0032a1c8-0

Nicolas Dade

COLLABORATORS

	TITLE : 0032a1c8-0	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Nicolas Dade	August 9, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

0032a1c8-0

1.1 uuOut

```
uuOut v1.14 © 1993-1995 by Nicolas Dade


   Copyrights~and~Distribution


         --Introduction--

  What~is~it?
  Why~should~I~use~this~one?
  Requirements.
  Reaching~the~author.


          --Usage--
         (CLI only!)

  Description~of~Options.
  Examples~of~Use.
  Details~of~the~Sanity~Checking,~for~those~who~have~to~know.
  FAQ.


          --Misc--

  Thanks~to...
  Benchmarks.~~How~fast~is~it?
  History.~~What~was~wrong~with~the~previous~version?
```

1.2 Copyright and Distribution

This is not shareware.

With~that~out~of~the~way,~what~is~it?

1.3 Quick Introduction

uuOut is a uudecoder to decode uuencoded text (an encoding scheme that is used to send binary files over links that only support textual characters, like email and usenet news)

What~makes~it~so~special?

1.4 Why this one?

--Why is this decoder better than those already available?--

* It's~very~fast~at~decoding (and it uses a different, even faster decode routine if a 68020 or higher is detected).
  On top of that it uses a simplification of the Boyer-Moore search algorithm when searching for 'begin' lines, so scanning files that contain uuencoded data interspersed with lots of non-data is also very rapid.
  (Of course, it's 100% assembly)

* It~sanity~checks~its~input so that non-uuencoded data lines are skipped over instead of being "decoded" and producing garbage. Sanity checking includes checking trailing checksums, M's and X's, and the 'size' line. Some sanity checking can be disabled in case compatibility with the original uudecode is desired, using the IGNORETERMINATION option.

* It allows you to specify the output directory in which to place the decoded file(s) so that they don't have to go to the current directory, and paths in the encoded filenames can be ignored or obeyed (any needed directories are created automatically)

* Output files have their protection flags set to reflect the owner's rwx unix protection bits specified in the uuencoded data.

* It will take its input from its standard input, as well as from a file.

* It decodes all files encoded in its input, not just the first one.

* Partially complete files (missing data lines or missing 'end' lines) don't mess up the other, complete, files.

* It does its own IO buffering using user sizeable buffers, so your drive won't be slowed by thrashing if uuOut's input and output come from and go to that same drive.

* It's pure and it's small---you can make it resident and it
  won't cost you over 5K

* uuOut reports all errors and strange conditions with descriptive
  error messages, and aborts if it is sent a break signal
  (Control-C).

        What~is~required~to~use~uuOut?

1.5 Requirements

WorkBench 2.04 or higher is all that is required.

        How~do~I~use~it?

1.6 Usage

 the template for uuOut is: INFILE,OUTPATH,BUFSIZE/K/N,VERBOSE/S

INFILE....the file from which to read the uuencoded data. If no
          INFILE is specified, standard input is used.

OUTPATH...a directory from which the uudecoded files's location is
          based. Usually that means that the output files go in
          OUTPATH, but if the output file's name contains a path
          along with a filename (i.e. "dir1/dir2/filename") then
          the uudecoded file will appear in OUTPATH/dir1/dir2/filename.
          (the directories "dir1" and "dir2" are created if they do
          not already exist)
          If OUTPATH is not specified the current-directory is used
          in its place.
          If this directory does not exist it too will be created.

BUFSIZE...allows you to specify the size of the input and
          output buffers uuOut should use in kilobytes. The default
          is 32K which is a little small for a harddrive to
          harddrive decode. Try 150K or higher, until the accesses
          to the harddrive become a second or so apart. If there
          isn't sufficient memory available to allocate buffers of
          the specified size, uuOut keeps trying small and smaller
          sizes (half as large each time) until the buffer would be
          smaller than 1K, at which point uuOut gives up and aborts
          with an error message explaining that there isn't enough
          memory in the system for the buffers. Note that if you
          don't have 2K free you've got more problems on your hands
          than uuOut refusing to run. :)

