# SPRY UUCode Help Contents 

The following Help Topics are available:

Introduction
Program Operation
Procedures
Keyboard
System Requirements and Usage
General Operating Information
Encoding Header Formats

UUCode Menu Commands

For Help using Help, press [F1]. Help revised May, 95.

# Introduction

When to Use UUCode
UU and XX Encoding & Decoding

## When to Use UUCode

Whenever you need to send binary data over communication links or networks, such as the Internet, which cannot handle 8-bit data, then UUCode provides you a solution.  Binary data is used in file types with extensions of .EXE, .COM, ZIP, .ARC, .LZH, .ZOO, and many others.

UUCode will also allow sending of binary data files between popular on-line services such as AT&T Mail, America On-Line, CompuServe, Delphi, GEnie, MCI mail, and educational and other sites on the Internet.

Another popular use for UUCode is on Usenet, the "Bulletin Board System (BBS) of the Internet." Again, the reason is to allow binary data to be posted as part of news articles.

# UU and XX Encoding & Decoding

UUCode uses a method called UU or XX encoding and decoding to convert the 8-bit binary data into a form usable by communications links or networks which cannot transport such data. This method generates (encodes) four ASCII characters from every three binary characters, which is basically converting 6 bits of binary information into the appropriate 6 bits within a 7-bit ASCII character. This guarantees that only ASCII text is sent when the encoded output is sent over computer communication links or networks. Similarly, the decoding process converts four ASCII characters into three binary characters. The penalty of this approach is that the encoded file is 33% larger than the original file, but that's better than not being able to transfer binary data at all!

XX encoding is similar, but uses a different character set than UU encoding so that character set translations will work better across multiple types of systems, i.e. between EBCDIC (IBM mainframe) and ASCII on UNIX.

Now you can handle these encoded files on a Windows 3.0/3.1 equipped PC such that UU or XX encoding and decoding is possible between two PCs or between your PC and any other computer system, such as a UNIX system or Macintosh.  UUCode can do both encoding and decoding, and is fully compatible with the standard "uuencode" and "uudecode" programs found on UNIX or other computer systems.

# Optional File Type Association

If you want to run UUCode from FILE MANAGER and take advantage of the **Command Line Mode**, you can cause UUCode to automatically be invoked when you want to decode a file. Files to be decoded are identified by either a ".UUE" or ".XXE" extension.

Encoding a file is also possible if you want to associate other file types having extensions different than ".UUE" or ".XXE". Usually this is not the case because you may want to run another program on such files, such as a word processor or archive program. The better way to do this is to use **Drag-and-Drop Mode**. If you do want to associate files to encode with UUCode from FILE MANAGER, then follow steps 3 and 4 below using the appropriate file extension.

If you are running Windows 3.1, you need do nothing extra to take advantage of **Drag-and-Drop Mode**.

To associate files for decoding with UUCode, follow the steps shown below.

Step 1. Run the FILE MANAGER.

Step 2. FILE MANAGER requires a file with the appropriate extension to exist so that associations can be made. Copy your AUTOEXEC.BAT or any other file to a file called Z.UUE.

Step 3. Highlight the Z.UUE file. (do not double click or press Enter to run it).

Step 4. From the top menu choose, **File|Associate**. When the Associate dialog box pops up, type the full path name of where UUCODE.EXE is located. Then press Enter or click OK. This completes the association. You may now delete the Z.UUE file.

Step 5. Repeat steps 2 through 4 with a file called Z.XXE.

FILE MANAGER will now automatically run the decoding or encoding portion of UUCode when the files with that extension are double-clicked from FILE MANAGER.

# Program Operation

UUCode has three modes of operation; interactive mode, command line mode, and drag-and-drop mode. **Interactive Mode** is automatically used when a file argument *is not specified* on the command line when starting UUCode. **Command Line Mode** is automatically used whenever any file argument *is specified* on the command line. **Drag-and-Drop Mode**, not available with Windows 3.0, is used when files are "dropped" on UUCode from FILE MANAGER. Since each mode operates differently, each mode is described separately.

Interactive Mode

Command Line Mode

Drag-and-Drop Mode

# Interactive Mode

All operation of UUCode in interactive mode is done via the menu or associated hot keys.

Interactive mode is most useful when several files will be encoded or decoded. In addition, while in interactive mode, unlike **Command Line Mode**, UUCode will prompt the user before the following actions are done:

A. Prompt for a file to be encoded or decoded.

B. After choosing the encoding or decoding operation, a suggested file name for output is shown. The user is allowed to modify the output file name, unless **Use Default File Names** is enabled.

