Contents

The following Help Topics are available:

For Help on Help, Press F1

Last revised: 25 February, 1997.

What is UUDeview?

UUDeview is a program that converts binary files to and from the "printable ASCII" formats used to transport binary data over the Internet by e-mail or Usenet. It has two basic modes, "encode" and "decode."

Why UUDeview?

It's not possible to directly send binary files via Internet Mail or to post them in the Usenet newsgroups. The protocols used to carry e-mail and news are designed to transport only "printable ASCII" characters.

Over the years, a number of different schemes have appeared for translating (or "encoding") binary files so they can travel safely via Net mail and Usenet.

These schemes go by names like MIME, UU-encoding, XX-encoding, and BinHex encoding. A binary file that has been encoded in this way looks like gibberish, often with a few intelligible lines at the beginning (for example, you might recognize the original filename).

No matter which scheme is used, the basic idea is the same: the sender of a binary file encodes it, then transmits the resulting text. To get back the original file, the recipient simply runs the message text through a decoder. UUDeview can perform both the encoding and decoding steps.

Further complexities can arise when a large file is transmitted. Some e-mail and news systems have limits on the maximum message size they can carry; to circumvent these limits, encoded files are broken into multiple parts. UUDeview can recognize, correctly arrange, and decode multi-part encoded files. It can also generate multi-part encoded files.

A History of Decoding

Frank Pilhofer's hopefully humorous introduction to how mail and news came to be encoded, along with some hints on how to make the best of the bad situation we always seem to find ourselves in.

Sections:

Bits of History

> *This text is fiction. There's some truth in it, but its main purpose is to entertainingly show how problems arise. It's also a little technical. And it's also not essential. If you don't want to know, skip the section. The other sections explain the different methods of encoding and give some guidelines for encoding files. If you are only interested in decoding, read on, too. The understanding of encoding will also help you to understand the decoding process.*

Once upon a time, in a not so far away land, hardware engineers thought of a basic calculation unit for their computers, packaged 8 bits together and called it a byte. It would store 256 different values that could be used for machine instructions. Well, this definition was broken from time to time with 7 or 9 bit machines, but the 8 bits were eventually accepted as a good choice. Note that at first, nobody thought of storing text strings in computer memory, which was far too precious to be filled with stupid messages like *The Result is* or *Hello World*. People were content with receiving a punch card with an equivalent of 3.14157354 coded into it after letting the machine calculate for 5 hours.

It was years later that memory was getting so cheap that some computers were equipped with a kilobyte or more, and results were printed on a small computer screen, when finally executives decided that a machine that cost millions to build should be able to express itself in a more human-readable form. It really quite probably was the executives that made this decision. Computer scientists didn't need such a feature; they could all read the punch cards and were busy fixing the fifth decimal of their pi approximation. Then the executives fixed pi to be 4, and relieved the scientists of their quest for more accuracy. (Look at the *Guinness Book of Records*. It was in a different context, but mentions the definition of pi=4 by some court).

Everything really took off when the first *terminals* were hooked up to the computer. Just imagine, no more punching the right unlabeled buttons, but now the computer could read commands in more or less plain english. Now the characters must be transferred from the keyboard into the machine, and then from the machine back onto the display. For this transmission the letters and digits were to be encoded. But how? The 256 values of a byte are obviously not useful for this task: there are much less than 256 usable characters. The engineers decided that there was no need to waste a full 8 bits on a single character, so they used only 7 bits with 128 values to encode a letter. And still this is too much, so they invented lots of special characters to tell the machine on one end or the human on the other end what's going on, like characters for *end of input* or *ring bell*. You can guess the executives were enthusiastic when the machine accepted their inputs with a gentle but firm *ping!*

This solution was perfect for a couple of years, yes, even for more than a decade, until the price for computers dropped below a couple of hundred thousand dollars, and some institutions found they had some money left for a *second* computer. Wow, to have two computers, *that* was power! Then students with too much time on their hands started to connect the machines together, and wrote software to communicate from the first one to the other one, and send messages over that link. For that, they adopted the established protocol of a machine talking with its terminal. Only they did not use 7 bits for encoding but 8 bits; the new bit was used as parity, as checksum to see if transmission was successful. But still, they used all the special characters that were invented for the terminal. For years, they were happy sending little pieces of mail from here to there, and eventually, after discovering the advantages of a telephone line, across the country.

The trouble started as computer companies managed to sell more than one computer of the same model, and the users of both machines got to know each other. This means that both machines were *compatible*, that programs written on the one could be copied and run on the other. And it wasn't long that the first user boasted to the second what a neat program he'd written. Then the second one, reading this message by mail thousands of miles away, asked the first user to send the program to him. The user shivered, "but, but ... it's a program! You see, this assembler instruction here is the terminal character of *end of transmission*, it would tell your mail receiving program to terminate the connection and you wouldn't get the rest of it. I can't possibly send it to you!".

Of course, this solution is inadequate. What's the purpose of linking computers together if you can't share programs, or any other data you like? That's a dilemma. The communication links are standardized, and we do not want to start all over again. We can transfer plain text but not binary files. But, what if we *encode* binary files into plain text on the one end, and *decode* it from plain text into the originally binary representation on the other? That'll work, so that's what we'll do!

So programs were written for encoding and decoding, until the next obstacle was hit: some mail transfer programs, that the programmers have just avoided to rewrite, read or wrote mail in fixed-size buffers. If a sender sent more than the recipient's buffer size, the end of the message got lost. Because computer users usually don't write novels, this limit has previously gone unnoticed; but images or movie clips just failed to fit and arrived, if at all, only in pieces. In pieces! That's a splendid idea; if we split up large files, we can then encode and transmit the pieces individually and let the recipient put them together, and then we can transfer everything we want!

Yes, once again the programmers have successfully avoided to go back to the drawing board. They were now free of their problems - because they've loaded them all onto the user, who has now to ask himself, "How do I split the original file up?" "How do I encode the pieces?" "How do I decode incoming pieces?" "How do I put the decoded pieces together?".

The Problem

If you didn't read the above text, or didn't catch the point, here's the major problem we have to face:

» Binary data, that includes software, images, audio, video etc., uses all 256 possible values of a byte.

» Some of these values represent control characters that would have undesired effects upon transmission, or wouldn't transfer correctly.

» So binary data is *encoded* into a set of characters that can be safely transferred. This encoded data is then *decoded* into its original form by the recipient.

» Some transfer programs can only transfer messages of limited size, so large files must be split into pieces.

Note that the word *mailing*, if used, can be replaced by *posting*. Posting messages to the Usenet news system is similar to mailing because the same methods of communication, with all above problems, are employed. When transferring binary files to and from newsgroups, the files must also be split up, encoded and decoded.

You will find that the *UUDeview* package will help you through all steps of encoding files and decoding incoming messages. However, this text continues not to be specific about the program. Following are discussions about four different methods of encoding and a few guidelines.

Four Methods

Everything becomes a little more complicated because there isn't just one way to accomplish all that. In particular, there are four different, incompatible methods to encode binary data into plain text. Sender and recipient must agree on the same method. The four methods are

» **uu-encoding.** This was historically the first method invented. The encoding was quite simple and caused frequent trouble. In particular, first implementation used the space character for encoding. But some mail gateways stripped spaces at the end of a line, so what the recipient got was invalid. Later, a special case was introduced to avoid the problem.

» **xx-encoding.** This rarely used method appeared after the initial problems with uuencoding, but before they were fixed. It also avoided using the space character by using a different character set for encoding.

» **Base64.** This method was introduced by the *MIME* standard, avoiding some rarely encountered problems with the other two method (which used some characters not available on some machines). Together with other benefits of *MIME*, this is the most secure method.

» **BinHex.** A method conceived for transferring files among Macintosh systems. Files on a Macintosh consist of two parts, the "data fork" and the "resource fork". This encoding first adds a third header part, and composes the three parts into a single data stream, which is then slightly compressed and encoded.

Actually, there are a couple of other, less widely used encodings, like *ship* or *btoa*. They are not covered here, since they are rarely seen in "real life".

Encoding Guidelines

Which method to use?

Of course, the question is, "what method shall I use?" The first rule is to avoid xxencoding, which is obsolete. And while BinHex is a "must" for Macintosh systems, it is usually not a wise choice on other systems, because decoders are not widespread elsewhere. True, BinHex claims to also compress the file and make the transferred data smaller, but don't believe this argument. It does not hold for already-compressed data.

Compression is also one important issue that needs to be mentioned. **Never** send an uncompressed file, this just wastes valuable bandwidth. GIF and JPEG images are well-compressed themselves, but other types of data should always be compressed into a ZIP file or something similar.

This leaves uuencoding and Base64. This argument should be decided with consideration of your mail software. If it is MIME-compliant and offers to "attach" files encoded in Base64, use this encoding. Base64 is the preferred method for MIME messages.

