Base64Coders

Edmund Vermeulen

COLLABORATORS			
	TITLE : Base64Coders		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Edmund Vermeulen	March 1, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Base64Coders

1.1 Contents

This is the AmigaGuide® documentation for:

Base64Decode & Base64Encode 1.3

Author: Edmund Vermeulen
  Primulastraat 2
  3202 RN Spijkenisse

The Netherlands
Tel. 0181-613925

Fidonet: 2:286/407.48
Internet: edmundv@grafix.xs4all.nl

Please report any bugs, problems or suggestions to me.

1.2 description

Description

Base64Decode & Base64Encode are a fast decoder and encoder for base64 (MIME) encoded binaries that can be sent through electronic mail.

Base64 encoding is very much like uuencoding, only it is part of the much more comprehensive MIME format. MIME allows you to send pictures, sound and other binary files via e-mail. If you do not use a MIME aware mail reader then you will not see the effect of this, but at least you can now decode the included binaries thanks to Base64Decode! Simply save the base64 encoded part as a text file and have Base64Decode decode it for you. If you are a Spot user then you can use the included ARexx script for this.

Features

- Simple and easy to use Shell commands.
- Written in 100% assembly for optimal speed and size.
- Faster than any other base64 decoder/encoder for the Amiga known to me.
- Separate (faster) '020 versions included.
- Decoder is not so fussy if the base64 code is embedded in some text. This is not fool proof though (it can't be). By using the USEMINLEN option this is greatly improved.
- Decoder can extract the filename from the MIME header.
- AUTONAME option for when the decoder can't find a filename in the MIME header.
- User settable I/O buffer size.
- The code is pure and can be made resident.
- ARexx script for Spot included.

Requirements

Base64Decode & Base64Encode require AmigaOS Release 2 or higher to operate.

The '020 versions require a 68020 or higher CPU.

1.3 usage

Usage

Base64Decode and Base64Encode are Shell commands. To use them you may copy them to your 'C:' directory and type their name from the Shell. You may also use them from the Workbench 'Execute Command' menu.

1.4 base64decode usage

Base64Decode usage

Base64Decode requires at least a filename for the FROM argument. The TO argument can be either a filename or a directory. A BUFSIZE number and the USEMINLEN and AUTONAME switches are optional. You can see the template if you type a questionmark.

```
 > Base64Decode ?
 FROM/A,TO,BUFSIZE/N,USEMINLEN/S,AUTONAME/S:
```

Example usage:

If the TO argument is a filename:

```
 > Base64Decode Work:ModemStuff/Base64Text RAM:Test
 33729 bytes written to "RAM:Test"
```

If the TO argument is a directory and there is a filename present in the MIME header:

```
 > Base64Decode Work:ModemStuff/MimeMessage RAM:
 11790 bytes written to "RAM:mods.lha"
```

If there is no filename present in the MIME header:

```
 > Base64Decode Work:ModemStuff/MimeMessage RAM: AUTONAME
 No filename found in MIME header
 11790 bytes written to "RAM:MimeMessage.bin"
```

The FROM argument

The FROM file should be the text file containing the base64 encoded data. If you want to be 100% sure then you should only save the actual encoded part. It looks something like this:

```
V2hhdD8gWW91IGFjdHVhbGx5IHRyaWVkIHRvIGRlY29kZSB0aGlzIGV4YW1wbGUgYml0PyBTaWxs
eSB5b3UhCg==
```

Base64Decode does try to be smart if there is some text before, after or in between the encoded data. However, this can never be fool proof. Especially lines that contain a single word with a character length that is an exact multiple of four will foul up decoding. You can use the USEMINLEN option to combat this problem.

Please note that Base64Decode will decode all base64 code to a single file. If there is more than one file encoded in the same message then you should save the part that you want to decode to a separate text file.

The TO argument

If you specify a filename for the TO argument, Base64Decode will use that name for the decoded file.

If you specify a directory, Base64Decode will try to extract the filename from the MIME header in the message body. The MIME header should look something like this:

```
  Content-Type: application/octet-stream; name="mods.lha"
  Content-Transfer-Encoding: base64
```

If Base64Decode cannot find a filename, you will be presented with an error, unless you specified the AUTONAME option.

If no TO argument is specified then the current directory will be used as destination.

The BUFSIZE argument

The BUFSIZE argument specifies how many bytes of RAM will be used for the I/O buffer. The default is 32768 bytes. Multiples of 8K (8192, 16384, 24576, 32768, etc.) work best. Generally it is true that the larger the buffer, the faster decoding will be. The smallest buffer size that you are allowed to set is 4096 bytes.

The USEMINLEN argument

When the base64 code is embedded in some text, part of that text can sometimes be seen as base64 code. By specifying the USEMINLEN option Base64Decode will only start decoding at a minimum line length of 60 base64 characters. This way, a single word on a line with a character length smaller than 60 (that is also an exact multiple of four) will not be decoded. This will catch all but the longest German words. :-)

The AUTONAME argument

Normally, Base64Decode will fail when you decode a MIME message to a directory and no filename can be found in the MIME header. You can get around this problem by specifying the AUTONAME option. In that case a filename will automatically be created by appending '.bin' to the FROM filename.