C. If a file to be output already exists, UUCode will ask permission before overwriting it, unless **Overwrite Existing Files without Asking** is enabled.

# Command Line Mode

All operation of UUCode in command line mode is done via the **File|Run** command of either PROGRAM MANAGER or FILE MANAGER or when you double-click a file in FILE MANAGER and you have previously set up your **Optional File Type Association**. Command line mode is most useful when one file will be encoded or decoded. Unlike **Interactive Mode**, UUCode will never prompt the user before performing an action.

UUCode will allow multiple files on the command line. Each file name must be separated by one or more spaces from the other names on the command line.

Command Line Arguments

Operation Based on Input File Name

# Command Line Arguments

UUCode supports one command line argument: **/J** or **-J**, which permits the user to **Decode Multiple Input Files as 1**. When this option is used, all input files specified on the command line are treated as one large input file.

## Operation Based on Input File Name

In command line mode, the choice of performing a encode or decode operation can be determined by the extension of the file name supplied on the command line, if the **Decode Method** is optioned to do so. Files with an extension of "`.UUE`" will be run through the UU decoding process. Files with an extension of "`.XXE`" will be run through the XX decoding process. Files with any extension *other than* "`.UUE`" or "`.XXE`" will be run through the encoding process.

You can also force UU or XX decoding as described in **Decode Method**.

# Drag-and-Drop Mode

Drag-and-Drop mode can be started in the same way as **Interactive Mode**; by first running UUCode without any file argument. Drag-and-Drop mode is then entered when files are dropped on UUCode via FILE MANAGER. Drag-and-Drop mode can also be started by dropping files on the UUCODE.EXE file in a FILE MANAGER window when UUCode is not yet running. In that case, FILE MANAGER will automatically start running UUCode and drop the selected files on it.

UUCode supports dropping of one or many files. If multiple files are dropped, they will be operated on one at a time until all files have been processed. Each file is treated independently, unless **Decode Multiple Input Files as 1** is enabled. Therefore, you may encode or decode files as determined by their file name. To determine how the dropped file name affects the operation selected by UUCode, see the **Operation Based on Input File Name** section.

Drag-and-Drop mode operates in the same way as **Interactive Mode**. Program operation and errors are reported as they occur and user confirmation of activity may be required, depending on the configuration settings.

## Procedures

Encoding a File
Decoding a File

# Encoding a File

Encoding a file allows you to convert a file containing 8-bit binary data into a 7-bit ASCII encoded readable form. The following steps apply to **Interactive Mode** or **Drag-and-Drop Mode**. In **Command Line Mode**, files with extensions other than "`.UUE`" or "`.XXE`" are automatically encoded without additional user prompting.

Entry. In **Interactive Mode**, selecting the **File|Encode** menu item enters the encoding process.

Step 1. A dialog box will appear showing all files in the directory.

Step 2. You will first be prompted to select a file to encode. Pick the file name and then click the OK button or press Enter. An output file name will be suggested with the default file extension of "`.UUE`" for UU encoding or "`.XXE`" for XX encoding. You may change it if you wish, unless **Use Default File Names** is enabled.

Step 3. The encoding process is started by clicking the "OK" button (or pressing Enter). If the output file already exists, UUCode will ask if you want to overwrite it, unless **Overwrite Existing Files without Asking** is enabled. You will notice a % complete bar being updated as UUCode encodes the input file.

Step 4. At any time during the encoding process, you may click the "Cancel" button with the mouse or ALT-C and be returned to the main menu, ready for another command. If you cancel the encoding operation, no encoded output file will be generated. When encoding is complete, you will see a dialog box indicating success (or failure).

# Decoding a File

Decoding a file allows you to convert a file containing a 7-bit ASCII encoded data into its original 8-bit binary data form. The following steps apply to **Interactive Mode** or **Drag-and-Drop Mode**. In **Command Line Mode**, files with the extension of "`.UUE`" or "`.XXE`" are automatically decoded without additional user prompting.

Entry. In **Interactive Mode**, selecting the **File|Decode** menu item enters the decoding process.

Step 1. A dialog box will appear showing all "`.UUE`" and "`.XXE`" files in the directory.

Step 2. You will first be prompted to select a file to decode, using the default extensions of "`.UUE`" and "`.XXE`". Pick the file name and then click the OK button or hit Enter. UUCode will read the encoded file to determine the name of the output file. You will be prompted with the output file name, which you can change if you wish, unless **Use Default File Names** is enabled.

