UUArc

COLLABORATORS			
	TITLE : UUArc		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 5, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

1 UUArc **1**

1.1 Index To UUArc Guide . . . 1
1.2 UUArc History . . . 1
1.3 introduction . . . 2
1.4 Technical Information . . . 3
1.5 acknowledgements . . . 4
1.6 bibliography . . . 5
1.7 Reason For Writing This Program . . . 6
1.8 Detailed Description Of The Program . . . 6
1.9 Details Of Installation Of UUArc . . . 7
1.10 Bug Reports, Suggestions etc. . . . 7
1.11 North 'C' V1.3 . . . 8
1.12 a68k . . . 8
1.13 blink . . . 8
1.14 Public Domain, Shareware, Freeware etc. Software . . . 8
1.15 UUE Encoding Decoding Algorithms . . . 9
1.16 ASCII . . . 9
1.17 Bytes, Binary and Bits! . . . 9
1.18 GuiArc . . . 10
1.19 arctypes . . . 10
1.20 Crohn's In Childhood Research Association . . . 10
1.21 Quick Usage Guide for UUArc . . . 11
1.22 Electronic Mail (E-Mail) . . . 12

Chapter 1

UUArc

1.1 Index To UUArc Guide

```
                    UUARC Version 1.3 04/10/93
              J.G.BRANDON

               FREEWARE
```

Introduction

Reason For Writing The Program

Detailed Description of Program

Installation Procedure

Technical Reference Information

History

Bug Reports and Future Updates

Acknowledgements

Bibliography

Quick Usage Reference

1.2 UUArc History

```
V1.3 - 04/10/93
  Cleaned up source to be more ANSI 'C' compatible and added function
  prototypes - in particular the function type of signal handler for
  signal exception processing, and corrected a checksum digit type bug (the
  checksum digit is now an unsigned int rather than a normal int.)
```

```
  Move command dropped, replaced with a generic move option/flag.  Added
  support for processing of more than one filename at a time.

V1.2 - 29/09/93
  Stripped the guide down.  Compatible with more UUEncoders - some naughty
  encoder programs out in the world don't convert ASCII 32 'Space' characters
  to ASCII 96 (`) characters, UUArc should now decode such encoded files
  successfully.

V1.1 - 08/07/93
  Recognises more signals to abort.

V1.0 - 07/07/93
  First version completed and released.
```

1.3 introduction

```
                UUArc is an archiving system designed to enable easy transmission ↩
                   of

                binary
                 files/archives over communcation links only capable of using

                ASCII
                , such as
                Electronic Mail
                .  It encodes binary
files into files containing only printable standard ASCII characters.

Written primarily for use with
                GuiArc
                 to add
                UUEncoding/UUDecoding
                facilities to it, it takes similar
                command line options
                 to other commonly
used archiving programs - though if you intend to use the program only via
GuiArc they will not be of much interest to you.

There is a fairly comprehensive
                installation script
                 included,
called 'Install' which will automatically install UUArc into your system,
including adding to the
                ArcTypes
                 file so that, if you have it, GuiArc
be able to use UUArc and play with UUEncoded files from now on.


If you like this program, use it a lot, and can afford to do so,
contributions to a charity in the U.K. called
                C.I.C.R.A.
                would be very much appreciated.
```

Enjoy. :-)

1.4 Technical Information

This program has been written in 'C' using
North 'C'
V1.3, and is
coded as nicely as I knew how to, but also as nicely as North 'C' will allow; unfortunately North 'C' V1.3 is not completely ANSI 'C' compatable and doesn't seem to allow function prototyping, but other than that North 'C' is an extremely usefull 'C' package for the Amiga. Although I have been a serious programmer since I was 13, I am rather new to 'C'!

The 'C' Source Code has of course been included, and
is fairly self-explanitory.

I have written the program with machine portability in mind, hid any documentation I had on my Amiga, and tried to stick to only the standard 'C' libraries. Unfortunately there is one machine specific part of the code which relates to extracting a filename from a filename with a full path attached to it; though this section of the code (like the rest of it) has been fairly thoroughly documented, so ought to be relatively easy to change as required by even the most novice of programmers. Having said that, although I have no experience writing 'C' code for UN*X, I just uploaded an exact copy of the source supplied onto a UN*X machine; it compiled and ran first time, successfully, without any errors at all (which rather made my day!)