Otherwise, use uuencoding, which is still the most common encoding method. Because more and more software becomes MIME-compliant, it is expected that uuencoding is completely replaced by Base64, but as long as there is the *possibility* that the recipient still uses old software, uuencoding is the safest method.

Note that MIME-compliance is something only the mail software can handle. For example if you encode data to Base64 using UUEnview and then *include* the encoded data in your message, the resulting message will **not** be MIME-compliant! This fact is important to realize. If your mail software does not allow "attachments" on its own and you have to use an external encoder, always use uuencoding.

To Split or not to Split

Then there's another question that applies similarly to all methods, the question of splitting files. Sending everything in a single mail or posting is easiest, both for the sender and the recipient, because you don't have to fuss around with splitting and putting parts together. But you must make sure that the mails or postings aren't stripped somewhere on its way. This isn't much of a problem with email any more, usually you can send megabytes or more at once. I suggest to try sending the full file at first, and then ask the recipient if (s)he got everything of it. If not, you can experiment to find the maximum size and then stick to it.

But news are a different topic. There are still some gateways around allowing no more than a fixed size. The semi-accepted limit is to send only thousand lines of encoding per post, and to split large files into parts of thousand lines each.

Include Information

When you mail or post a file, people will want to know what they can expect from it before having to decode it. Also, most people will have to *download* the encoded file before they can even decode it. These people can get quite annoyed if they discover they've downloaded something entirely inappropriate. To avoid annoying people, you should **always** send a small message what this file is all about. Don't just say, "you must have this!", make the message informative.

For small files with less than thousand lines of encoding, you can include this message with the encoding, but if it's more, you should send a separate mail or post.

Composing a Subject Line

The last problem we have to face is how to build up a subject line for the mail or the posting, so that people can easily spot the file and, in the case of multiple parts, know which postings belong together for the single file. Here's an example:

```
UUDeview 0.5a for Windows - uudvw05a.zip (001/004)
```

First on the subject line is a short description of the file, less than 40 characters. Then, separated with a dash is the original filename. Last, enclosed in brackets, is the number of this part, and the total number of parts. In this case, the reader will know that (s)he also has to get parts two to four to decode the file. A subject line like this includes all necessary information.

The informative message from above is usually sent as the zeroeth part. This zeroeth part should only be a textual description and should not include any encoded data. If a part number zero is present, people will read it and only this part to see whether they'll want to decode the rest of the file or not.

BTW, you should also include the part numbering if there's only a single part. This should then read (001/001). Then people will know they don't have to search for more.

Summary

To summarize our discussion ...

» If your mail or news software is MIME-compliant and allows to directly "attach" binary files, use Base64. If you use an external encoder to include encoded data into your messages, use uuencoding.

» There's no real need to split encoded files for email transmission; but restrict yourself to no more than 1000 lines per part when posting into a newsgroup (unless it's a really huge file; don't post more than 100 parts).

» Create a subject line with all necessary information for the reader and the decoder. *UUDeview* will need a proper subject line to combine the parts.

» Send a part number zero with a description of the file.

Well, I hope you've enjoyed this little introduction, and learned a little from it. In fact, I hope you'll stick to the mentioned guidelines. Me and lots of other people are sick of trying to decode files that were sent without any guidelines in mind.

UUDeview DLL

If you want to incorporate UUDeview's encoding and decoding functionality into your own programs, there is a DLL version of the program available. The DLL is suitable for use with Visual Basic, C, or any other program that can call Windows Dynamic Link Libraries.

You can download the DLL from the official UUDeview Web page (http://www.uni-frankfurt.de/~fp/). Currently, the DLL is only available in 32-bit form (for Windows 95 and NT), but a 16-bit version will probably eventually be released.

Revision History

Version 1.0 – July 31, 1996

» Initial version.

Version 1.1 – December 15, 1996

» Fixed problems affecting certain BinHex files.

» Improved decoding algorithms.

» Slightly reduced consumption of Windows Resources.

» Context menus for "files to decode" and "decode preview" windows.

» New "Launch" and "Decode & Launch" commands.

» Decode Preview window now shows marker indicating text information ahead of binary data.

» "Tool Tips."

» Encoding to clipboard.

» Online help.

» Option added to show debug output during loading and decoding.

» Decode Preview operation shows explicit error message if no encoded data found.

» Example file added to distribution.

Version 1.1a – February 25, 1997

» Fixed problem that caused UUDeview to crash if a file is renamed during a decode operation.

FAQ: No "GO" Button

Why can't I find the "Go!" button?

There are two common reasons why the "Go" button is not visible:

» UUDeview may not have been able to find anything to decode in the input files you supplied.

» You may have skipped a step in the decode process.

Let's review, with a specific case example. Let's suppose you have a file called `NARF.UUE` that you want to decode, and that this file is found in the directory `C:\POIT`.

1. Start UUDeview.
2. Make sure the "Decode" button is clicked in the "Function" box.
3. Click the "Add..." button in the lower left corner, below the "Decode these files" window.
4. You'll get a normal Windows "Open File" box. Navigate to the directory `C:\POIT`, highlight the file `NARF.UUE`, and click "Open" or hit <Enter>.
5. You should see "`c:\poit\narf.uue`" appear in the "Decode these files" box. The "Preview" button should turn from gray to "normal."
6. Click the Preview button. UUDeview will search `NARF.UUE` looking for files to decode. If it finds any files, their names will appear in the large "To:" box on the right half of the window. The Preview button will change to "Go."
7. If nothing appears in the "To" box, then there may not be any decodable files in `NARF.UUE`. If you want to, you can click the Options button and turn on "Desperate Mode." This will allow you to decode some corrupted files that normal mode won't find. Then you would click "Preview" again. However, Desperate mode isn't recommended for most situations.
8. If files have appeared in the "To" box, you are ready to decode. Set the "Path" field to the name of the directory where you want the decoded files sent, then click "Go." UUDeview will decode the files.

If you have followed these steps and the "Go" button does not appear, there just may not be anything in your source file to decode, or the file may be too badly corrupted to be comprehensible. If you are pretty certain that the file is correct, you might want to e-mail it to Frank or myself: we collect strange encoded files, with the hope that future versions of UUDeview will decode ever more formats.

FAQ: encoding MIME and e-mail

Why can't I create MIME-encoded files that e-mail programs will understand?

UUDeview can encode in Base64 format, which is generally what is used with MIME, but there is a reason that UUDeview doesn't attempt to create "full" MIME headers: to be truly MIME compliant, some of the information has to go into the Header block of the message, that is, before the message body.

For example, take a look at a typical e-mail message in MIME format. If you look at the headers (not the message body), you should see things like:

```
To: ....
From: ....
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Subject: ....
```

With an attached file, there'd also be some other stuff.

The problem here is that most mail programs won't let you directly insert things into the header block. You can set the "Subject" field, for example, but you can't insert the necessary content-type and content-boundary lines. If you put them into the body of the message (i.e., paste in the output from UUDeview), the receiver probably won't realize that anything is there. It'll assume that the MIME flag lines are part of the message body.

Eventually, especially on Win32, we plan to offer direct mailing and posting of files, and for those situations (where we control the header block) "real" MIME should be no problem. Until then, the safest way to send messages is to use UU, which, though less efficient, is recognized by nearly all mail programs.

Theoretically, it might be worth optionally inserting the simplistic headers (filename, etc.). This would allow the recipient to at least decode the files with UUDeview, even if his mail program doesn't recognize the MIME tags. We may do this in a future version of UUDeview.

Note that even if we did this, some (or even most?) mail programs might not recognize the enclosed files... they tend to look for flags in the header block.

If no MIME flags are found, they often treat the message as pure text, even though an enclosed document may be correctly "framed" by UUDeview.

FAQ: Crash During Startup

Why does UUDeview crash during startup, usually in my video driver?

A small number of UUDeview users have experienced a crash (GPF) during startup, usually in their video driver (e.g., "`GPF in module OAK8640L.DRV at 0007:05BD`"). In some cases, this crash can be corrected by downloading a more recent version of the Visual Basic Runtime DLL, `VBRUN300.DLL`, which can be found on the UUDeview for Windows homepage.

As far as we can tell so far, the problem is limited to certain rather antique video cards, usually pre-1993 vintage. Unfortunately, if replacing the VB runtime doesn't cure the problem, we currently have no fix, other than to suggest checking to see if an upgraded video driver for your card exists. One very good place to look for updated drivers is the Microsoft Download Library (http://www.microsoft.com/download).

FAQ: Can't Find MHGLBX.VBX

Win95: Why does UUDeview stop during startup, complaining it can't find a VBX?

If the VBX in the error box is indeed in the UUDeview directory, you're experiencing a known problem with Windows 95.

When you double-click an EXE in the Explorer, Windows sometimes starts the target program and then sets the "current directory" to your Windows directory, which can confuse the loading process.