Step 3. The decoding process is started by clicking the "OK" button. UUCode will check if the output file already exists. If it does, you will be asked if you want to overwrite it, unless **Overwrite Existing Files without Asking** is enabled. You will then notice a % complete bar being updated as UUCode decodes the input file.

Step 4. At any time during this process, you may click the "Cancel" button using the mouse or ALT-C and be returned to the main menu, ready for another command. If you cancel the decoding operation, no decoded output file will be generated. When decoding is complete, you will see a dialog box indicating success (or failure).

What if Decoding Fails?

## What if Decoding Fails?

When UUCode cannot complete decoding successfully due to its inability to properly determine a multipart format, it will display a dialog box explaining the problem. The user may then retry decoding with either a specific format or try to decode using all formats supported. If decoding still fails after trying all supported formats, see **What if the Multiple Part UU Format I Use Isn't Supported?** for suggestions in dealing with this situation.

# Menu Commands

The following topics explain the menu system of UUCode.

File Menu

Configure Menu

## File Menu

The file menu allow you to encode or decode files, or exit the program.

Encode Menu Item

Decode Menu Item

Exit Menu Item

# Encode Menu Item

Selecting **Encode** will cause UUCode to begin the process of encoding a binary file (any file type actually) into a encoded ASCII file.  See the **Encoding a File** topic for details.

# Decode Menu Item

Selecting **Decode** will cause UUCode to begin the process of decoding a previously encoded ASCII file into its binary (original) form.  See the **Decoding a File** topic for details.

# Exit Menu Item

Selecting **Exit** will quit the UUCode program.

If UUCode is running while when exit is pressed, UUCode will first cancel the operation (as if the Cancel button were pressed) and clean up any temporary files used during the encoding or decoding process.  Therefore, it may take a few seconds before UUCode completely exits.

# Configure Menu

You can configure UUCode various ways to operate according to your preferences.  There are configuration options for general file handling and window display, encoding, and decoding operations.

General Configuration

Encoding Configuration

Decoding Configuration

# General Configuration

This dialog allows you to configure general UUCode options.

In this configuration dialog box, there are both "Configure" and "Save" buttons. If you choose "Configure," all configuration options chosen will be used for the remainder of the time UUCode is operation (unless you re-configure UUCode again). The next time UUCode is run, the configuration options will be set to the previous defaults. If you choose "Save," then all configuration options will be put into effect and saved as the defaults which will be active the next time UUCode is run. These defaults are placed in a local file called `UUCODE.INI`.

Become ICON When En/Decoding

Stay ICON When En/Decoding

Exit When Done En/Decoding

Overwrite Existing Files without Asking

Use Default File Names

No Status Messages

## Become ICON When En/Decoding

Setting this option applies only when UUCode is run in **Interactive Mode**.

Enable this option if you want UUCode to run as an ICON while encoding or decoding files. UUCode will automatically restore to its previous size when the operation is completed.

## Stay ICON When En/Decoding

Setting this option applies only when UUCode is run in **Interactive Mode**.

Enable this option if you want UUCode to remain an ICON after encoding or decoding files.

The default is that UUCode will automatically restore to its previous size when the operation is completed.

# Exit When Done En/Decoding

Setting this option applies only when UUCode is run in **Interactive Mode**. This option is automatically enabled when UUCode is run in **Command Line Mode**.

Enable this option if you want UUCode to automatically exit after encoding or decoding files.

The default is that UUCode will remain an active application after the operation is completed.

# Overwrite Existing Files without Asking

Setting this option applies only when UUCode is run in **Interactive Mode** or **Drag-and-Drop Mode**. This option is automatically enabled when **Command Line Mode** is used or when **Multi Part Encoding** is selected.

Enable this option if you want UUCode to automatically overwrite any existing output files.

In **Interactive Mode**, the default is to ask user permission before any previously existing output files are overwritten.

# Use Default File Names

Setting this option applies only when UUCode is run in **Interactive Mode** or **Drag-and-Drop Mode**. This option is automatically enabled when **Command Line Mode** is used or when **Multi Part Encoding** is selected.

Enable this option if you want UUCode to automatically choose the output file name for you. The output file name is either the result of encoding a file or the file name as specified in the encoded file when decoding is used.

