MFUUD version 3.6.7
Multiple File UUDecoder

WHAT IS MFUUD?

MFUUD (Multiple File UUdecoder) is a program designed to restore uuencoded binary files to their original format. Although the standard uudecode program will do this, due to restrictions on message size, many uuencoded binary files must be segmented before being posted to internet bulletin boards. This greatly increases the complexity of the decoding process, which in turn, increases the amount of human intervention required to successfully decode binary posts. The design concept behind MFUUD is to remove all human intervention and restore simplicity to the decoding process.

MFUUD is a breed of smart uudecoding software that have now become widely available but incorporates many of the features found in other decoding packages into a small, simple, portable C program. MFUUD is almost as easy to use as uudecode but is vastly more powerful.

IMPORTANT BACKGROUND INFORMATION

MFUUD, unlike uudecode, is capable of processing uuencoded file segments. That is to say that MFUUD can decode uuencoded binary information that span multiple files. These segments may be parts of one or more complete uuencoded files. Certain information must be present in each file segment that enables the decoder to restore the contents of the original binary file. Regardless of the format, this information must provide a clue as to which uuencoded file each segment belongs to and what order the file segments must be decoded. Since the only reason to segment uuencoded files is to facilitate proper transmission of binary information through the internet mail system, the information leading to the proper grouping of the file segments is usually included in the subject line or caption of the messages containing the files segments. For example, the captions for messages containing binary information, in this case image data, might read as follows:

```
Subject: alison03.jpg (Part 0/3) Alison Armitage
Subject: alison03.jpg (Part 1/3) Alison Armitage
Subject: alison03.jpg (Part 2/3) Alison Armitage
Subject: alison03.jpg (Part 3/3) Alison Armitage
Subject: Naomitrip.jpg (1/6)
```

```
Subject: Naomitrip.jpg (2/6)
Subject: Naomitrip (3/6)
Subject: Naomitrip.jpg (4/6)
Subject: Naomitrip.jpg (5/6)
Subject: Naomitrip.jpg (6/6)
```

MFUUD uses the information present in the subject line or caption; it looks for the filename and the part numbers. Due to the fact that most of the captions are typed in, it is not uncommon that errors may occur that may throw off a very strict decoding program. In addition, not all posters use the same format when typing in the caption. MFUUD, however, is capable of correcting for most errors and can read a variety of different subject line formats. MFUUD also uses other information present in the message to aid in the reconstruction of the original binary files and, in most cases, can decode file segments as well as a very careful and patient human, only many times faster.

HOW TO USE MFUUD

MFUUD is simple to use. The syntax is as follows:

```
mfuud [-del] [-quiet] [-dest directory name] [-mode nnn]
     [-plain] [-keywords "keywords"] [-ext] [-err filename]
     file#1...file#n
                          or
                 mfuud -log directory
                          or
                mfuud -unlog directory
```

The text in brackets represent command line options. (When typing the options, do not include the brackets.) The options must be typed in lowercase and can be abbreviated so long as there is no ambiguity. Some of the options require arguments: -temp, -dest, -mode, -keywords, and -err. The proper arguments must follow each of these options if they are used, and the option and argument must be separated by a space character.

If you do not specify any filenames, then MFUUD will decode from the standard input. If you do specify filenames, you may use the wildcard (*) character if you like. MFUUD will be expecting message files (i.e. text files that contain message headers). If MFUUD can't find message headers then the program will assume you are decoding plain uuencoded files and will behave like uudecode. The order of the files you specify does not matter as long as the message headers are intact since MFUUD is "smart" enough to combine files in the proper order and rebuild the original uuencoded file. You can also specify more than one group of files.

MFUUD will combine all related parts in the proper order to reconstruct each uuencoded file. MFUUD uses advanced pattern matching and sorting techniques to ensure proper decoding of files; however, the program is not perfect and can fail due to errors in the uuencoded files.

COMMAND LINE OPTIONS

This sections describes in detail the functionality of each MFUUD option. The unabbreviated and abbreviated option precedes each description.

OPTION: -delete, -del

This option instructs MFUUD to delete the uuencoded file segments specified on the command after they have been decoded. Only the input files that were successfully decoded will be deleted. All non-uuencoded input files will also be ignored.

OPTION: -quiet, -q

This option prevents MFUUD from displaying any output. Error messages, however, will still be sent to the standard error device (stderr).

OPTION: -extended, -ext

This option activates MFUUD's extended search function that finds uuencoded file segments that are stored within a single file. It is especially useful if you want to decode a file that contains all the message files concatenated together. Many newsreaders will append message files you select to a specified file. In order to decode that type of file, you must use the following syntax.

```
mfuud -ext filename
```

OPTION: -destination *directory*, -dest *directory*

This option is used to specify a destination directory for the decoded files. By default, all uudecoded files are written into the current directory.

OPTION: -mode *nnn*, -m *nnn*

This option will override the default file permission of each decoded file. This default file information is contained in the uuencoded file and is used to set its file permission when created. A three digit octal number must follow the **-mode** option. Each file created by MFUUD will be set to this specified file mode. Consult your operating system manual for more information on file modes.

OPTION: -plain, -p

Use of this option will force MFUUD to ignore any message headers encountered, and MFUUD will decode file segments in the order they are specified on the command line. MFUUD will behave like uudecode, except you can specify one or more uuencoded file segments.

```
mfuud -p part1 part2 part3
```

In the above example, part1, part2, and part3 are parts of a uuencoded file. In addition, MFUUD will remove extraneous text while it decodes.

```
mfuud -p Cindy1of1 Steph1of2 Steph2of2
```

The above example could produce two files named "cindy.jpg" and "steph.jpg". If you specify uuencoded parts, you must ensure that they are specified in order. There is no restriction on how many files you can specify.

