NAME

UUDeview − a powerful decoder for binary files

SYNOPSIS

uudeview [options] [@*file*] *file(s)*

DESCRIPTION

UUDeview is a smart decoder for attachments that you have received in encoded form via electronic mail or from the usenet. It is similar to the standard **uudecode**(1) command, yet with more comfort and flexibility. *UUDeview* supports the *uuencoding, xxencoding, Base64, yEncoding* and *BinHex* encoding methods, and is able to handle split-files (which have been sent in multiple parts) as well as multiple files at once, thus greatly simplifying the decoding process. Usually, you will not have to manually edit files to prepare them for decoding.

After invoking **uudeview,** it will scan all given files for encoded data, sort them and their parts and then present you with the list of files that seem like they can be decoded properly. You can then pick files individually for decoding.

OPTIONS

BEHAVIOR

-i Disables interactivity. After scanning the files and sorting everything out, the program will not prompt you for whether a file shall be decoded or not, but batch-decodes all available files. This is the default when reading from standard input.

-a Autorename option. If a target file already exists, and this option is given, a dot and a unique sequence number is appended to the file name. I.e., foo.gif becomes foo.gif.1 if decoded a second time.

+a An alternative incarnation of autorename. If a target file already exists, an underscore and a unique sequence number is inserted into the filename before the first dot, i.e., foo.gif becomes foo_1.gif.

-o Gives the OK to overwrite existing files when decoding. In interactive mode, the default is to prompt the user whether to overwrite, rename or skip the file. This option takes precedence over **-a.** In non-interactive mode (using **-f**), the default is to overwrite files without asking.

+o Says it's not OK to overwrite files. This is useful in non-interactive mode, so that existing files are untouched. This has lesser precedence than -a.

-c Autoclear. Remove all input files that were successfully decoded. Use with care! UUDeview only checks if any data was decoded from an input file, but does not care about any other contents of that input file, or whether a file also held an incomplete attachment.

-p *path* Sets the path where decoded files shall be written to. This must be a valid pathname, or you'll get errors when trying to decode anything. Defaults to the current working directory.

-m Ignore file mode. Uuencoded and xxencoded files have the original file permissions stored on the begin line. Unless this option is given, *UUDeview* will restore them without checking if they are sensible. With this option, the permissions are reset to a default of 0666.

TWEAKING

-z Enforces stricter MIME adherance. Normally, the program tries to find encoded data even in "text/plain" plaintext parts of MIME messages. With this option given, *UUDeview* will limit this capability, and will not accept apparently incomplete encoded messages (for example, seemingly uuencoded data without begin or end lines). You can tighten this option even more by using it twice, or by using **-z2.** Then, *UUDeview* will not check plaintext sections of MIME messages for encoded data at all and behave fully MIME-compliant. Neither option affects the behavior on non-MIME input files. This option needs a better name, but I'm slowly running out of option letters.

-f Uses fast mode for file scanning. The program assumes that each input file holds at most one part, which is usually true for files in a news spool directory. This option **breaks decoding** of input files with multiple articles. Also, certain sanity checks are disabled, probably causing erroneous files to

be presented for decoding. Sometimes you'll get error messages when decoding, sometimes you'll just receive invalid files. Don't use **-f** if you can't live with these problems.

-r Ignore reply messages, i.e. all messages whose subject starts with Re:

-t Use plaintext messages. Usually, UUDeview only presents encoded data for decoding. Plaintext messages are only shown if they have an associated file name. With this option set, unnamed text parts from *MIME* messages and non-encoded messages are also offered. Unnamed messages are assigned a unique name in the form of a sequential four-digit number.

-d Sets the program into desperate mode. It will then offer you to decode incomplete files. This is useful if you are missing the last part of a 50-parts posting, but in most cases the desperately-decoded files will simply be corrupt and unusable. The degree of usefulness of an incomplete file depends on the file type.

-b This changes*UUDe view's* "bracket policy." *UUDeview* looks at a message's subject line, and reads numbers in brackets as the part number, as in (3/7), which is read as the third message in a series of seven. By default, numbers in parentheses () are preferred over numbers in brackets []. You can change this using either **-b** or, for clarity **-b***[]*.

-s Read "minus smartness". This option turns off automatic part number detection from the subject line. Try this option if*UUDe view* fails to parse the subject line correctly and makes errors at guessing part numbers, resulting in incorrect ordering of the parts. With this option, parts are always put together sequentially (so the parts must be correctly ordered in the input file). Also, with this option, the program cannot detect that parts are missing. **Note:** The correct part number found in proper *MIME* files is still evaluated. If this option is given twice, the subject itself is ignored, too, and won't be used to group parts. Use if the messages that the parts come delivered in have different subject lines.

OTHER OPTIONS

-q (Quiet) Disables verbosity. Normally, the program prints some status messages while reading the input files, which can be very helpful if something should go wrong. Use if these messages disturb you. Disables progress bars. See**-n** option.

-v (disables Verbosity) Disables verbose messages, i.e. notes are not displayed, but does not remove warnings and errors. Is not as quiet as the **-q** (Quiet) option.

-n No progress bars. Normally, UUDeview prints ASCII bars crawling up to 100 percent, but does not check if your terminal is capable of displaying them. Use this switch if your terminal isn't, or if you find the bars annoying.

+e *exts* Selects only the files with the given extensions for decoding, others will be ignored. **+e** *.gif .jpg* would decode all gif and jpeg files, but not tif or other files. The list of extensions works case-insensitive.

-e *exts* The reverse of the above.

You will experience unwanted results if you try to mix +e and −e options on the command line.