For encoding and a single file output, the default file name is the input file name with the extension replaced with ".UUE" for UU encoding or ".XXE" for XX encoding. For example, the default file name for encoding ANYFILE.EXE is ANYFILE.UUE.

For **Multi Part Encoding** output, the output name of each file part is determined automatically. The output file names consist of the first six (6) characters of the input file name or the entire input file name, whichever length is less, with a 1 or 2 digit part number (1 to 99) concatenated, and having an extension of ".UUE" or ".XXE". For example, if an encoded output file required four (4) parts and the input file name was MYARCHIV.ZIP, the output parts would be named MYARCH1.XXE, MYARCH2.XXE, MYARCH3.XXE, and MYARCH4.XXE. Similarly, if a file to be encoded was named ABC.EXE, then the four output parts would be ABC1.UUE, ABC2.UUE, ABC3.UUE, and ABC4.UUE.

For decoding, the default file name is that which is specified in the encoded input file. For example, if the file being decoded is named FILE.UUE and contains the file ABC.ZIP, then the default file name is ABC.ZIP.

The default is to ask user permission to use the proposed output file name. This allows the user to manually change the proposed name if desired.

# No Status Messages

Setting this option applies only when UUCode is run in **Interactive Mode** or **Drag-and-Drop Mode**. This option is automatically enabled when UUCode is run in **Command Line Mode**.

Enable this option if you want UUCode to run without requiring user confirmation of success or failure of operations when they are completed.

# Encoding Configuration

This dialog allows you to configure options for UUencoding.

In this configuration dialog box, there are both "Configure" and "Save" buttons. If you choose "Configure," all configuration options chosen will be used for the remainder of the time UUCode is operation (unless you re-configure UUCode again). The next time UUCode is run, the configuration options will be set to the previous defaults. If you choose "Save," then all configuration options will be put into effect and saved as the defaults which will be active the next time UUCode is run. These defaults are placed in a local file called `UUCODE.INI`.

Multi Part Encoding

Encode Method

End of Line String

Max Encoded File Size

Generate Checksums on Encoded Output

# Multi Part Encoding

UUCode supports 12 formats for single or multi part encoding. These are:

- Single file
- SIMTEL multiple part format
- Comp.binaries multiple part format
- Four of the most common multiple part formats used in Alt.binaries Usenet newsgroups
- UNIX shell archive multiple part format
- DOS UUENCODE/UUDECODE multiple part format
- X-File header style format
- UUXFER multiple part format
- WINCODE multiple part format

Select the format required from the list shown.

All but the single file format are considered multiple part formats.

## Encode Method

UUCode supports encoding using either the UU or XX character set. XX encoding uses a different character set which allows for EBCDIC to ASCII conversions. The most popular method, however, is UU encoding. Select UU for UU encoding or XX for XX encoding.

## End of Line String

Since UUCode encoded output may be transferred to either PC or UNIX systems, an option is provided to allow an end-of-line (EOL) terminator string specific to the destination computer type. DOS uses a carriage return (CR) followed by a line feed (LF), while UNIX uses only a LF. Select DOS/Windows for the CR/LF end-of-line or UNIX for LF end-of-line terminator.

# Max Encoded File Size

When encoding using any of the multiple part formats, the maximum output file size parameter limits the maximum number of characters per part to the value specified. The range for this size is 4500 to 1,048,576 (1M) bytes. The default is 60,000 bytes, which is somewhat less than a typical e-mail program limit of 64K (65,536) bytes. Note that the output file(s) may be slightly larger or smaller that this number due to the variable header and trailer data required for the multiple part formats.

The number of output parts is automatically calculated from the input file size and the maximum output file size. There is a ninety-nine (99) part maximum.

When the single file option is chosen, the encoded output file is in one piece, regardless of size. Therefore the maximum output file size field has no effect with this option (and is grayed to so indicate).

See the **Use Default File Names** section for information on how the output files are named.

# Generate Checksums on Encoded Output

To assist in detecting errors introduced by the transport network carrying UU or XX encoded files, a checksum can be placed in each encoded output file. Decoding programs capable of detecting these checksums, such as UUCode, can indicate whether the encoded input appears corrupted.

UUCode uses the same checksum algorithm as the UNIX "sum -r" command. Each part, whether single or multiple parts are output, has appended at the end of the part a checksum line containing the checksum data for that part as well as it's encoded size in bytes. Additionally, the last part also has appended checksum data for the original input file and input file size so that the decoded output can be checked against the original input file that was encoded.

Enable this option to automatically generate the appropriate checksum data. Please note that the UNIX shell archive format does not support the checksum data option.

