NAME

shar − create shell archives

SYNOPSIS

shar [options] file ...
shar −S [options]

DESCRIPTION

Shar creates "shell archives" (or shar files) which are in text format and can be mailed. These files may be unpacked later by executing them with /bin/sh. The resulting archive is sent to standard out unless the −*o* option is given. A wide range of features provide extensive flexibility in manufacturing shars and in specifying shar "smartness". Archives may be "vanilla" or comprehensive.

OPTIONS

Options have a one letter version starting with − or a long version starting with −−. The exception is −−*help*, −−*version*, −−*no-i18n* and −−*print-text-domain-dir* which does not have short versions. Mandatory arguments to long options are mandatory for short options too. Options can be given in any order. Some options depend on each other:

The −*o* option is required if the −*l* or −*L* option is used.
The −*n* option is required if the −*a* option is used.
See −*V* below.

Giving feedback:

−−*help* Print a help summary on standard output, then immediately exits.

−−*version*
Print the version number of the program on standard output, then immediately exits.

−*q* −−*quiet* −−*silent*
Do not output verbose messages locally when producing the archive.

Selecting files:

−*p* −−*intermix-type*
Allow positional parameter options. The options −*B*, −*T*, -*z* and −*Z* may be embedded, and files to the right of the option will be processed in the specified mode.

−*S* −−*stdin-file-list*
Read list of files to be packed from the standard input rather than from the command line. Input must be in a form similar to that generated by the find command, one filename per line. This switch is especially useful when the command line will not hold the list of files to be packed. For example:

find . −type f −print | sort | shar −S −Z −L50 −o /tmp/big

If −*p* is specified on the command line, then the options −*B*, −*T*, −*z* and −*Z* may be included in the standard input (on a line separate from filenames). The maximum number of lines of standard input, file names and options, may not exceed 1024.

Splitting output:

−*o* XXX −−*output-prefix*=XXX
Save the archive to files XXX.01 thru XXX.nn instead of sending it to standard out. Must be used when the −*l* or the −*L* switches are used.

−*l* XX −−*whole-size-limit*=XX
Limit the output file size to XXk bytes but don’t split input files.

−*L* XX −−*split-size-limit*=XX
Limit output file size to XXk bytes and split files if necessary. The archive parts created with this option must be unpacked in correct order.

Controlling the shar headers:

−*n* name −−*archive-name*=name
Name of archive to be included in the header of the shar files. See the −*a* switch.

−*s* who@where −−*submitter*=who@where
Override automatically determined submitter name.

−*a* −−*net-headers*
Allows automatic generation of headers:
Submitted-by: who@where
Archive-name: <name>/part##
The <name> must be given with the −*n* switch. If name includes a '/' "/part" isn't used. Thus:

−n xyzzy	produces: xyzzy/part01 xyzzy/part02
−n xyzzy/patch	produces: xyzzy/patch01 xyzzy/patch02
−n xyzzy/patch01.	produces: xyzzy/patch01.01 xyzzy/patch01.02

The who@where can be explicitly stated with the −*s* switch if the default isn't appropriate. Who@where is essentially built as ‘whoami‘@‘uname‘.

−*c* −−*cut-mark*
Start the shar with a cut line. A line saying 'Cut here' is placed at the start of each output file.

Selecting how files are stocked:

−*M* −−*mixed-uuencode*
Mixed mode. Determine if the files are text or binary and archive correctly (default). Files found to be binary are uudecoded prior to packing (USE OF UUENCODE IS NOT APPRECIATED BY MANY ON THE NET).

−*T* −−*text-files*
Treat all files as text.

−*B* −−*uuencode*
Treat all files as binary, use uuencode prior to packing. This increases the size of the archive. The recipient must have uudecode in order to unpack. (USE OF UUENCODE IS NOT APPRECIATED BY MANY ON THE NET).

−*z* −−*gzip*
Gzip and uuencode all files prior to packing. The recipient must have uudecode and gzip in order to unpack (USE OF UUENCODE AND GZIP IS NOT APPRECIATED BY MANY ON THE NET).

−*g* LEVEL −−*level-for-gzip*=LEVEL
When doing compression, use '−LEVEL' as a parameter to gzip. Default is 9. The −*g* option turns on the −*z* option by default.

−*Z* −−*compress*
Compress and uuencode all files prior to packing. The recipient must have uudecode and compress in order to unpack (USE OF UUENCODE AND COMPRESS IS NOT APPRECIATED BY MANY ON THE NET). Option −*C* is synonymous to −*Z*, but is being deprecated.

−*b* BITS −−*bits-per-code*=BITS
When doing compression, use '−bBITS' as a parameter to compress. The −*B* option turns on the −*Z* option by default. Default value is 12.