Basically, the
UUEncoding
algorithm works by taking sets
of 3 8-bit bytes from the source file, and translates them into sets of 4 ASCII characters, each ASCII character being between decimal 33 ("!") and decimal 96 ("`"). Each line of UUEncoded data is proceeded by a UUEncoded number indicating the number of bytes required to be extracted from that line, is within normal line width boundaries (generally around 61 characters long) and is terminated like a normal text file line. The start of a UUEncoded file in an archive is indicated by a line containing a 'begin' statement (in lower case) followed by the file mode protection bits (3 digit octal number - ignored in this version as it would make the code rather system specific) followed by the name of the file. The end of a file is indicated by a line containing an 'end' statement (again in lower case.) Optionally a 'size' statement followed by the size of the file (decoded) in bytes can be included in a line after an 'end' statement line.

Checksums
have been used in UUEncoding algorithm - outputed in UUEncoded form after the data bytes at the end of the line; although checksum output can be stopped by removing the compiler pre-processor define called 'ADDCHECK'. The UUDecoding algorithm will check

checksums digits if they have been included, but does not require
them. The 'size' statement is included in UUEncoding, but is optional
for decoding - if it is there then it will be checked, but again the
decoding algorithm doesn't actually require it to be there.

The program makes its best attempt to work out if an archive is corrupt
by checking that all lines containing UUEncoded data are of the
right length, and all UUEncoded characters are between the
right limits, and checks any checksums (if present.) If a UUEncoded line
appears to be corrupt then that line is ignored and an error is
displayed on the screen; though the rest of that file will be still be
processed - this means that you ought to be able to give UUArc a mailbox
full of UUEncoded files, even if each UUEncoded file has been split up
into many seperate
e-mails
, as long as all the lines of the UUEncoded
files are in the mailbox and in the right order the mailbox file could
be processed by UUArc just as any other UUEncoded archive would be;
UUArc would spew out a few errors about encountering bad lines - but
these lines would presumably be the 'human' textual parts of the E-Mail
and have nothing to do with the UUEncoded file; UUArc would just ignore
such lines and only extract from the definitely UUEncoded lines! Infact,
just to test this was completely true, I stuck my current 100Kbytes of
mailbox right into the middle of a UUEncoded archive (including all the
e-mail headers etc. of course) 8-O - UUArc successfull managed to decode the
files in the archive without any difficulty still. 8-)

I am neither an Amiga 'Guru' (though I've owned one back since the
days the Fish disks were only into double digits) or an amazing
'C' hack; so the code is most certainly not the fastest of the
UUEncoder/UUDecoder around, but hopefully it makes up for this in its
completeness and portability. If you find any bugs or have any
suggestions, please do get in
contact with me
 - I'm an
electronics/computing student and am currently spending time trying to
get up to date with programming langauges; I sort-of avoided 'C' for
quite a while and am now paying my penance by spending many hours
practising writing 'C' code, particularly portable 'C', so any comments
you have on my programming style would actually be greatly appreciated.

Enjoy. :-)

1.5 acknowledgements

I would like to give great thanks to all those who have ↩
contributed to
the
Public Domain
 utilities/languages for the Commodore Amiga;
especially the writers of
North 'C'
,
A68k
, and

Blink
, and
of course not forgetting Fred Fish for his work in putting
together a renowned reliable source for this software; specifically
because I haven't ever had nearly enough money to buy a 'C'
compiler for my Amiga, infact since I bought
my Amiga back in the late 80's I haven't been able to afford to
purchase any programming languages or any Amiga documentation!
(Other than the
'The Kickstart Guide To The Amiga'
and very recently
Kernighan & Ritchie's 2nd Edition of
'The C Programming Language.'
I owe all the 'usefulness' I get out of my Amiga is due to the ↩
fantastic high
quality utilities/langauges available on Public Domain, especially via
the Fish Disk collection. Without the Public Domain versions of
many languages available for the Amiga, the grades I got during my first
and second year at University would have probably been very
considerably lower, and most certainly would have ment many many
dreary nights stuck in terminal rooms all night fighting for use of a
computer and printer. In my second year 10 miles away from my place
of residence: away from piece, tranquility, a nice graphics user
interface, cups of tea, Marmite sandwiches, decent music, a nice
bath/shower, and a bed immediately on completion of any programming work.

If I knew the origins of the
UUEncoding/UUDecoding
algorithms, then I'd
have acknowledged the programmer here!