# Decoding Configuration

This dialog allows you to configure UUCode options for uudecoding.

In this configuration dialog box, there are both "Configure" and "Save" buttons. If you choose "Configure," all configuration options chosen will be used for the remainder of the time UUCode is operation (unless you re-configure UUCode again). The next time UUCode is run, the configuration options will be set to the previous defaults. If you choose "Save," then all configuration options will be put into effect and saved as the defaults which will be active the next time UUCode is run. These defaults are placed in a local file called `UUCODE.INI`.

Automatic Detection of Decoding Type

Decoding Type

Decode Method

Decode Multiple Input Files as 1

Test Checksum Data if Present

## Automatic Detection of Decoding Type

When decoding, the default is to automatically detect the file format used and decode as required. Due to the prolific number of formats, for which there is unfortunately no standard - defacto or otherwise, it is possible that UUCode can be confused by an unsupported format. To assist in decoding, a specific Decoding Type format can be requested.

## Decoding Type

When **Automatic Detection of Decoding Type** is disabled, the specific format to look for is chosen via a list of radio buttons.

# Decode Method

UUCode supports decoding using either the UU or XX character set. Normally, UUCode determines UU or XX decoding **Operation Based on Input File Name**. If a specific decoding method is required, select UU for UU encoding or XX for XX encoding. This will force the particular decoding method regardless of the input file name.

# Decode Multiple Input Files as 1

Normally, all parts of an encoded file are present in a single file to decode. If the encoded input is spread across multiple files, enabling this option will cause all input files to treated as one large input file which UUCode can then decode. This option allows decoding of a single input file split across any number of encoded files with the parts in ***any order***. This option is also available from the command line as explained in **Command Line Arguments**.

## Test Checksum Data if Present

UUCode will automatically determine if any file checksum data is present to help detect possible errors introduced during transport across a network. If such checksum data is present, UUCode will test that data against it's own calculation and report if any mismatches occur. UUCode does not abort decoding if mismatches are found, but rather warns that a potential problem with the decoded file exists.

# Keyboard

The following hot keys are defined for UUCode.

| Hot Key | Command/Procedure |
|---|---|
| F1 | Access on-line help file |
| Shift-F1 | Display the About dialog box |
| F2 | Encoding a File |
| F3 | Decoding a File |
| F5 | General Configuration of UUCode for its file handling and window display characteristics of operation |
| F6 | Encoding Configuration of UUCode for the UU or XX encode operation |
| F7 | Decoding Configuration of UUCode for the UU or XX decode operation |
| F10 | Exit UUCode |

## System Requirements and Usage

CPU Instruction Set

Windows Operating Modes

Disk Usage

Memory Usage

## CPU Instruction Set

UUCode was written for Windows 3.0/3.1 equipped PCs using the 80286 instruction set and will only work on PCs using 80286, 80386, 80386SX, 80486, 80486DX, 80486DX2, 80486SX, and Pentium or compatible processors.

## Windows Operating Modes

UUCode runs only in Windows standard or enhanced modes; *not* in real mode.  This is the same as Windows 3.1.

## Disk Usage

UUCode uses approximately 250K of disk space if you are running Windows 3.1 and an additional 140K if you are running Windows 3.0.  The exact amount depends on your disk's cluster size. If you don't know about your cluster size, don't worry about it.

## Memory Usage

While running, UUCode uses about 60K of memory.

# General Operating Information

DOS and UNIX End Of Line (EOL) Characters
Cooperative Multitasking
Temporary File Usage
Multiple Instances and File Sharing
File Permissions
Encoding Character for Binary Zero
Multipart File Limitations

# DOS and UNIX End Of Line (EOL) Characters

For decoding, UUCode automatically handles any combination of DOS and UNIX end-of-line terminators; those with both Carriage Return (CR) and Line Feed (LF), or just CR or just LF. There is no need to pre-process encoded input to convert the UNIX format to DOS.

When generating encoded files, UUCode can be configured to use either a DOS or UNIX **End of Line String**.

## Cooperative Multitasking

UUCode is a cooperative program with the other applications running under Windows such that it shares the CPU during the encoding and decoding processes which use extensive file I/O. This results in slightly slower performance than if no CPU sharing were done. Even so, it is not recommended that modem communications take place while UUCode is running; file transfers might fail.

## Temporary File Usage

To allow multiple part decoding with the file parts in any order, UUCode creates temporary files.  This means that the disk on which UUCode is operating must be writable and have sufficient free space of twice the size of the input file(s).