USEBASENAME...instead of (if need be) creating directories and placing
          the uudecoded files in those directories, all uudecoded files
          are created in the current-directory (or OUTPATH if specified).
          i.e. "begin 644 dir1/dir2/file" would be decoded into

```
          <current-dir>/file or OUTPATH/file, instead of
          <current-dir>/dir1/dir2/file or OUTPATH/dir1/dir2/file.

IGNORETERMINATION...ignores all data from the last of the data characters
          to the EOL of each line.
          This option is available so that uuOut can be forced to decode
          all sorts of uuencoded data. The original un*x uudecode ignored
          everything between the data bytes and the end of the dataline (LF
          or CR). Later modifications added all sorts of terminating
          characters. uuOut understands some of these, but evidently there
          are some terminations out there that uuOut does not expect.
          This condition is evidenced most often by empty output files,
          or lots of bad checksum error messages. If you see this happening,
          I suggest you pass the data through uuOut a second time with
          IGNORETERMINATION enabled before declaring the data lost.

VERBOSE...uuOut reports each step it takes to standard-error.


        May~I~see~some~examples?
```

1.7 Examples

```
 To decode any uuencoded files in the file "t:FreshEmail.txt" and create the
paths+files starting from the "work:" directory

  uuOut t:FreshEmail.txt work:

and to use 150K io buffers instead of the default 32K when doing this

  uuOut t:FreshEmail.txt work: BUFSIZE=150

and to have all the files go in the work: directory and not some
sub-directories of work:

  uuOut t:FreshEmail.txt work: BUFSIZE=150 USEBASENAME

To decode the standard input stream and create the output files in the current
directory

  uuOut

(kind of easy, no?) And to decode the standard input stream and create the
output files in "t:"

  uuOut OUTPATH=t:

just saying "uuOut t:" will not work because t: will be interpreted to be the
name of the input file, and uuOut will then try to open and read from the
"file" t:, find that it cannot, and abort with an error message.

Decoding data that is stored in multiple files can be done with the help of
the C= join command: (BTW if you're just going to feed the data to uuOut then
don't bother with multiple files, they just slow things down---instead create
one giant file and feed it to uuOut)
```

```
  run join file1 file2 file3 ... to pipe:file
  uuOut pipe:file
```

or, if you have set the _pchar local variable to | and have the pipe command installed, then it is easier to use

```
  join file1 file2 file3 ... to out: | uuOut
```

(yes, the C= shell can do anonymous piping, it's just not documented.) Or if you have csh installed,

```
  cat file1 file2 file3 ... | uuOut
```

And as usual, to see the template

```
 uuOut ?
```

and to see the full built-in help and copyright text, give a second "?" at the prompt following the template.

```
        I~like~it.~To~whom~do~I~owe~this~great~program?
```

1.8 Reaching the Author

If you would like to talk to me about uuOut for any reason, here's were you can reach me:

```
        on the internet: nicolas-dade@uiuc.edu

        at the post office:
                 Nicolas Dade
                 405 W Delaware
                 Urbana, IL 61801  (USA)
```

1.9 Description of Sanity-Checks

uuOut's input-checking keeps it from trying to decode lines that obviously aren't data lines, and it's difficult to fool, unlike a certain other uudecoder that advertises this feature. uuOut insists a data line contain only the characters 0x20 (space) through 0x40 (backquote) inclusive (0x40 is considered equivalent to 0x20), and have the right number of characters (the first byte of a data line is a count). Furthermore the data line must end with (optionally) a checksum character, (optionally) an X or M, and then either a LF or a CR, and the checksum, if it is present, must be correct. (If a line looks good except that it's checksum is bad, uuOut ignores the data in the line, but warns you that a line that looked like data but had a bad checksum is being skipped.)

This means that mail and news header lines aren't considered data because the header names contain lowercase characters or, if they don't, rarely (that's

the .1% of the time) contain the right number of characters. You _can_ 99.9% of the time pass mail or news right through uuOut without stripping off anything. Furthermore uuOut does not give up after decoding one file, but rather continues looking for and decoding any more uuencoded files in the text.

On top of that, if there is a 'size' line after the 'end' line, uuOut will check to see if the file size specified in the 'size' line matches the actual size of the file just created, and warn you if it does not. This protects you against missing parts of a multi-part file, and the very rare lines that fool uuOut's line-by-line input checking. If you think the data is good, then first try editing the file containing the uuencoded data so that it contains only the 'begin' line, the data, and the 'end' and 'size' lines. Pass this through uuOut. If uuOut says the size line is still wrong then the data probably is bad, or part of the data is missing.

1.10 Frequently Ask Question

There's only one question I've been asked more than once, so here it is:

Q: Why does my shell not come back after I look at the template or the help text?