To prevent this, you can try using the right mouse button to drag `UUDeview.exe` from the Explorer to the desktop. When the context menu pops up, choose "Create shortcut here." Then double-click the shortcut: this will launch UUDeview while making sure the current directory is the one where UUDeview is located.

If this works, you can try creating a shortcut somewhere on your Start Menu rather than double-clicking `UUDeview.exe` in the Explorer.

FAQ: Decoding Problem Files

What can I do when UUDeview won't decode a file?

We have tried to design UUDeview so that it will decode even the toughest encoded files. Even so, experience has proven that there are always new mutations. There's also no way to predict all the bad things that can happen to an encoded file in transit.

So, what to do if UUDeview can't figure out a file you need to decode? The first step is to turn on Desperate mode in the Decode Options, then click "Preview" again. This will cause UUDeview to work harder at parsing the input file.

Unfortunately, there's a downside. Normally, when UUDeview tells you that it has successfully decoded a file, you can be fairly certain that the result is exactly what was originally sent. In Desperate mode, this may no longer be true.

Desperate mode tries to decode any data that can be identified. This means that there may well be gaps in the data or other incorrectly-retrieved information. As the name implies, you should really use Desperate mode when you need to get something—anything—from an encoded file.

If Desperate mode doesn't help, you can try looking at the Debug Info. UUDeview can generate extensive information about what it is doing as it parses and decodes your input files; enabling Show Debug Info in the Decode Options will cause UUDeview to show you this information. Sometimes the Debug Info will tell you enough to manually edit the encoded file and repair it.

If not, you should prepare yourself for the worst: you simply may not have a decodable file. If you're truly desperate you can ZIP the encoded file and send it to one of the UUDeview's authors; otherwise, try to get the sender of the problem file to try again.

Whatever happens, remember to turn off Decode Info and Desperate mode when you're done wrestling with your troublesome file. Desperate mode in particular can interfere with the decoding of correctly-formatted files, so you should only use it when you have no other choice.

Reminder: Please ZIP encoded files that you sending to UUDeview's authors. This will prevent your problem file from being mangled by the various Internet mail gateways.

FAQ: What about a 32-bit version?

Many people are migrating from Windows 3.*x* to Windows 95 or Windows NT. Given the added power and flexibility of the 32-bit platform, why is UUDeview available only in a 16-bit version?

The answer is surprisingly simple: download size. The 16-bit version of UUDeview was developed with Visual Basic 3. VB3 programs can run on any Windows computer and they require only a single support DLL (dynamic link library): `VBRUN300.DLL`. Even including this help file, the current UUDeview download archive is only about 500K in size.

The authors of UUDeview have created a 32-bit port of UUDeview using Visual Basic 4.0. VB4 builds executables that are notably larger than VB3, but that isn't the real problem. Unfortunately, VB4 requires several very large support DLLs; for example, the 32-bit equivalent of `VBRUN300.DLL`, which is called `VB40032.DLL`, is about 700K in size. Then there are several other DLLs which may be needed depending on what has already been installed on the user's system.

Including the UUDeview executable, the support DLLs, the necessary OCXes (the 32-bit equivalent of VBXes), and a Win95-compliant install/uninstall program, the archive file for a 32-bit version of UUDeview would reach about 3 megabytes. Yes, that's 3,000,000+ bytes, even in compressed form… nearly six times the size of the current download archive.

As far as UUDeview is concerned, the only major benefit the 32-bit platform offers is the ability to use long filenames. The decoding speed of the 32-bit version is only slightly better, and the program's other functions operate more or less the same in both versions.

Long filenames don't seem to be worth a sixfold increase in download time. UUDeview's design goals included a reasonably-sized download archive, a small footprint on the user's hard disk, and a simple installation procedure. Moving to 32 bits would negate most of these goals.

Remarkably, it is technically possible to use long filenames from 16-bit mode; a future version of UUDeview will support long filenames when run on Windows 95 or Windows NT even though UUDeview will remain a 16-bit program.

UUDeview will move to 32 bits when Microsoft (or some other vendor) offers a way to distribute 32-bit applications without requiring enormous support files and complex installation procedures.

Contacting the Authors

You can get more information about UUDeview and the UUDeview DLL from the official UUDeview Web Page: http://www.uni-frankfurt.de/~fp/

In the last extremity, you can send mail to:

VB Front End: Michael Newcomb

Mail: Michael Newcomb <uud@miken.com>

Web: http://www.miken.com

UUDeview page: http://www.miken.com/uud

UUDeview: Frank Pilhofer

Mail: Frank Pilhofer <fp@informatik.uni-frankfurt.de>

Web: http://www.uni-frankfurt.de/~fp/

Reminder: If you are sending us an encoded file that you can't decode, please ZIP it. This will prevent your problem file from being mangled by the various Internet mail gateways between you and us.

Decoding

If you need to retrieve a binary file that's been encoded, you'll need to use the Decode mode. There are five steps to decoding: selecting Decode mode, adding the file or files to decode, setting the destination path, previewing the decode, and performing the decode.

Quick Summary

If you want to get started in a hurry, just follow these steps:

» Click the "Add..." button and add the files you want to decode. or click the "Clipboard" button to add the current contents of the Windows Clipboard.

» Set any Options if desired (though the defaults should be fine in nearly all cases).

» Click the "Preview" button to see what encoded files UUDeview has found.

» Set the "To" path to determine where UUDeview will write out the decoded files.

» Click the "Go!" button to decode the files.

» Click the "Clear" button if you want to start over and decode another set of files.

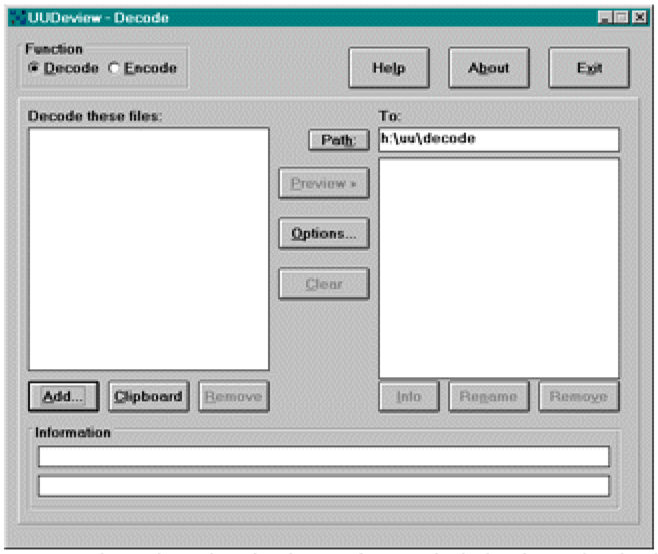
To learn more, consult the Decoding Details section.

This help file also includes an Example of how to decode an encoded file using UUDeview.

Decoding - Example

To help you learn how to decode files with UUDeview, the distribution ZIP file contains a small sample encoded file called `SAMPLE.UUE`. This file contains a GIF graphic that's been UU-encoded.

This example will show you how to decode the sample file in a simple step-by-step way. When you start UUDeview, you'll see a window that looks something like this:



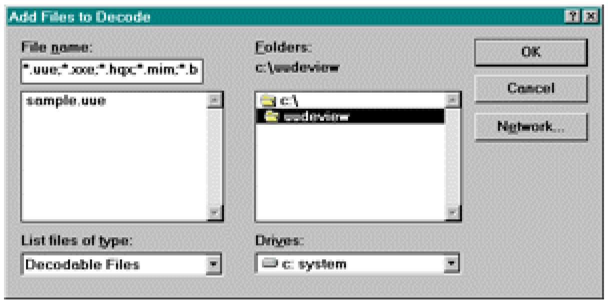
Notice that both the left- and right-hand windows are empty. This means that no files have yet been selected for decoding.

Step 1. Adding the file to decode

In this case, we want to decode the example file `SAMPLE.UUE`. It will be in the same directory as UUDeview. In the example, UUDeview is installed in the directory `c:\uudeview`.

You need to tell UUDeview that you want to decode this file. There are two ways to do this:

» The first way is to click the "Add" button below the "Decode these files" list. This will bring up a standard Windows "Open File" dialog box. Navigate to the directory where UUDeview is installed; you'll probably see something rather like this (though the drive and path will depend on your system's setup):



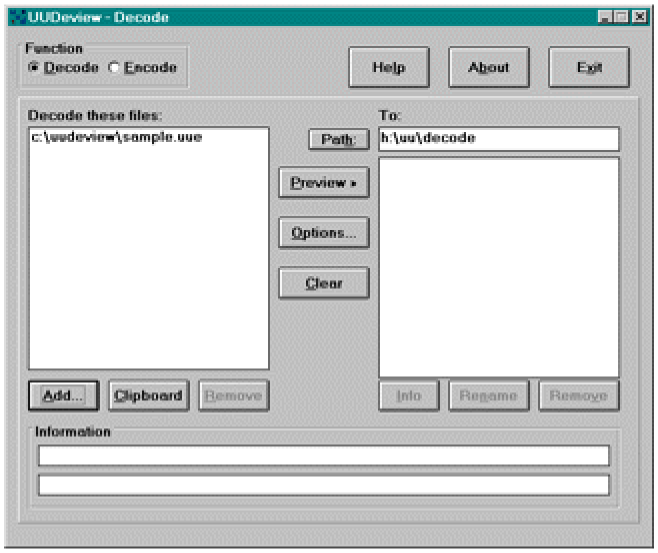
Click on "`sample.uue`" in the left window and click OK. This will tell UUDeview that `sample.uue` is the file you want to decode.