Protecting against transmission errors:

−*w* −−*no-character-count*
Do NOT check each file with 'wc −c' after unpack. The default is to check.

−*D* −−*no-md5-digest*
Do NOT use 'md5sum' digest to verify the unpacked files. The default is to check.

−F −−force-prefix
Forces the prefix character (normally 'X' unless the parameter to the *−d* option starts with 'X') to be prepended to every line even if not required. This option may slightly increase the size of the archive, especially if *−B* or *−Z* is used.

−d XXX *−−here-delimiter*=XXX
Use XXX to delimit the files in the shar instead of SHAR_EOF. This is for those who want to personalize their shar files.

Producing different kinds of shars:

−V −−vanilla-operation
Produce "vanilla" shars which rely only upon the existence of sed and echo in the unsharing environment. In addition, "if test" must also be supported unless the *−x* option is used. The *−V* silently disables options offensive to the "network cop" (or "brown shirt"), but does warn you if it is specified with *−B*, *-z*, *−Z*, *−p* or *−M* (any of which does or might require uudecode, gzip or compress in the unsharing environment).

−P −−no-piping
Use temporary files instead of pipes in the shar file.

−x −−no-check-existing
Overwrite existing files without checking. If neither *−x* nor *−X* is specified, the unpack will check for and not overwrite existing files when unpacking the archive. If *−c* is passed as a parameter to the script when unpacking:

sh archive -c

then existing files will be overwritten unconditionally.

−X −−query-user
When unpacking, interactively ask the user if files should be overwritten. (DO NOT USE FOR SHARS SUBMITTED TO THE NET).

−m −−no-timestamp
Avoid generating 'touch' commands to restore the file modification dates when unpacking files from the archive.

−Q −−quiet-unshar
Verbose OFF. Disables the inclusion of comments to be output when the archive is unpacked.

−f −−basename
Restore by filename only, rather than path. This option causes only file names to be used, which is useful when building a shar from several directories, or another directory. Note that if a directory name is passed to shar, the substructure of that directory will be restored whether *−f* is specified or not.

Internationalization:

−−no-i18n
Do not produce internationalized shell archives, use default english messages. By default, shar produces archives that will try to output messages in the unpackers preferred language (as determined by the LANG/LC_MESSAGES environmental variables) when they are unpacked. If no message file for the unpackers language is found at unpack time, messages will be in english.

−−print-text-domain-dir
Prints the directory shar looks in to find messages files for different languages, then immediately exits.

EXAMPLES

```
shar *.c > cprog.shar                  # all C prog sources
shar −Q *.[ch] > cprog.shar            # non-verbose, .c and .h files
shar −B −l28 −oarc.sh *.arc            # all binary .arc files, into
                                       # files arc.sh.01 thru arc.sh.NN
shar −f /lcl/src/u*.c > u.sh           # use only the filenames
```

WARNINGS

No chmod or touch is ever generated for directories created when unpacking. Thus, if a directory is given to shar, the protection and modification dates of corresponding unpacked directory may not match those of the original.

If a directory is passed to shar, it may be scanned more than once. Therefore, one should be careful not change the directory while shar is running.

Be careful that the output file(s) are not included in the inputs or shar may loop until the disk fills up. Be particularly careful when a directory is passed to shar that the output files are not in that directory (or a subdirectory of that directory).

Use of the *−B*, *−z* or *−Z*, and especially *−M*, may slow the archive process considerably, depending on the number of files.

Use of *−X* produces shars which *WILL* cause problems with many unshar procedures. Use this feature only for archives to be passed among agreeable parties. Certainly, *−X* is NOT for shell archives which are to be submitted to Usenet. Usage of *−B*, *−z* or *−Z* in net shars will cause you to be flamed off the earth. Not using *−m* or not using *−F* may also get you occasional complaints.

SEE ALSO

unshar(1)

DIAGNOSTICS

Error messages for illegal or incompatible options, for non-regular, missing or inaccessible files or for (unlikely) memory allocation failure.

AUTHORS

The shar and unshar programs is the collective work of many authors. Many people contributed by reporting problems, suggesting various improvements or submitting actual code. A list of these people is in the THANKS file in the sharutils distribution.

NAME

unshar − unpack a shar file

SYNOPSIS

unshar [options] [file ...]

DESCRIPTION

Unshar scans mail messages looking for the start of a shell archive. It then passes the archive through a copy of the shell to unpack it. It will accept multiple files. If no files are given, standard input is used.

OPTIONS

Options have a one letter version starting with − or a long version starting with −−. The exception is −−*help* and −−*version*, which does not have a short version.

−−*version*
: Print the version number of the program on standard output, then immediately exits.