A: Because if you just press <return> at the template prompt uuOut thinks it's in the "decode standard input" case, and whatever you type after that goes into uuOut, not your shell. To get out of it you have to send uuOut an end-of-file, which is control-\ on the standard amiga keymaps. uuOut will exit, but it will also warn you that no 'begin' line was found in its input file (or, if you typed 'begin nnn f*', that no 'end' line was found.) (Unless, of course, you then also typed 'end' :)

1.11 Benchmarking this Baby

--Speed Comparisons--:

On a 7.14MHz NTSC 68000 based Amiga (ie A500, A1000 or A2000) using the command line "uuOut ram:440KByteFile ram:", decoding the 440Kbyte file which contains a 310 KByte uuencoded file:

using:	version:	takes:	speed:	(relative to slowest)
uuOut	1.14	5.7 secs	77Kb/s	(13.0 $\times$ faster)
UUDecodeX	1.02	27.7 secs	16Kb/s	(2.7)
UUxT	3.0b	33.6 secs	13Kb/s	(2.2)
uudecode**	?	64.3 secs	7Kb/s	(1.1)
uudecode***	36.6	74.0 secs	6Kb/s	(1.0)

** by Mark Horton and Alan J. Rosenthal. I couldn't find any version information.

** from the uuCode archive, by Ralph Seichter.

1.12 People who helped make uuOut what it is today

```
 These people sent me their suggestions and bug reports:

        Alan T Shutko
        Marcus Damberger
        David Zaroski
        Olivier Jeannet
        Timothy Needham

And I did the programing. :)
```

1.13 History of uuOut, from 1.00 to the present

```
 1.00  o First version debugged enough to pass around 1.01
       o Changed verbose output to go to standard-error if one is
         defined instead of standard-out. Standard-out is still
         used as a fallback. (pr_CES instead of pr_COS)

1.02  o Since Alan T Shutko so kindly sent me the code for the checksum,
        I added checksum checking.
      o Added size line check.
      o Fixed bug where a begin line that was loaded at the end of the
        buffer was passed over. This showed up easily when you typed
        the 'begin' line into a console window yourself.
      o And fixed the 'we're stuck in a loop' code. Now lines like
        "'" will not trigger it. Actually the bug was more complicated
        and more general than this. There were a couple places where
        the logic was screwy and a the 'infinite loop' condition could
        be detected when it wasn't the case.
      o Added checking for early 'begin' lines. Now uuOut checks for a
        'begin' line even if the previous file's 'end' line has not yet
        been found. When one is found a warning is, of course, issued.
      o Thanks to Marcus Damberger for testing uuOut many times on his
        030 machine.
      o Added USEBASENAME and the automatic directory creation due to
        David Zaroski's email/suggestion. (but not after a fight :)
      o Changed, again because David Zaroski pointed it out to me,
        the 'begin' line detection code. begin lines now must match:
        begin<sp>#<sp>#[0-7][0-7][0-7][0-7]<sp>#<sp>(~<sp>)*<EOL>
      o Added check for 'M' terminated lines, and IGNORETERMINATION
        because David Zaroski managed to find a uuencoded file with
        each line terminated with an 'M'!

1.03  o No changes, but I upped the version number so that it matched
        that of the uuIn and uuPrepare with which it is going to be
        distributed. Also they are too many betas of v1.02 floating
        around. Going to 1.03 makes all those people think that they
        should upgrade.
      o Thanks to Olivier Jeannet for testing the final version.

1.4-1.14
      o Changed version to be 1.x and not 1.0x to conform with C=
        Also I changed to my automatic revision system, which means
```

that revision will be bumped every time I make a new comment
or a new backup.
- o Tweaked the 68020+ uuOut code just a little bit.
- o Fixed CR-LF problem. Due to a (obscure and amusing) bug in
the "commit data line as being valid if the termination looks
good" code, lines ending in just a CR+LF were considered bad.
This means that files coded by, for example, WinCode, that
still had their original CR's came out as 0 length files.
Thanks to Timothy Needham for emailing me a 200K "bad" uu data
file which he had to upload at 2400 baud.
- o Sped up the "skip over rest of bad data line" code, and fixed
an obscure bug there involving buffers larger than 64K and
the bad data line spanning a 64K buffer boundary.
- o Made the handling of good lines with bad checksums more reasonable.
Now only the first few are shown, and a total count is given at
the end of the file.
- o Added suggestion to use IGNORETERMINATION when a zero-length file
is written.
- o Added call to Flush() so now appearance of the VERBOSE progress
messages is synchronized with what is actually going on.
- o Fixed bug where the data line immeadiately following a good line
with bad checksum was sometimes skipped over.