## Multiple Instances and File Sharing

UUCode allows multiple instances of itself.  Since Windows 3.0/3.1 is a multi-tasking system, it is possible that the same encoded or decoded file might be used by UUCode or other applications at one time. This could happen if the user chooses the same file name twice by mistake. To solve this problem, UUCode takes advantage of Windows file locking mechanism and does not allow any other application to write the UUCode output file as it is being created. However, multiple read accesses to the input files are allowed.

## File Permissions

When decoding a file, the permissions identified in the encoded file are ignored.  The output file is always created with read and write permissions for all, which is what DOS can support.

To be consistent with the UNIX versions of encoding, the file permission data are set to owner read/write, with read-only permission for group and other (file mode "644").

# Encoding Character for Binary Zero

Although the UU encoding specification allows either an ASCII space (hex character 20) or a ASCII back-quote (` - hex character 60), UUCode will always UU encode its binary zero (0) output using the back-quote (`) character.  This allows for easier viewing of the UU encoded file.

For decoding, UUCode will accept input files with either the space or the back-quote character without problems.

XX encoding always uses the plus '+' character for binary 0.

## Multipart File Limitations

For the decoding operation, UUCode allows the individual parts of a multiple part encoded binary file to exist in any order, e.g. not be restricted to sequential order. However, *all* the parts of one encoded binary file *must* be present in one or more input files to be decoded.

In addition, no more than one binary file's worth of encoded data can be present in any set of encoded files.

In summary, the encoded file(s) must contain all the parts of one and only one binary file for the decoding operation. In either case, UUCode will detect and report any violations of these requirements should they occur.

# Encoding Header Formats

Ideally, only one header format would be required for UU or XX encoded files. However, the multiple part encoded formats were designed to allow a large file transfer by encoding one large binary file into multiple parts for transmission across a network. This is necessary because the E-mail on many systems cannot handle more than 64K of data (or less) for any one file. Today, large binary files are common which would prohibit sending them via E-mail unless the multiple part format is used. The multiple part formats supported by UUCode are briefly described below. In the examples, it is assumed that three parts are used and that they are in order (not a requirement for UUCode).


Single File Format

Internet's SIMTEL20 Archive Site Format

Usenet Newsgroup comp.binaries Format

Usenet Newsgroup alt.binaries Format 1

Usenet Newsgroup alt.binaries Format 2

Usenet Newsgroup alt.binaries Format 3

Usenet Newsgroup alt.binaries Format 4

UNIX Shell Archive Format

DOS uuencode/uudecode Program Format

X-File Format

UUXFER Program Format

WINCODE Program Format

## Single File Format

```
begin 644 archive.zip
Encoded data goes here
end
```

## Internet's SIMTEL20 Archive Site Format

```
------------ Part 1 of 3 ------------
begin 644 archive.zip
Encoded data goes here
-------- End of part 1 of 3 --------
------------ Part 2 of 3 ------------
Encoded data goes here
-------- End of part 2 of 3 --------
------------ Part 3 of 3 ------------
Encoded data goes here
end
-------- End of part 3 of 3 --------
```

## Usenet Newsgroup comp.binaries Format

```
Archive-name: archive/part01
BEGIN--cut here--cut here--
begin 644 archive.zip
Encoded data goes here
END--cut here--cut here--
Archive-name: archive/part02
BEGIN--cut here--cut here--
Encoded data goes here
END--cut here--cut here--
Archive-name: archive/part03
BEGIN--cut here--cut here--
Encoded data goes here
end
END--cut here--cut here--
```

## Usenet Newsgroup alt.binaries Format 1

NOTE: For the four following alt.binaries formats, UUCode will accept decoding "Subject:" lines with any of the following formats. For encoding, however, the output is exactly as shown in the examples below. 'X' identifies the current part number and 'Y' identifies the total number of parts.

The filename may have an extension (filename.ext), no extension (filename or description) or be preceded by a dash '-' (- filename.ext).

