, such as 0xc1 if it were EBCDIC.)

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 shall be 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 then shall be translated into the local character set.

Each encoded line contains 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.

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

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

Since this utility is intended to create files to be used for data in? terchange between systems with possibly different codesets, and to rep? resent binary data as a text file, the ISO/IEC 646:1991 standard 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 646:1991 standard codeset, it might not be a text file (at least because the <newline> characters 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 codeset, it would be perfectly compatible with that system's uudecode. If it was transmit?

ted over a mail system or sent to a machine with a different codeset, 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 codeset. This translation only makes sense from the local codeset, not if the file has been put into a ISO/IEC 646:1991 standard representation first. Similarly, files pro? cessed by uuencode can be placed in pax archives, intermixed with other text files in the same codeset.

EXAMPLES

None.

RATIONALE

A new algorithm was added at the request of the international community to parallel work in RFC 2045 (MIME). As with the historical uuencode format, the Base64 Content-Transfer-Encoding is designed to represent arbitrary sequences of octets in a form that is not humanly readable. A 65-character subset of the ISO/IEC 646:1991 standard is used, enabling 6 bits to be represented per printable character. (The extra 65th char? acter, '=', is used to signify a special processing function.)

This subset has the important property that it is represented identi? cally in all versions of the ISO/IEC 646:1991 standard, including US ASCII, and all characters in the subset are also represented identi? cally in all versions of EBCDIC. The historical uuencode algorithm does not share this property, which is the reason that a second algorithm was added to the ISO POSIX?2 standard.

The string "====" was used for the termination instead of the end used in the original format because the latter is a string that could be valid encoded input.

In an early draft, the -m option was named -b (for Base64), but it was renamed to reflect its relationship to the RFC 2045. A -u was also present to invoke the default algorithm, but since this was not histor? ical practice, it was omitted as being unnecessary.

See the RATIONALE section in uudecode for the derivation of the /dev/stdout symbol.

FUTURE DIRECTIONS

None.

SEE ALSO

chmod, mailx, uudecode

The Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guidelines

COPYRIGHT