Tip: If you want to decode more than one file in a particular directory, you can select a group of files by clicking on the first file, holding down the <Shift> key, then clicking on the last file. To select multiple files that are not adjacent, click on the first file, then hold down the <Ctrl> key and click on the other files you want to decode.

Tip: If the file you need to decode doesn't end in a "standard" extension for encoded data (e.g., `.UUE`, `.MIM`, `.HQX`), change the "List files of type" box to "All Files." This will show you all the files in a particular directory, regardless of the file extension.

» The second way to add files is to drag them onto the UUDeview window from the Explorer or File Manager. To do this, examine the directory where UUDeview is installed using Explorer or File Manager. Find `sample.uue` and click on it with the left mouse button and *hold the mouse button down.* Next, drag the file from the File Manager / Explorer window over the UUDeview window and release the mouse button.

Whichever way you add `sample.uue`, you should see a UUDeview window that looks something like this:



c:\uudeview\sample.uue

h:\uu\decode

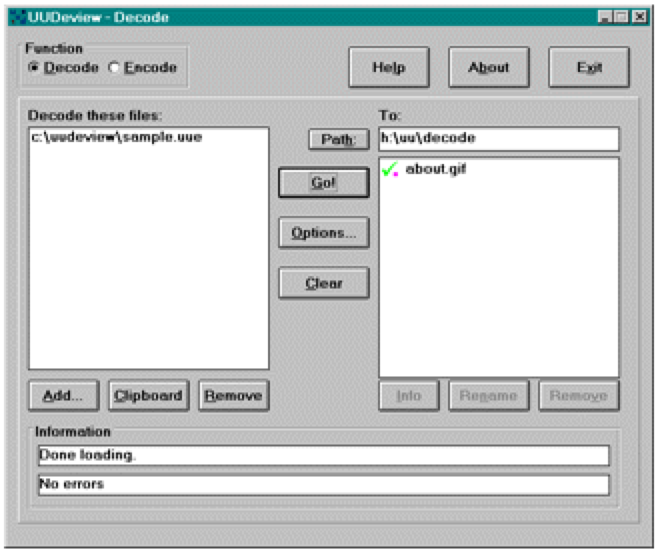
Notice that `sample.uue` now appears in the left-hand window "Decode these files." This window shows the files that UUDeview will examine when you click "Preview."

For the example case, `sample.uue` is the only file you need to have in the left window. When you decode your own files, you can place as many encoded files as you want in the "Decode these files" list, in any order. UUDeview will automatically figure out which files go together.

Step 2. Preview

Once you've told UUDeview what files you want to decode (just `sample.uue` in this case), you're ready to perform the Preview step. This tells UUDeview to examine the encoded files and see what binary data they contain.

To perform the preview, click the "Preview" button. UUDeview will examine the encoded file `sample.uue` and show you what it finds in the right hand window, like this:



c:\uudeview\sample.uue

h:\uu\decode

✓ about.gif

Done loading.

No errors

If you look at the right window, you can see that the example file `sample.uue` contains only one binary file, `about.gif`. The check mark to the left of `about.gif` indicates that UUDeview thinks the file will decode OK.

Step 3. Set the Output Path

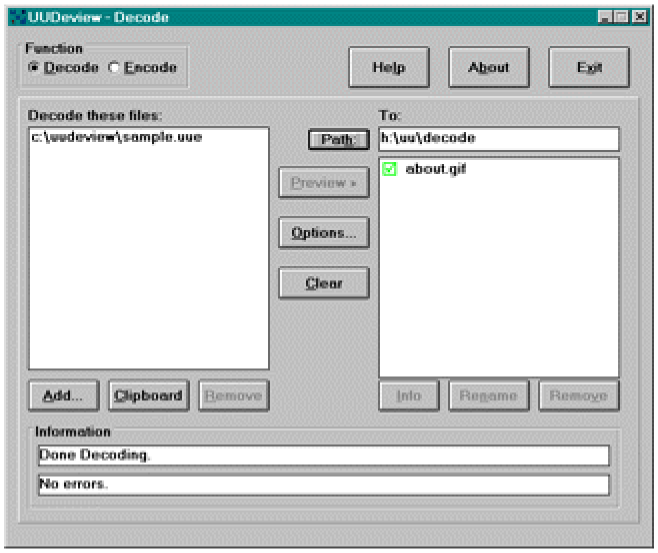
Now you need to tell UUDeview where to write the decoded file: the Output Path. The Output Path is found in the small box above the preview list. In the example, the Output Path is `h:\uu\decode`.

When you start UUDeview for the first time, it will fill in the Output Path with whatever happens to be the current directory at the time. To decode the file into another directory, fill in its name in the Output Path box. You can also browse for the correct Output Path by clicking the "Path" button next to the Output Path box.

Step 4. Decode the File!

Almost done! Once you've set the Output Path, all you need to do is click the "Go!" button. UUDeview will decode the file it found, `about.gif`, writing it into the directory in the Output Path box (`h:\uu\decode` in this case).

After you click the "Go!" button, UUDeview will decode the file, then the window will look something like this:



Function: [✓ Decode] Encode

Decode these files: c:\uudeview\sample.uue

To: h:\uu\decode

✓ about.gif

Done Decoding.

No errors.

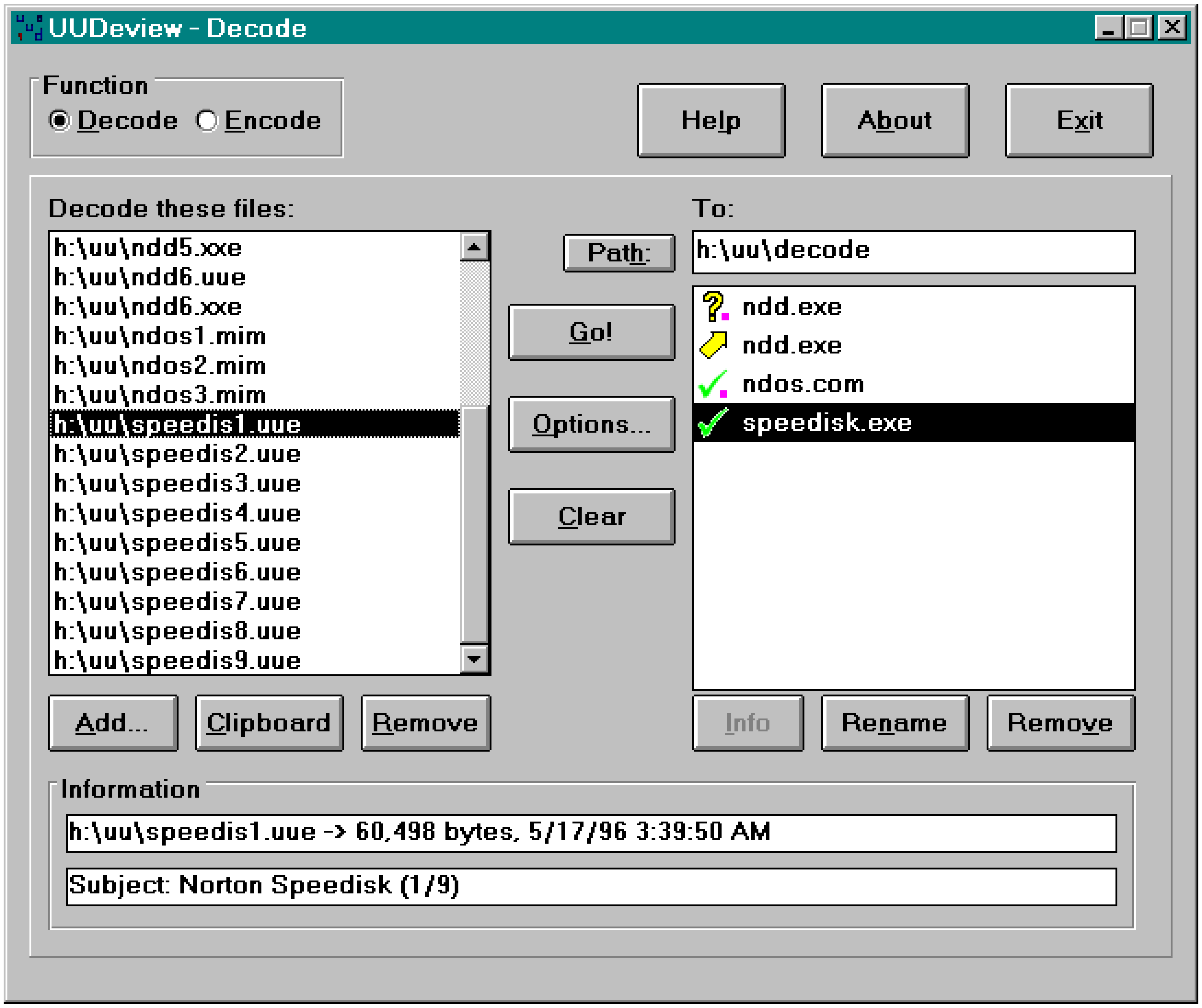
You'll notice that the check box next to `about.gif` now has a box around it. This means the file has been decoded successfully. In the example case, you'd now find the decoded file as `h:\uu\decode\about.gif`.