The parts count identifiers can be in any of the formats (X/Y), (partX/Y), (part X/Y), (part X of Y), [X/Y], [partX/Y], [part X/Y], or [part X of Y].

```
Subject: archive.zip (1/3)
e-mail header data goes here
BEGIN --- CUT HERE --- Cut Here --- cut here --- archive.zip
begin 644 archive.zip
Encoded data goes here
--
Subject: archive.zip (2/3)
e-mail header data goes here
BEGIN --- CUT HERE --- Cut Here --- cut here --- archive.zip
Encoded data goes here
--
Subject: archive.zip (3/3)
e-mail header data goes here
BEGIN --- CUT HERE --- Cut Here --- cut here --- archive.zip
Encoded data goes here
end
--
```

## Usenet Newsgroup alt.binaries Format 2

Subject: archive.zip [part 1/3]

*e-mail header data goes here*

BEGIN------------------>  cut here  <------------------------

begin 644 archive.zip

*Encoded data goes here*

END--------------------->  cut here  <------------------------

Subject: archive.zip [part 2/3]

*e-mail header data goes here*

BEGIN------------------>  cut here  <------------------------

*Encoded data goes here*

END--------------------->  cut here  <------------------------

Subject: archive.zip [part 3/3]

*e-mail header data goes here*

BEGIN------------------>  cut here  <------------------------

*Encoded data goes here*

end

END--------------------->  cut here  <------------------------

## Usenet Newsgroup alt.binaries Format 3

Subject: archive.zip [1/3]

*NOTE: blank lines after e-mail header indicate start of encoded data*

begin 644 archive.zip

*Encoded data goes here*

--

Subject: archive.zip [2/3]

*NOTE: blank lines after e-mail header indicate start of encoded data*

*Encoded data goes here*

--

Subject: archive.zip [3/3]

*NOTE: blank lines after e-mail header indicate start of encoded data*

*Encoded data goes here*

end

--

## Usenet Newsgroup alt.binaries Format 4

Subject: archive.zip [1/3]

*e-mail header data goes here*

BEGIN----------CUT HERE----------

begin 644 archive.zip

*Encoded data goes here*

END----------CUT HERE----------

Subject: archive.zip [2/3]

*e-mail header data goes here*

BEGIN----------CUT HERE----------

*Encoded data goes here*

END----------CUT HERE----------

Subject: archive.zip [3/3]

*e-mail header data goes here*

BEGIN----------CUT HERE----------

*Encoded data goes here*

end

END----------CUT HERE----------

## UNIX Shell Archive Format

```
#!/bin/sh
#
# This is a shell archive. Cut everything off before
# the #!/bin/sh and feed the rest to /bin/sh
#
part=1
file=archive.zip
sed -e '/^BEGIN/d' -e '/^END/d' << \End_of_Section > $file.uue.$part
BEGIN------------ archive.zip ------------ part 1/3 ---
begin 644 archive.zip
Encoded data goes here
END-------------- archive.zip ------------ part 1/3 ---
End_of_Section
echo $file, part $part extracted.
if [ `echo $file.uue.[0-9]* | wc -w` = 3 ]; then
cat $file.uue.* | uudecode
if [ $? -gt 0 ]; then
  echo Error encountered when uudecoding pieces...
exit 1
fi
echo $file successfully uudecoded. Removing uuencoded pieces.
rm $file.uue.[0-9]*
fi
exit
#!/bin/sh
#
# This is a shell archive. Cut everything off before
# the #!/bin/sh and feed the rest to /bin/sh
#
part=2
file=archive.zip
sed -e '/^BEGIN/d' -e '/^END/d' << \End_of_Section > $file.uue.$part
BEGIN------------ archive.zip ------------ part 2/3 ---
Encoded data goes here
END-------------- archive.zip ------------ part 2/3 ---
```

```
End_of_Section
echo $file, part $part extracted.
if [ `echo $file.uue.[0-9]* | wc -w` = 3 ]; then
cat $file.uue.* | uudecode
if [ $? -gt 0 ]; then
  echo Error encountered when uudecoding pieces...
exit 1
fi
echo $file successfully uudecoded. Removing uuencoded pieces.
rm $file.uue.[0-9]*
fi
exit
#!/bin/sh
#
# This is a shell archive. Cut everything off before
# the #!/bin/sh and feed the rest to /bin/sh
#
part=3
file=archive.zip
sed -e '/^BEGIN/d' -e '/^END/d' << \End_of_Section > $file.uue.$part
BEGIN------------ archive.zip ------------ part 3/3 ---
Encoded data goes here
end
END-------------- archive.zip ------------ part 3/3 ---
End_of_Section
echo $file, part $part extracted.
if [ `echo $file.uue.[0-9]* | wc -w` = 3 ]; then
cat $file.uue.* | uudecode
if [ $? -gt 0 ]; then
  echo Error encountered when uudecoding pieces...
exit 1
fi
echo $file successfully uudecoded. Removing uuencoded pieces.
rm $file.uue.[0-9]*
fi
exit
```