−−*help* Print a help summary on standard output, then immediately exits.

−*d* DIRECTORY −−*directory*=DIRECTORY
: Change directory to DIRECTORY before unpacking any files.

−*c* −−*overwrite*
: Passed as an option to the shar file. Many shell archive scripts (including those produced by ‘shar’ 3.40 and newer) accepts a −*c* argument to indicate that existing files should be overwritten.

−*e* −−*exit-0*
: This option exists mainly for people who collect many shell archives into a single mail folder. With this option, ‘unshar’ isolates each different shell archive from the others which have been put in the same file, unpacking each in turn, from the beginning of the file towards its end. Its proper operation relies on the fact that many shar files are terminated by a ‘exit 0’ at the beginning of a line.

  Option −*e* is internally equivalent to −*E* "exit 0".

−*E* STRING −−*split-at*=STRING
: This option works like −*e*, but it allows you to specify the string that separates archives if ‘exit 0’ isn’t appropriate.

  For example, noticing that most ‘.signatures’ have a ‘−−’ on a line right before them, one can sometimes use ‘−−*split-at*=−−’ for splitting shell archives which lack the ‘exit 0’ line at end. The signature will then be skipped altogether with the headers of the following message.

−*f* −−*force*
: The same as −*c*.

SEE ALSO

shar(1)

DIAGNOSTICS

Any message from the shell may be displayed.

AUTHORS

The shar and unshar programs is the collective work of many authors. Many people contributed by reporting problems, suggesting various improvements or submitting actual code. A list of these people is in the THANKS file in the sharutils distribution.

NAME

uuencode − encode a binary file

uudecode − decode a file created by uuencode

SYNOPSIS

uuencode [−m] [file] name

uudecode [-o outfile] [file]...

DESCRIPTION

Uuencode and *uudecode* are used to transmit binary files over transmission mediums that do not support other than simple ASCII data.

Uuencode reads *file* (or by default the standard input) and writes an encoded version to the standard output. The encoding uses only printing ASCII characters and includes the mode of the file and the operand *name* for use by *uudecode.* If *name* is */dev/stdout* the result will be written to standard output. By default the standard UU encoding format will be used. If the option *−m* is given on the command line **base64** encoding is used instead.

Uudecode transforms uuencoded *files* (or by default, the standard input) into the original form. The resulting file is named *name* (or *outfile* if the −o option is given) and will have the mode of the original file except that setuid and execute bits are not retained. If *outfile* or *name* is /dev/stdout the result will be written to standard output. *Uudecode* ignores any leading and trailing lines. The program can automatically decide which of the both supported encoding schemes are used.

EXAMPLES

The following example packages up a source tree, compresses it, uuencodes it and mails it to a user on another system. When *uudecode* is run on the target system, the file ‘‘src_tree.tar.Z’’ will be created which may then be uncompressed and extracted into the original tree.

```
tar cf − src_tree | compress | uuencode src_tree.tar.Z | mail sys1!sys2!user
```

SEE ALSO

compress(1), mail(1), uucp(1), uuencode(5)

STANDARDS

This implementation is compliant with P1003.2b/D11.

BUGS

If more than one file is given to *uudecode* and the −o option is given or more than one *name* in the encoded files are the same the result is probably not what is expected.

The encoded form of the file is expanded by 37% for UU encoding and by 35% for base64 encoding (3 bytes become 4 plus control information).

HISTORY

The *uuencode* command appeared in BSD 4.0.

NAME

uudecode - decode a binary file

SYNOPSIS

uudecode [-o *outfile***][***file***]**

DESCRIPTION

The *uudecode* utility shall read a file, or standard input if no file is specified, that includes data created by the *uuencode* utility. The *uudecode* utility shall scan the input file, searching for data compatible with one of the formats specified in *uuencode*, and attempt to create or overwrite the file described by the data (or overridden by the **-o** option). The pathname shall be contained in the data or specified by the **-o** option. The file access permission bits and contents for the file to be produced shall be contained in that data. The mode bits of the created file (other than standard output) shall be set from the file access permission bits contained in the data; that is, other attributes of the mode, including the file mode creation mask (see *umask*()), shall not affect the file being produced.

If the pathname of the file to be produced exists, and the user does not have write permission on that file, *uudecode* shall terminate with an error. If the pathname of the file to be produced exists, and the user has write permission on that file, the existing file shall be overwritten.

If the input data was produced by *uuencode* on a system with a different number of bits per byte than on the target system, the results of *uudecode* are unspecified.

OPTIONS

The *uudecode* utility shall conform to the Base Definitions volume of IEEE Std 1003.1-2001, Section 12.2, Utility Syntax Guidelines.