That's it! You're done!

Decoding - Details

To enter Decode mode, click the "Decode" button in the "Function" box. After the preview step, the UUDeview decoding window looks something like this:

(Click on the image below to get information on the controls)



Function: [✓ Decode] Encode

Decode these files:

```
h:\uu\ndd5.xxe
h:\uu\ndd6.uue
h:\uu\ndd6.xxe
h:\uu\ndos1.mim
h:\uu\ndos2.mim
h:\uu\ndos3.mim
h:\uu\speedis1.uue
h:\uu\speedis2.uue
h:\uu\speedis3.uue
h:\uu\speedis4.uue
h:\uu\speedis5.uue
h:\uu\speedis6.uue
h:\uu\speedis7.uue
h:\uu\speedis8.uue
h:\uu\speedis9.uue
```

To: h:\uu\decode

```
ndd.exe
ndd.exe
✓ ndos.com
✓ speedisk.exe
```

Information:

```
h:\uu\speedis1.uue -> 60,498 bytes, 5/17/96 3:39:50 AM
Subject: Norton Speedisk (1/9)
```

Adding Files to Decode

Begin by adding the file or files that you need to decode, using one of these methods:

» **Add button.** This button lets you choose files from your hard disk using the standard Windows "Open File" dialog box.

Click the "Add..." button and browse to the files, then select them and click "Open" (or press <Enter>). As you add files, they will appear in the "Decode these files" window.

Tip: If the file you need to decode doesn't end in a "standard" extension for encoded data (e.g., `.UUE`, `.MIM`, `.HQX`), change the "List files of type" box to "All Files." This will show you all the files in a particular directory, regardless of the file extension.

» **Drag and Drop.** You can also drag files from the Explorer or File Manager and drop them on the UUDeview window; files dropped in this way will automatically be added to the list of files to decode.

» **Clipboard.** If the information you want to decode is on the clipboard (e.g., you've copied it from an e-mail message), you can click the "Clipboard" button. UUDeview will save the contents of the clipboard for decoding.

  In fact, you can copy and store as many blocks of encoded text as you want in this way: each time you click the Clipboard" button, UUDeview adds the current contents of the clipboard to its storehouse.

Once you've added some files to decode:

» You can click the files in the "Decode these files" window to get information about the files there. When you click on a file, its size and creation date will appear in the "Information" box. If the file is formatted as an Internet message and contains a "Subject" line, that line will also appear.

» If you've mistakenly added a file to decode, you can remove it by selecting the file and clicking "Remove." This doesn't remove the file from your disk; it simply removes the file from the list of files UUDeview will examine.

Using the Context Menus

UUDeview has context menus for the two file lists. Clicking on either the to-decode list or the preview list with the right mouse button will summon a context menu that will allow you to perform certain less-commonly-used commands. For more information, see:

» The To-Decode List Context Menu

» The Decode Preview List Context Menu

Setting the destination path

This might be a good time to specify where you want your decoded files to end up. You can either type a destination path directly into the "To Path" box, or you can click the "Path" button to browse to your destination.

The Preview

As soon as you've added at least one file to the list of files to decode, the "Preview" button will be enabled. Clicking this button will cause UUDeview to examine the encoded files you've supplied and list any binary files it finds. These encoded files will appear in the right-hand window, the output file list.

Each encoded file found by UUDeview will be preceded by a symbol:

Symbol	Meaning
✓	File should decode OK.
✓▪	File should decode OK (with "Info"). The small purple box below the check mark (or any of the other symbols) indicates that there was text information ahead of the binary data. To see this information, select the file and click the "Info" button.
✗	File probably won't decode OK. Generally, this means that the

	source files were formatted incorrectly or that parts of a multi-part file are missing.
?	File *might* decode properly. This symbol usually means that the "end" marker of an encoded file was missing.
✎	Duplicate filenames. This encoded file has the same name as the file above it (see the two files called `ndd.exe` in the example).
→←	Conflict. A file with this name already exists in the output directory you have selected.
☑	Decoded OK. After the decode process, this symbol means that encoded file should be identical with the original source file.
☒	Decoding errors. Something went wrong during the decoding process. You can click on the file and look in the "Information" box to see what the problem was (e.g., missing parts, illegal format, etc.).

» If you click on a file in the detected file list, the "Information" window will tell you the final path and file name that UUDeview will decode that file to. UUDeview will also tell you other things about the file, such as what scheme was originally used to encode it, which source files it came from, and so on.

» **Info.** If you select a file and click the "Info" button, UUDeview will examine the first source file it originated from. Often, this file will contain information about the original binary, for example, a text description, the sender's name, etc. If UUDeview finds any such data, it will display the results in a window. Click "Close" when you've finished examining the file info.

Files that have text information available will include a small purple box at the lower right corner of their status indicator, as shown above.

» **Rename.** To change the name a file will be decoded to, select it and click "Rename." Note that this changes only the name UUDeview will write the decoded data to; no actual existing files are altered.

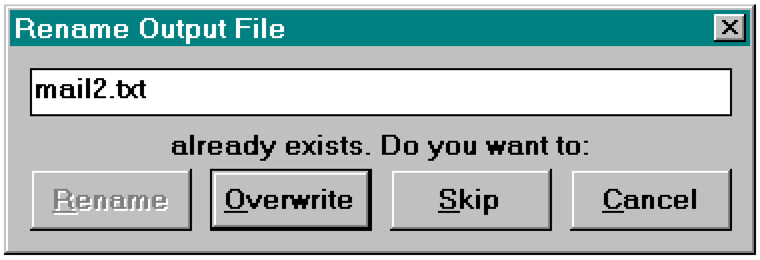
» **Remove.** To prevent UUDeview from decoding a particular file or group of files, select it or them and click "Remove." Note that this doesn't delete any files; it simply prevents UUDeview from decoding the selected files.

Decoding the files

Once you've set the output path and made changes to the output file list (if any), you're ready to decode. Just click the "Go" button! UUDeview will decode the files and set the symbols in the output file list to reflect the results of the operation.

Overwriting files

By default, UUDeview will stop before overwriting any existing files in the output directory. For example, if you're decoding a file called `mail2.txt` and there's already a `mail2.txt` in the output directory you've given to UUDeview, you'll see a dialog box like this during the decode ("Go!") operation:



» To change the file that UUDeview will write the decoded data to, edit the filename in the text field and click "Rename." The "Rename" button will only be enabled if the filename is valid and if there is no file by the same name in the output directory.

» You can elect to overwrite the existing file (`mail2.txt` in this case) by clicking "Overwrite."

» Clicking "Skip" will tell UUDeview not to decode the indicated file.

» "Cancel" stops the decoding operation immediately.

Tip. If you want UUDeview to overwrite existing files without prompting, you can check the Overwrite Files box in the Decode Options.

Starting over

If you want to decode another set of files, just click the "Clear" button. UUDeview will empty the file lists and prepare for the next round of decoding.

Options

UUDeview provides a number of options to alter how it processes your encoded files. Most of the time, the default options should be fine. In the event you need the options, consult the Options topic.

Context Menu, To-Decode List

If you click the right mouse button on the list of files that UUDeview is to decode, the program will display a context menu. This menu includes the following commands:

Remove	Removes the highlighted file or files from the list of files that UUDeview is to decode.
Select None	Clears the highlighting from all the files in the list.
Select All	Highlights all the files in the list.
Invert Selection	Highlights all files that are not currently highlighted; removes the highlight from currently-highlighted files.
Launch	Allows you to launch the application that Windows associates with the highlighted file(s). For more information, see "using the Launch box."
Help	Displays this help.