## DOS uuencode/uudecode Program Format

section 1 of uuencode 5.22 of file archive.zip

*NOTE: blank lines indicate start of encoded data*

begin 644 archive.zip

*Encoded data goes here*

*NOTE: blank lines indicate end of encoded data*

section 2 of uuencode 5.22 of file archive.zip

*NOTE: blank lines indicate start of encoded data*

*Encoded data goes here*

*NOTE: blank lines indicate end of encoded data*

*section 3 of uuencode 5.22 of file archive.zip*

*NOTE: blank lines indicate start of encoded data*

*Encoded data goes here*

end

*NOTE: blank lines indicate end of encoded data*

## X-File Format

```
X-File-Name: archive.zip
X-Part: 1
X-Part-Total: 3
BEGIN------------------------cut here-----------------------
begin 644 archive.zip
Encoded data goes here
END------------------------cut here-----------------------
X-File-Name: archive.zip
X-Part: 2
X-Part-Total: 3
BEGIN------------------------cut here-----------------------
Encoded data goes here
END------------------------cut here-----------------------
X-File-Name: archive.zip
X-Part: 3
X-Part-Total: 3
BEGIN------------------------cut here-----------------------
Encoded data goes here
end
END------------------------cut here-----------------------
```

## UUXFER Program Format

```
archive.zip    section  1/3   UUXFER X.Y
BEGIN----------------------CUT HERE--------------------------
begin 644 archive.zip
Encoded data goes here
END------------------------CUT HERE--------------------------
archive.zip    section  2/3   UUXFER X.Y
BEGIN----------------------CUT HERE--------------------------
Encoded data goes here
END------------------------CUT HERE--------------------------
archive.zip    section  3/3   UUXFER X.Y
BEGIN----------------------CUT HERE--------------------------
Encoded data goes here
end
END------------------------CUT HERE--------------------------
```

## WINCODE Program Format

[ Section: 1/3  File: archive.zip  Encoder: Wincode vX.Y ]

*NOTE: blank lines indicate start of encoded data*

begin 644 archive.zip

*Encoded data goes here*

*NOTE: blank lines indicate end of encoded data*

[ Section: 1/3  File: archive.zip  Encoder: Wincode vX.Y ]

[ Section: 2/3  File: archive.zip  Encoder: Wincode vX.Y ]

*NOTE: blank lines indicate start of encoded data*

*Encoded data goes here*

*NOTE: blank lines indicate end of encoded data*

[ Section: 2/3  File: archive.zip  Encoder: Wincode vX.Y ]

[ Section: 3/3  File: archive.zip  Encoder: Wincode vX.Y ]

*NOTE: blank lines indicate start of encoded data*

*Encoded data goes here*

end

*NOTE: blank lines indicate end of encoded data*

[ Section: 3/3  File: archive.zip  Encoder: Wincode vX.Y ]

# If You Encounter Problems

If you encounter problems, check the following areas for more information. If those suggestions don't work, please contact technical support.

What if the Multiple Part UU Format I Use Isn't Supported?

What if the Output File Seems Corrupt?

## What if the Multiple Part UU Format I Use Isn't Supported?

If you *know* that the format of your encoded input file is not one of those supported by UUCode, then you must manually put the encoded data portion only of each file part into the correct order for UUCode to work. Any e-mail header information must also be stripped out. Keep only the portions starting with 'begin' and up to and including 'end'. Use any standard text editor, such as Windows NOTEPAD or DOS's EDIT, to do this.

If you are not sure what multiple part type is used, try running UUCode anyway and check that UUCode reports no errors and check the output file as per the **What if the Output File Seems Corrupt** section. The formats supported by UUCode are described in **Encoding Header Formats**.

## What if the Output File Seems Corrupt?

Typically UUCode is used to send binary files containing compressed archives. Those files typically end in extensions of .ZIP, .ARC, .LZH, or .ZOO. If you find that UUCode reports a complete and correct decoding but your archive file has problems (e.g. checksum errors), then it is likely that UUCode cannot filter any e-mail header information contained within the file or the multiple part format used is not supported by UUCode.

Try manually putting together the encoded file using any ASCII text editor such as Windows NOTEPAD or DOS's EDIT. Then re-run decoding on that file.

# Exit

File Menu / Exit Menu Item

General Configuration / Exit When Done En/Decoding