1.5 base64encode usage

Base64Encode usage

Base64Encode requires two arguments. A FROM filename and a TO filename. A BUFSIZE number and NOHEADER switch are optional. You can see the template if you type a questionmark.

```
 > Base64Encode ?
 FROM/A,TO/A,BUFSIZE/N,NOHEADER/S:
```

Example usage:

```
 > Base64Encode Work:ModemStuff/MyArchive.lha RAM:Test
 50544 bytes written
```

The FROM argument

The FROM filename is the binary file that will be encoded using the base64 algorithm.

The TO argument

The TO filename is the name of the text file that will be created containing the FROM file in base64 encoded form.

The BUFSIZE argument

The BUFSIZE argument specifies how many bytes of RAM will be used for the I/O buffer. The default is 32768 bytes. Multiples of 8K (8192, 16384, 24576, 32768, etc.) work best. Generally it is true that the larger the buffer, the faster encoding will be. The smallest buffer size that you are allowed to set is 4096 bytes.

The NOHEADER argument

Base64Encode normally generates a MIME header for the base64 encoded data. To turn this behaviour off, you may specify the NOHEADER argument.

Please note that Base64Encode will create a fully MIME conformant mail message when the header isn't switched off. You cannot simply add some text to this message. If you do, the message won't be MIME compliant any more and it won't be recognized properly by a MIME aware mail reader. It is best to

send the generated base64 encoded file as a separate message.

1.6 usage from directory opus

Usage from Directory Opus

Many people will want to decode files from within a directory utility such as Directory Opus. This is how I have it set up myself.

```
Function : AmigaDOS   Base64Decode {f} USEMINLEN AUTONAME

   Flags : Output to window
           CD destination
           Do all files
           Rescan dest
           Window close button
```

Calling this function will result in all selected files being decoded to the destination directory.

1.7 spot script

Base64Decode.spot

Base64Decode.spot is an ARexx script for Spot (the Fido offline mail reader by Nico François) that allows you to decode a base64 encoded file from the current message. It uses Base64Decode for that purpose; which must be installed.

You must be using a registered version of Spot; otherwise ARexx is not available. To be able to use it, you should copy the 'Base64Decode.spot' file to Spot's ARexx directory (normally 'REXX:Spot'). You may then install it in the ARexx menu by using the ARexx settings editor within Spot. You should set the output for the script to 'NIL:'.

To use it, simply call it from the message window when you want to decode a base64 encoded binary from the current message. You will be prompted with a file requester to select the destination path for the decoded file. If Base64Decode cannot find a filename in the MIME header, it will ask you for one via a file requester.

1.8 technical info

Technical info

Base64 encoding was designed to be able to send binary files across systems that only support pure text (ASCII) messages. To do so it uses its own base64 alphabet which looks like this:

```
  ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

A binary file is encoded in groups of three bytes (24 bits). These 24 bits
are then separated into four 6-bit groups. Each 6-bit value is represented
by the corresponding character from the base64 alphabet (0='A', 63='/').

Thus, every three bytes in the original file are represented by four
characters in the encoded file (resulting in a 33% size increase). If a file
is not an exact multiple of three bytes then the remaining quartet of base64
characters will be padded with one or two '=' characters.

The maximum line length for base64 encoding is 76 characters. Shorter lines
are possible but their length is always an exact multiple of four.

1.9 acknowledgments

Acknowledgments

I'd like to thank John Hendrikx for answering my questions and for
suggesting some optimizations. Thanks, John!

Thanks to Arthur Pijpers for testing on his A4000/040.

1.10 to do

To do

- Decode multiple files from one message.

1.11 release history

Release history

Version 1.3 released 29-Nov-1996

- Base64Decode now ignores stuff after the filename. E.g. in the line

  ```
  [...] name=screens.lha; type=LHArc-archive
  ```

  the '; type=LHArc-archive' bit was seen as part of the filename. Now
  everything after the ';' character is ignored.

Version 1.2 released 27-Jun-1996

- Base64Decode is more flexible at finding filenames in the MIME header.

- The TO argument is not required anymore in Base64Decode. If none is
  specified, the current directory will be used as destination.

- Added AUTONAME argument. If you specify this on the command line and
  Base64Decode cannot find a filename in the MIME header, a filename will

automatically be created by appending '.bin' to the FROM filename.

Version 1.1 released 11-Mar-1996

- Base64Decode is about 10% faster. This means that it is just as fast as uuOut (by Nicolas Dade) is with decoding uuencoded files. This was a goal of mine. :-)

- Fixed bug in non-base64 code skipping routine in Base64Decode.

- Changed MIME header generated by Base64Encode. The name of the encoded file is put on the Content-Type line.

- Base64Decode will try to extract the TO filename from the MIME header in the message body if you only specify a directory for the TO argument.

- Included Base64Decode.spot. An ARexx script for Spot to decode a base64 encoded binary from the current message.

Version 1.0 released 25-Dec-1995

- First public release.