Context Menu, Decode Preview List

✓ What the Preview List symbols mean

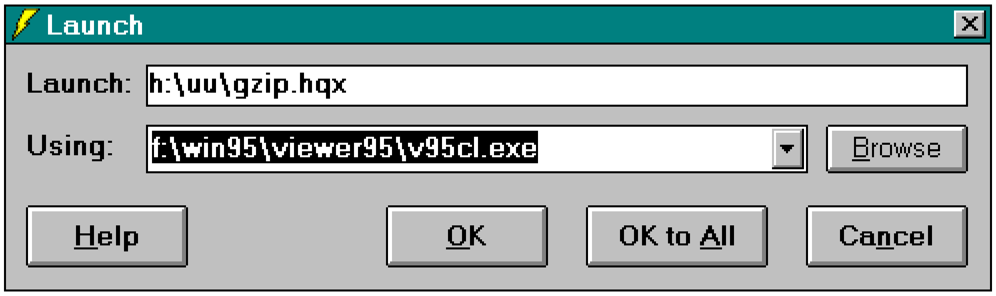
If you click the right mouse button on the list of files that UUDeview has found during a Preview operation, the program will display a context menu. This menu includes the following commands:

Info	Sometimes, there is text information ahead of the binary data in an encoded file. If a file in the preview list has such text, a small purple box will appear next to its status indicator. Choosing this command will display the "ahead of data" text for any file that has such text.
Remove	Removes the highlighted file or files from the list of files that UUDeview will decode.
Rename	To change the name a file will be decoded to, select it and choose the "Rename" command.
Select None	Clears the highlighting from all the files in the list.
Select All	Highlights all the files in the list.
Invert Selection	Highlights all files that are not currently highlighted; removes the highlight from currently-highlighted files.
Decode & Launch	Allows you to launch the application that Windows associates with the highlighted file(s). To carry out this command, UUDeview will decode each highlighted item to a temporary file, then launch the associated application. For more information, see "using the Launch box." **Note:** For your protection, UUDeview will not launch executable files (`.EXE`, `.COM`, `.PIF`, `.BAT`) in this way. Unless you trust a file's source completely, executables downloaded from the Net should be checked for viruses before they are used.
Help	Displays this help.

Launch Box

If you activate the "Launch" or "Decode & Launch" command from the context menus on the Decode page, you'll see a dialog box like this:

(Click on the image below to get information on the controls)



Launch: h:\uu\gzip.hqx

Using: f:\win95\viewer95\v95cl.exe

The "Launch" field shows the file that will be launched. By default, the "Using" field will show the application that Windows "associates" with the Launch file's extension. In the example, the extension `.HQX` is associated with a viewer called `f:\win95\viewer95\v95cl.exe`.

In this case, if you click OK, UUDeview will start the application `f:\win95\viewer95\v95cl.exe` and pass it the filename `h:\uu\gzip.hqx`.

If you want to launch the indicated file with another application, you can type the complete path name in the "Using" box or click the "Browse" button and locate the target program. The last several applications that have been used in the Launch box are listed in the "Using" field's drop down list.

» The "Cancel" button will cancel the current Launch operation.

» If you have selected multiple files and click "OK to All," UUDeview will use the indicated application to launch all of the selected files that have the same extension. In the example, if you were to click "OK to All," UUDeview would use `f:\win95\viewer95\v95cl.exe` to launch all of the select files that have an extension of `.HQX`.

Click this button to display this help file.

This button shows UUDeview's current version and the version of the underlying DLL.

Close UUDeview.

Select Decode mode: convert files *to* binary *from* text.

Select Encode mode: convert files *to* text *from* binary.

List of files to decode. Add to this list with the "Add..." button, the "Clipboard" button, or by dragging files from Explorer / File Manager. Click on this list box with the right mouse button to bring up the context menu.

Add files to the list of files to decode.

Add the current contents of the Clipboard to the items to decode. Repeat as many times as needed; each click appends the current Clipboard contents to whatever has already been collected, then clears the Clipboard.

Remove a file from the list of items to decode. This button removes the file *only* from the to-decode list; the file itself is not affected.

This area shows information about the currently selected file in either the decode list or the preview list. It also shows the progress of lengthy operations.

Click this button to browse for the desired output directory.

This directory is where UUDeview will write the files it decodes when you click "Go."

This is where it all happens. When this button says "Preview," clicking it will cause UUDeview to load and examine the files you've placed in the list of files to decode. The button will then change to "Go!" Clicking "Go!" causes UUDeview to decode the files it found and write them to the output directory.

Click to set the Decoding Options.

Clears both the list of files to decode and the preview list. Click here to start a new decoding session.

List of files that UUDeview found when it searched the list of files to decode. Blank until you press "Preview," or if nothing was found during the preview operation. Click on this box with the right mouse button to bring up the context menu.

Sometimes, there is text information ahead of the binary data in an encoded file. If a file in the preview list has such text, a small purple box will appear next to its status indicator. Clicking the "Info" button will display the "ahead of data" text for any file that has such text. For details, see Decode - View File Info.

To change the name of a file in the preview list (before it has been decoded), highlight it, then click this button. UUDeview will allow you to set a new name.

Click the "Remove" button to delete a file from the preview list. This will prevent the selected file from being decoded. Note that no actual existing files are affected by this command.

This is the file that will be launched.

The application that will be used to launch the indicated file. You can use the default (whatever Windows has associated with the given extension); type the complete path of an application; click the "Browse" button and locate the desired program; or use the pulldown list to select an application that has been recently used in the Launch box.

Click this button to browse your disk drives and choose the application UUDeview will use to launch the indicated file.

Displays this help topic.

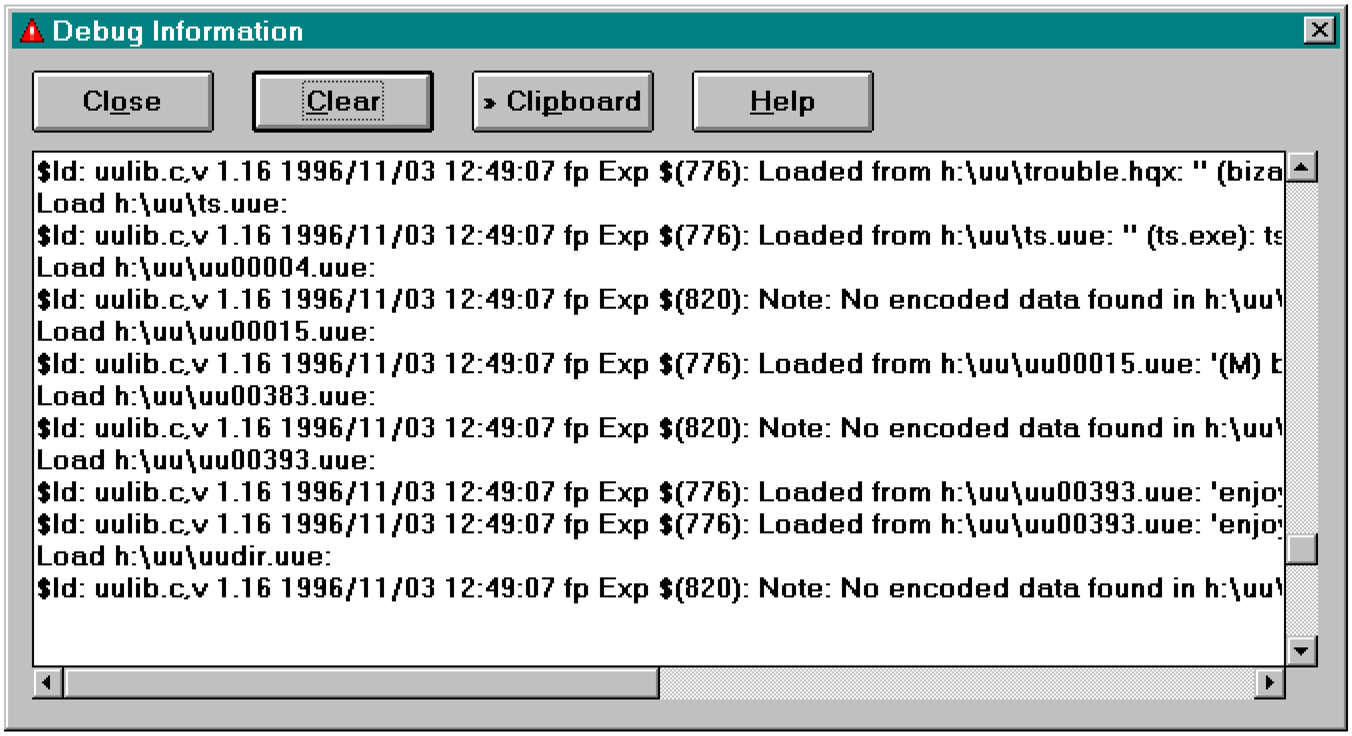
Launches the "Using" application for this file only.

Launches the "Using" application for this file and all selected files with the same extension. For this example, clicking "OK to All" would use the indicated application to launch all selected files with an `.HQX` extension.

Cancels the Launch operation.