The following option shall be supported by the implementation:

-o *outfile*
A pathname of a file that shall be used instead of any pathname contained in the input data. Specifying an *outfile* option-argument of **/dev/stdout** shall indicate standard output.

OPERANDS

The following operand shall be supported:

file The pathname of a file containing the output of *uuencode*.

STDIN

See the INPUT FILES section.

INPUT FILES

The input files shall be files containing the output of *uuencode*.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of *uudecode*:

LANG Provide a default value for the internationalization variables that are unset or null. (See the Base Definitions volume of IEEE Std 1003.1-2001, Section 8.2, Internationalization Variables 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 error.

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

ASYNCHRONOUS EVENTS
Default.

STDOUT
If the file data header encoded by *uuencode* is **-** or **/dev/stdout**, or the **-o /dev/stdout** option overrides the file data, the standard output shall be in the same format as the file originally encoded by *uuencode*. Otherwise, the standard output shall not be used.

STDERR
The standard error shall be used only for diagnostic messages.

OUTPUT FILES
The output file shall be in the same format as the file originally encoded by *uuencode*.

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 user who is invoking *uudecode* must have write permission on any file being created.

The output of *uuencode* is essentially an encoded bit stream that is not cognizant of byte boundaries. It is possible that a 9-bit byte target machine can process input from an 8-bit source, if it is aware of the requirement, but the reverse is unlikely to be satisfying. Of course, the only data that is meaningful for such a transfer between architectures is generally character data.

EXAMPLES
None.

RATIONALE
Input files are not necessarily text files, as stated by an early proposal. Although the *uuencode* output is a text file, that output could have been wrapped within another file or mail message that is not a text file.

The **-o** option is not historical practice, but was added at the request of WG15 so that the user could override the target pathname without having to edit the input data itself.

In early drafts, the [**-o** *outfile*] option-argument allowed the use of **-** to mean standard output. The symbol **-** has only been used previously in IEEE Std 1003.1-2001 as a standard input indicator. The developers of the standard did not wish to overload the meaning of **-** in this manner. The **/dev/stdout** concept exists on most modern systems. The **/dev/stdout** syntax does not refer to a new special file. It is just a magic cookie to specify standard output.

FUTURE DIRECTIONS
None.

SEE ALSO
umask() , *uuencode*

COPYRIGHT
Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the

original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

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 *decode_pathname*, for re-creation of the file on another system that conforms to this volume of IEEE Std 1003.1-2001.

OPTIONS

The *uuencode* utility shall conform to the Base Definitions volume of IEEE Std 1003.1-2001, 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 results 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 *uuencode*:

LANG Provide a default value for the internationalization variables that are unset or null. (See the Base Definitions volume of IEEE Std 1003.1-2001, Section 8.2, Internationalization Variables 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 error.

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>s.

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 concatenated 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 uuencode Base64 Values . **Table: 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 *uudecode*).

Special processing shall be performed if fewer than 24 bits are available 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:

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.

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.

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>s.

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 character 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 interchange between systems with possibly different codesets, and to represent 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>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 codeset, 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 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 processed 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 character, **'='** , is used to signify a special processing function.)

This subset has the important property that it is represented identically in all versions of the ISO/IEC 646:1991 standard, including US ASCII, and all characters in the subset are also represented identically 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 historical 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*

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1, 2003 Edition, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 6, Copyright (C) 2001-2003 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

NAME

uuencode − format of an encoded uuencode file

DESCRIPTION

Files output by uuencode(1) consist of a header line, followed by a number of body lines, and a trailer line. The uudecode(1) command will ignore any lines preceding the header or following the trailer. Lines preceding a header must not, of course, look like a header.

The header line is distinguished by having the first 6 characters *begin* The word *begin* is followed by a mode (in octal), and a string which names the remote file. A space separates the three items in the header line.

The body consists of a number of lines, each at most 62 characters long (including the trailing newline). These consist of a character count, followed by encoded characters, followed by a newline. The character count is a single printing character, and represents an integer, the number of bytes the rest of the line represents. Such integers are always in the range from 0 to 63 and can be determined by subtracting the character space (octal 40) from the character.

Groups of 3 bytes are stored in 4 characters, 6 bits per character. All are offset by a space to make the characters printing. The last line may be shorter than the normal 45 bytes. If the size is not a multiple of 3, this fact can be determined by the value of the count on the last line. Extra garbage will be included to make the character count a multiple of 4. The body is terminated by a line with a count of zero. This line consists of one ASCII space.

The trailer line consists of *end* on a line by itself.

SEE ALSO

uuencode(1), uudecode(1), uusend(1), uucp(1), mail(1)

HISTORY

The *uuencode* file format appeared in BSD 4.0 .