OPTION: -directory, -dir

The **-dir** option instructs MFUUD to honor the pathnames found on the begin lines. For example:

```
begin 644 \users\usr5\pe0e\pictures.jpg
```

By default, MFUUD ignores pathnames and only looks at the basename. Only in special circumstances will the poster's tree structure agree with yours. Use this option to override the default.

OPTIONS: -keywords "*key1 key2 ... keyn*", -k "*key1 key2 ... keyn*"

This option is useful when you are performing "mass decoding". See the section on mass decoding for more information. This option lets MFUUD know which files to decode. The keywords you specify must be in double quotes, and each must begin with either a "+" or a "-". The "-" preceding a keyword, tells MFUUD to ignore messages whose subject line contains this keyword. Any keyword preceded by a "+" are decoded regardless. For example:

```
mfuud -keywords "-male +female" *.*
```

Specifying this command line will decode all message files in the current directory that don't contain the word "male" in the captions unless they contain "female", in which case it makes no difference as MFUUD will decode it regardless. MFUUD, when scanning for keywords, is not case sensitive so it make no difference if the keywords are uppercase or lowercase. You should note that the following syntax will get you undesired results:

```
mfuud -keywords "-male" *.*
```

The keyword "female" also contains the word "male"; therefore, mfuud will not decode message files that contain the word "female" in the caption. You can instruct mfuud to be more strict in matching keywords by using the "~" character. If you precede and end the keyword with a tilde, then MFUUD will match complete words. For example:

```
mfuud -keywords "-~male~" *.*
```

If you are searching for specific files then you can use a special notation. If you specify a "-" followed by no keyword than MFUUD will ignore all messages. This is not useful unless you specify at lease one keyword preceded by a "+". For example,

```
mfuud -keywords "- +cindy +crawford" *.*
```

This instructs MFUUD to ignore all messages except those that contain "cindy" or "crawford" in their caption.

OPTION: -error *filename*, -err *filename*

The **-error** option tells MFUUD to append messages it cannot decode due to an error or errors to the filename specified. The most typical error encountered occurs when MFUUD is unable to find all the parts of a uuencoded file. All uuencoded messages not decoded are stored in the specified error file. This file can later be decoded with the **-ext** option.

```
mfuud -ext errorfile newfile
```

In the above example, "errorfile" contains the undecoded messages and "newfile" contains the parts that were missing. Also note that if the **-del** option is used with the error option then the input files not decoded will also be deleted, but a copy of them will be stored in the specified error file.

Non-uuencoded file segments will be ignored as usual.

OPTION: -log *directory*, -l *directory*

This option enters the directory name into the log file specified in the enviroment variable MFUUDLOG. The directory name must be the full path including the drive letter. All input files later decoded from this directory AND its subdirectories will be logged. That is to say that MFUUD will remember the names of these files and will avoid decoding them if MFUUD encounters them again. See the section on **REMEMBERING DECODED FILES** for more information.

OPTION: -unlog *directory*, -l *directory*

This option undoes the effect of the **-log** option. The directory name entry specified and all of its subdirectories will be removed from the log file.

DECODING FROM STANDARD INPUT

If you don't specify filenames then MFUUD will assume you want to decode from the standard input. You can use this feature to pipe uuencoded files for MFUUD to decode; however, when decoding from standard input, MFUUD requires that the files segments it encounters be in the proper order. In addition, MFUUD ignores extraneous text.

MASS DECODING

Mass decoding, as it pertains to MFUUD, is the process of uudecoding message files by skipping the intermediate steps. If you know the pathname to the directory containing the messages you are interested in, then you can instruct MFUUD to decode files straight out of this directory. I recommend that you create a temporary directory where you would store the decoded files, especially if you are planning to decode many files.

```
cd path_to_message_directory
mfuud -dest \usr\tmp\pictures *.*
```

The temporary directory in this case is \usr\tmp\pictures. You could also be in \usr\tmp\pictures and specify the path of the directory you want to decode from followed by a wildcard. This may or may not work. It depends on the length of the path you specify, but in many cases you will overflow the command line.

REMEMBERING DECODED FILES

It is possible when performing mass decoding more than once on the same directory that files that were already decoded will be decoded again. To prevent the situation of having duplicate files, MFUUD can store the names of the input files in a log file you specify. The log file is specified in an environment variable called **MFUUDLOG**. For example:

```
set MFUUDLOG=C:\logfile
```

If while in the process of scanning the input files, MFUUD encounters files that are listed in the log file, MFUUD will skip these files. If the file is not listed in the log file, MFUUD will add the input file to the log ONLY if it was successfully decoded. It is important to note that MFUUD also purges files from the log. This process is invisible to the user but it is important to understand how this process works. Each input file stored in the log file is attached to its corresponding directory that was specified with the **-log** option. When MFUUD logs input files, it assumes that the input files specified on the command line represent the contents of the directory that you are decoding from. All files in the log for that directory that aren't found on the command line are erased from the log. For example, suppose you are logging the current directory and you enter the command:

```
mfuud -dest \usr\tmp\pictures *.*
```

MFUUD will preceed to decode all files in the current directory and log all successfully decoded input files. If several days later you type the same command, some of the older files in that directory may have been deleted. MFUUD takes the contents of the command line and compares it with the log file. All files in the log for the directory in question that are not present on the command line are deleted. That is why it is important to be consistent with your command line options. For instance, if you enter the command:

```
mfuud -dest \usr\tmp\pictures 11876
```

MFUUD assumes that file "11876" is the only content of that directory; therefore, MFUUD will purge all files in the log under that directory except for 11876.