INPUT OPTIONS

file(s) The files to be scanned for encoded files. You can also give a single hyphen ´−´ to read from standard input. Any number of files may be given, but there is usually a limitation of 128 options imposed by the shell. If you are composing the list of files with wildcards, make sure you don't accidentally feed the program with binary files. This will result in undefined behaviour.

@ *file* Makes *UUDeview* read further options from the file. Each line of the file must hold exactly one option. The file **is erased** after the program finishes. This feature may be used to specify an unlimited number of files to be scanned. Combined with the powers of **find**(1), entire directory trees (like the news spool directory) can be processed.

Options may also be set in the $UUDEVIEW environment variable, which is read before processing the options on the command line.

DECODING

After all input files have been scanned, you are asked for each file what do do with it. Of course, the usual answer is to decode it, but there are other possibilities. You can use the following commands (each command is a single letter):

d (D)ecode the file and write the decoded file to disk, with the given name.

y (Y)es does the same as (d).

x E(x)tract also decodes the file.

a Decodes all remaining files without prompting.

n Skips this file without decoding it.

b Steps back to the previous file.

r Rename. You can choose a different name for the file in order to save it under this new name.

p Set the path where decoded files shall be written to. This path can also be set with the -p command line option.

i Displays info about the file, if present. If a multipart posting had a zeroeth part, it is printed, otherwise the first part up to the encoded data is printed.

e Execute a command. You can enter any arbitrary command, possibly using the current file as an argument. All dollar signs '$' in this command line are replaced with the filename of the current file (speaking correctly, the name of a temporary file). You should not background processes using this temporary file, as programs might get confused if their input file suddenly disappears.

l List a file. Use this command only if you know that the file in question is a textfile, otherwise, you'll get a load of junk.

q Quits the program immediately.

? Prints a short description of all these commands.

If you don't enter a command and simply hit return at the prompt, the default command, decoding the file, is used.

RUNTIME MESSGAGES

In verbose mode (that is, if you didn't disable verbosity with the -v option), progress messages will appear. They are extremely helpful in tracing what the program does, and can be used to figure out the reason why files cannot be decoded, if you understand them. This section explains how to interpret them. Understanding this section is not essential to operate the program.

First, there are "Loading" messages, which begin with the string "Loaded". Each line should feature the following items:

Source File
The first item is the source file from which a part was loaded. Many parts can be detected within a single file.

Subject Line
The complete subject is reproduced in single quotes.

Identifier
The program derives a unique identification for this thread from the subject line, for grouping articles that look like they belong to the same file. The result of this algorithm is presented in braces.

Filename
If a filename was detected on the subject line or within the data (for example, on a begin line, or as part of the Content-Type information).

Part Number
The part number derived from the subject line, or, in the case of properly MIME-formatted messages, from the "part" information.

Begin/End
If a "begin" or "end" token was detected, it is printed here.

Encoding Type
If encoded data was detected within this part, either "UUdata", "Base64", "XXdata" or "Binhex" is printed here.

More messages are printed after scanning has completed. A single line will be printed for each group of articles. The contents of this line are best understood by looking at an example. Here is one:

Found ’mailfile.gz’ State 16 UUData Parts begin 1 2 3 4 5 end 6 OK

This indicates that the file *mailfile.gz* has been found. The file was uuencoded ("UUData") and consists of 6 parts. The "begin" token was found in the first part, and the "end" token was found in the sixth part. Because it looks like everything’s there, this file is tagged as being "OK". The *State* is a set of bits, where the following values may be or’ed:

1 Missing Part

2 No Begin

4 No End

8 No encoded data found.

16 File looks Ok

32 An error occurred during decoding of the file.

64 File was successfully decoded.

NOTES

Because the program cannot receive terminal input when a file is being read from standard input, interactivity is automatically disabled in this case.

UUDeview is aware of MIME messages, but normally ignores strict MIME compliance in favor of finding unproperly encoded data within them, e.g. to succeed when individual parts of a uuencoded file have been sent with a MIME mailer as MIME messages. For that, it subjects all "text/plain" parts of a message to encoding detection. You can use the **-z** option (see above) for more strict RFC2045 compliance.

The scanner tends to ignore short Base64 data (less than four lines) outside of MIME messages. Some checks for this condition are used in desperate mode, but they may cause misdetection of encoded data, resulting in some invalid files.

Files are always decoded into a temporary file first, then this file is copied to the final location. This is to prevent accidentally overwriting existing files with data that turns out too late to be undecodeable. Thus be careful to have twice the necessary space available. Also, when reading from standard input, all the data is dumped to a temporary file before starting the usual scanning process on that file.

uudeview tries to derive all necessary information from the Subject: line if present. If it holds garbage, or if the program fails to find a unique identification and the part number there, **uudeview** might still be able to decode the file using other heuristics, but you’ll need major luck then.
Yet this is only a concern with split-files. If all encoded files only consist of single parts, don’t worry.

If you rename, copy or link the program to **uudecode**, it may act as a smart replacement for the standard, accepting the same command-line options. This has not been well-tested yet.

SEE ALSO

uuenview(1), **uudecode**(1), **uuencode**(1).
The *UUDeview* homepage on the Web,
http://www.fpx.de/fp/Software/UUDeview/

BUGS

To read a file whose name starts with a hyphen ’-’, prepend a path name, for example ’./’.

The checksums found in *BinHex* data are ignored.

The program cannot fully handle partial multipart messages (MIME-style multipart messages split over several mail messages). The individual parts are recognized and concatenated, and the embedded multipart message is "decoded" into a plain-text file, which must then be fed again to **uudeview.** Don’t worry, these kinds of messages are rare.

UUDeview cannot decipher RFC 1522 headers.