Decode - Debug Info

If you turn on "Show Debug Info" in the Decode Options, UUDeview will display a lot of information about what it is doing during the decoding process. For example, if you were to turn on "Show Debug Info" and then click the "Preview" button, a dialog box like this might appear:



```
$Id: uulib.c,v 1.16 1996/11/03 12:49:07 fp Exp $(776): Loaded from h:\uu\trouble.hqx: '' (biza
Load h:\uu\ts.uue:
$Id: uulib.c,v 1.16 1996/11/03 12:49:07 fp Exp $(776): Loaded from h:\uu\ts.uue: '' (ts.exe): ts
Load h:\uu\uu00004.uue:
$Id: uulib.c,v 1.16 1996/11/03 12:49:07 fp Exp $(820): Note: No encoded data found in h:\uu\
Load h:\uu\uu00015.uue:
$Id: uulib.c,v 1.16 1996/11/03 12:49:07 fp Exp $(776): Loaded from h:\uu\uu00015.uue: '(M) b
Load h:\uu\uu00383.uue:
$Id: uulib.c,v 1.16 1996/11/03 12:49:07 fp Exp $(820): Note: No encoded data found in h:\uu\
Load h:\uu\uu00393.uue:
$Id: uulib.c,v 1.16 1996/11/03 12:49:07 fp Exp $(776): Loaded from h:\uu\uu00393.uue: 'enjo
$Id: uulib.c,v 1.16 1996/11/03 12:49:07 fp Exp $(776): Loaded from h:\uu\uu00393.uue: 'enjo
Load h:\uu\uudir.uue:
$Id: uulib.c,v 1.16 1996/11/03 12:49:07 fp Exp $(820): Note: No encoded data found in h:\uu\
```

The information in this box is not necessarily meant to be terribly comprehensible. It's mostly intended to help you see where problems are occurring when you have trouble decoding a file.

If you can't decode a file that you *really need* to decode, you can contact UUDeview's authors. In this case, the Debug Info box can supply data that will be vital in helping us figure out why a file won't decode. You can use the "Clipboard" button to copy the current contents of the Debug Info box to the Windows Clipboard, then paste this into the e-mail message you use to report your problem.

The Debug Info box has four buttons:

» **Close.** Closes the Debug Info box.

» **Clear.** Erases any text currently in the Debug Info box. You might want to do this before undertaking another UUDeview operation.

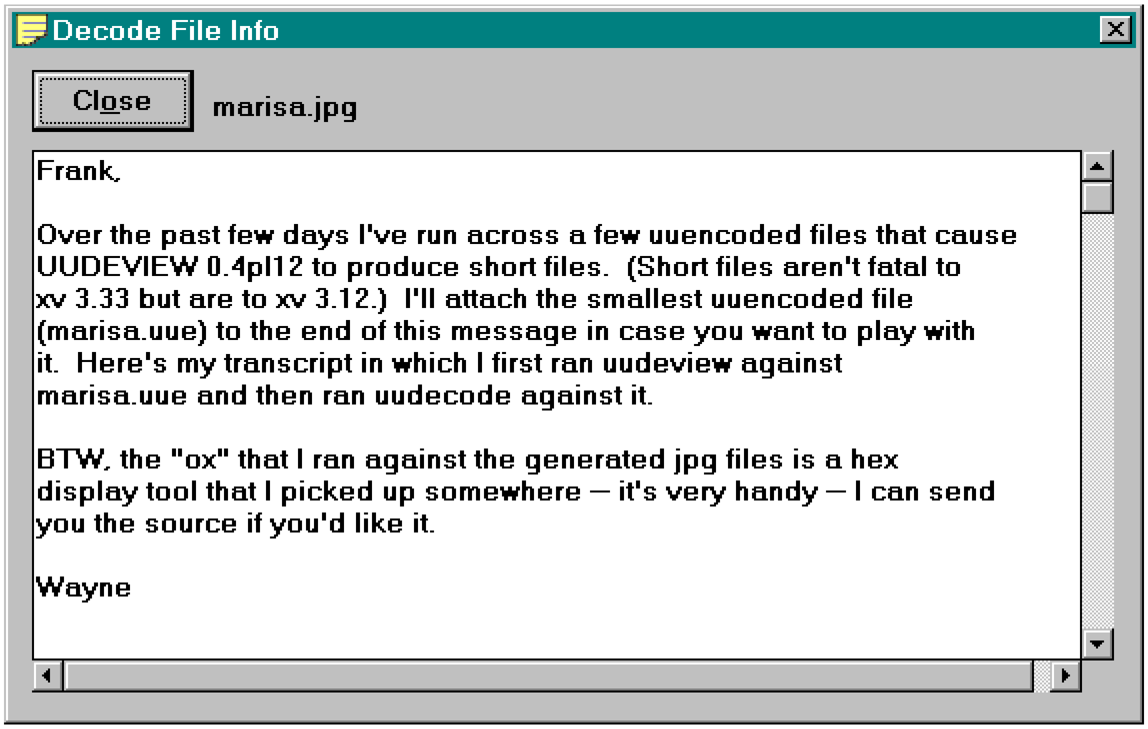
» **Clipboard.** Sends the current contents of the Debug Info box to the Windows Clipboard. You can then paste this into a file or an e-mail message if desired.

» **Help.** Displays this help information.

Decode - View File Info

The Decode File Info box appears when you click on a file in the Decode Preview list and then click the "Info" button (or select the "Info" command from the context menu).

Here's a sample Decode File Info display:



Frank,

Over the past few days I've run across a few uuencoded files that cause UUDEVIEW 0.4pl12 to produce short files. (Short files aren't fatal to xv 3.33 but are to xv 3.12.) I'll attach the smallest uuencoded file (marisa.uue) to the end of this message in case you want to play with it. Here's my transcript in which I first ran uudeview against marisa.uue and then ran uudecode against it.

BTW, the "ox" that I ran against the generated jpg files is a hex display tool that I picked up somewhere -- it's very handy -- I can send you the source if you'd like it.

Wayne

In this case, a UUDeview user sent the program's authors a file that wouldn't decode properly. He added a description of the problem ahead of the actual encoded data. The Info command displays this sort of information.

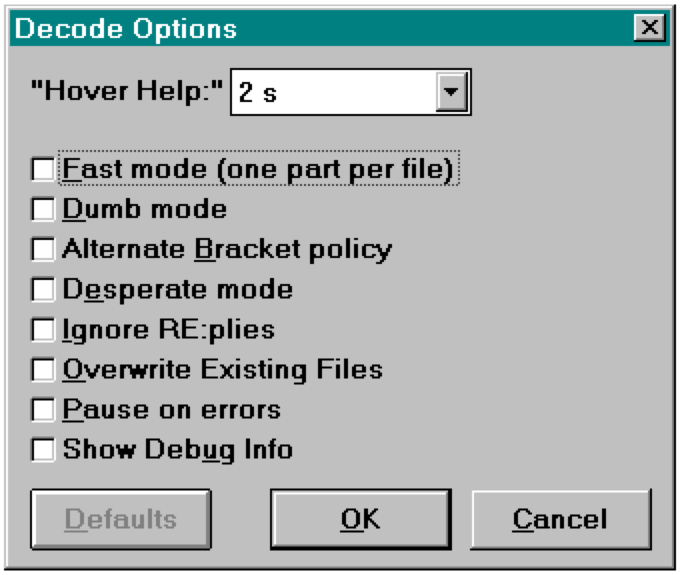
In essence, that's what the Decode File Info box does: it displays text that appears before the encoded data in an encoded file. When sending an encoded file, many people write a description ahead of the "gibberish" that makes up the encoded data. Using the Info button to look at this information can tell you what's in a file and whether or not you really want to decode it.

When UUDeview finds text ahead of encoded data in a file, it adds a small purple box to the file's status indicator in the Decode Preview (right hand) list box.

The name of the file that will be decoded appears above the Info text box. The "Close" button will close the Decode File Info box and return you to the main UUDeview window.

Decoding Options

UUDeview provides a number of options to alter how it processes your encoded files. Most of the time, the default options should be fine. In the event you need the options, here they are:



Option	Meaning
Hover Help	When this option is enabled, a tiny descriptive window will appear when you let the mouse cursor "hover" above a button or field on the UUDeview window. The "Hover Help" setting determines how long the mouse cursor must hover before the information window pops up. The default is one second (1 s). Select "off" to disable the "Hover Help."
Fast mode	If set to 1, the library will assume that each input file consists of exactly one e-mail message or newsgroup posting. After finding encoded data within a file, the scanner will not continue to look for more data below. This strategy can save a lot of time, but has the drawback that files also cannot be checked for completeness-since the scanner does not look for "end" lines, it won't notice them missing. This flag does not have any effect on MIME multi-part messages, which are always scanned to the end (alas, the Epilogue will be skipped). Actually, with this flag set, the scanner becomes more MIME-compliant.
Dumb mode	UUDeview evaluates information found in each part's "Subject" header line if available. The heuristics here are versatile, but cannot be guaranteed to be completely failure-proof. If false information is derived, the parts will be ordered and grouped

	wrong, resulting in wrong decoding.
	If "Dumb" mode is set, the code that derives part numbers is disabled; it will then be assumed that all parts within a group appear in correct order: the first one is assigned number 1 etc. Part numbers found in MIME headers are still used.
Alternate Bracket Policy	Multi-part postings on Usenet usually have subject lines like "You must see this! [1/3] (2/4)". How to parse this information? Is this the second part of four in a series of three postings, or is it the first of three parts and the second in a series of four postings? The library cannot know, and simply gives numbers in () parentheses precedence over number in [] brackets. If this assumption fails, the parts will be grouped and ordered completely wrong.
	Setting the "bracket policy" option changes this precedence. If now both parentheses and brackets are present, the numbers within brackets will be evaluated first.
Desperate Mode	By default, UUDeview refuses to decode incomplete files and generates errors. But if switched into "desperate mode" these kinds of errors are ignored, and all available data is decoded. The usefulness of the resulting corrupt file depends on the type of the file.
Ignore Replies	If set, UUDeview will ignore e-mail messages and news postings which were sent as "Reply," since they are less likely to feature useful data.
Overwrite Existing Files	By default, UUDeview will warn you before overwriting any existing file. You can silence these messages by setting this option.
Pause on Errors	Normally, UUDeview will decode all files without stopping to display error messages. The status markers in the output file list will show which decode operations succeeded and which failed.
	If you set this option, the program will stop and display a detailed message every time it encounters an error.
Show Debug Info	Turning this option on will cause UUDeview to display detailed information about what is happening during the preview and decoding process. This information, which tends to be quite cryptic, is likely to be of interest only when you are having problems decoding a file. For details, see Decode - Debug Info.

» If any non-default options are set, the word "Options" on the main UUDeview form Options button will appear in *italics*.

» Options are saved from session to session.

» The **Defaults** button resets the default decoding options recommended by

UUDeview's creators.

» **Important.** Option settings affect the *next* UUDeview operation, not any that have already been completed.

Resets the default decoding options recommended by UUDeview's creators.

Encoding

To encode a binary file for transmission, you'll need the Encode mode. There are four steps to encoding: selecting Encode mode, selecting the file to encode, setting the destination path, and performing the encode.

Quick Summary

If you want to get started in a hurry, just follow these steps:

» Click "Encode" in the "Function" box to switch to Encoding mode.

» Type the name of the file you want to encode in the "Encode File" box, or click "Browse" and navigate to it. You can also drag a file to encode from Explorer or File Manager and drop it on the UUDeview window.

» Set the encoding mode that you wish to use in the "Encoding Scheme" box.

» In the "Multipart" box, choose either "single file" or "Break every *n* lines" as appropriate

» Set the destination path and filename in the "To" box, or click the "Path" button to browse for the destination directory.

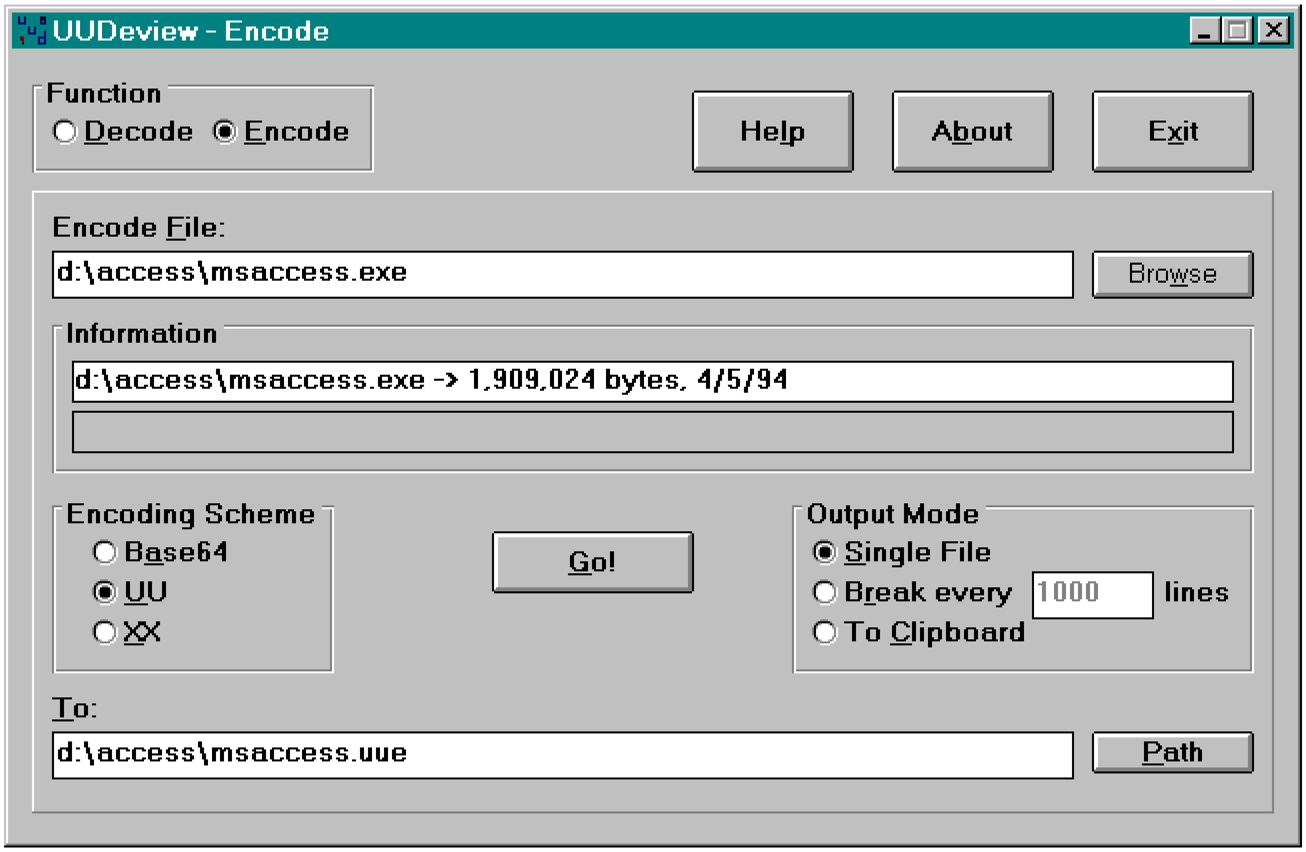
» Click the "Go!" button to encode your file.

To learn more, consult the Encoding Details section.

Encoding - Details

The UUDeview encoding window looks something like this:

(Click on the image below to get information on the controls)



Function: Decode [✓ Encode]

Encode File: d:\access\msaccess.exe

Information: d:\access\msaccess.exe -> 1,909,024 bytes, 4/5/94

Encoding Scheme: Base64 [✓ UU] XX

Output Mode: [✓ Single File] Break every 1000 lines To Clipboard

To: d:\access\msaccess.uue

- The "Encode File" box contains, fairly obviously, the name of the file you want to encode.
- The "Information" box tells you about the file you've chosen to encode, its size, creation date, etc. It will also display progress information during the encoding process.
- The "Encoding Scheme" box determines which format will be used to encode your data. UU is the safest and most popular.
- The "Output Mode" box allows you to break the encoded file into sections if that is necessary. If "Break every *n* lines" is selected, UUDeview will start a new output file every *n* lines. The file extension will indicate the part numbers: .001, .002, and so on.

  If you select "To Clipboard," UUDeview will encode the selected file directly to the Windows Clipboard in a single part. You can then paste the encoded text into another program, e.g., an Internet Mail client.
- UUDeview will warn you before overwriting any existing files during the encoding process.

This field contains the name of the binary file that UUDeview is to encode. You can type the name of the file you want to encode in this box, or click "Browse" and navigate to it. You can also drag a file to encode from Explorer or File Manager and drop it on the UUDeview window.

Click here to browse your system's disk drives for the file you want to encode.

This box shows information about the currently-selected input file. It also indicates progress during lengthy operations.

Base64 is the encoding format used in MIME-formatted messages. It has low overhead and transfers successfully across many different platforms, but is unfortunately not supported by many mail and news clients. For safety, it's best to use UU encoding in most cases.

UU-encoding is the most common binary encoding format. It's accepted and understood by a wide variety of client programs.

XX-encoding is rarely used. UUDeview offers it only for compatibility.

The "Go" button starts the encoding process, once you've set the input and output filenames.

When the Output Mode is set to "Single File," UUDeview will encode the designated binary file to a single file.

Setting the Output Mode to "Break every *n* lines" causes UUDeview to start a new output file every *n* lines. The files are named .001, .002, and so on.

The "To Clipboard" mode tells UUDeview to encode the designated file and place the result on the Windows Clipboard so you can paste the encoded data directly into another application.

This field indicates the name UUDeview will send the encoded output to. Note that this box is disabled in the "To Clipboard" Output Mode; in that case, no output file is created.

Click this button to choose the directory where UUDeview will write the encoded file.