1.6 bibliography

THE 'C' PROGRAMMING LANGUAGE - 2nd Edition
Brian W. Kernighan
Dennis M. Ritchie
ISBN 0-13-110362-8
Prentice Hall Software Series

Recommended reading for anyone wishing to pursue learning the 'C'
language, with a good reference section also for those well
experienced with programming other langauges. Not recommended for
novice programmers.

THE 'KICKSTART' GUIDE TO THE AMIGA
ISBN 0-9512921-0-2
Ariadne Software Ltd,
273 Kensal Road, London W10 5DB, ENGLAND.

The version I have came out just when Kickstart 1.2
was released, and therefore is rather out of date. If this is still in
print and if an up-to-date version of it is available, definitely
the 'paupers' replacement to having all the official books.

1.7 Reason For Writing This Program

```
                At the moment, it is more practical for me to obtain most of my
software via
                E-Mail
                , which means having to
                UUEncode/UUDecode
                all the files & archives, as E-Mail can only handle
                ASCII
                .

Now that the rather fabulous
                GuiArc
                 program is available, all
my archiving can by done via a nice graphical user interface
rather than the horrid CLI that I used to be stuck with....

almost......

BUT as I have to UUEncode/UUDecode, I was stuck having to return to
the CLI to use the rather old UUE/UUD programs.  I hunted around for
a UUE archiver that would interface nicely with GuiArc, but
couldn't find any.  Certainly the few others that were available were
restricted in thier use, and couldn't do some of the rather simple but
very useful acts of deleting/listing/moving files in an archive - all
functions that can be accessed by GuiArc and are generally available
with other archiving/coding systems.

Hence the birth of UUArc!
```

1.8 Detailed Description Of The Program

```
                This program functions like most other archivers available, and ↩
                   has
very similar
                command line options
                 to them; so if you regularly use other
archiver systems then using UUArc should be fairly intuitive - a brief
discription of the command line options is given if you enter 'UUArc'
or 'UUArc ?' at the CLI (obviously after installing the
program!)

It is based on the standard
                UUEncode/UUDecode
                 utilities already very
commonly in use throughout various computer systems, and will read/write
files compatable with these utilities (I have test all the
ones I've managed to locate with UUArc); although UUArc allows you to
store multiple files in an archive, is contained in one
single executable file, and unlike most other versions
available UUArc also has options to list, delete etc. files in an
archive.

UUArc is completely compatable with
```

```
                GuiArc
                 (so I hope anyway)
and all the archiving commands that GuiArc expects to be available
have been included.  The main reason for writing this program was to
enable me to perform all file conversions/archiving/decoding
that I ever required within the GuiArc program, so as to avoid
requiring the use of a CLI at any point.  As things stood I had to do
all my UUEncoding/UUDecoding via the CLI, which seemed rather a shame
and somewhat irritating - as everything else I could do with GuiArc's
very nice and very friendly graphical interface.  Included is
a suitable
                ArcTypes
                 file for GuiArc, to add to your current
ArcTypes file if you so wish to do so, to allow GuiArc to the UUE
system; this is automatically added by the
                installing
                 program
included.  The one incompatibility between UUArc and GuiArc is that
you can only do operations on archives relating to one file or all
files in an archive, i.e. extract all files, or selectively extract
one file at a time.  You can't select to extract 3 out 8 files (for
instance) in one go; in that case you would select each of the three
files in turn and extract them individually under GuiArc, or
extract all 8 of them in one go and ignore or delete the ones you aren't
interested in - though this shouldn't really pose much of a problem;
one generally wants to decode a whole archive or just one or two
files from the archive, not half of it.
```

1.9 Details Of Installation Of UUArc

```
                Installation should be relatively easy, an installation script  ↩
                   called
'Install' has been included which will attempt to do all the work for you.
Basically, all it does is to copy the UUArc executable program
from this directory into your 'C:' directory, and
if you have
                GuiArc
                 installed in SYS:Utilities, it
will attempt to add a suitable
                ArcTypes
                 file onto
the end of your current ArcTypes file.  If the script fails then
you can easily do the installation manually; copy 'UUArc' to your
'C:' directory, and add 'ArcType' onto the end of your 'ArcTypes' file
(which would normally be in the same directory as GuiArc.)

The program ought to work on just about any Amiga computer system,
(and most other machines if you re-compile the 'C' source
code on them!)
```

1.10 Bug Reports, Suggestions etc.

My current contact details are:

Postal mail-

Julie Brandon,
1, Olivers Mill,
New Ash Green,
Longfield,
Kent DA3 8RE,
ENGLAND.

E-Mail (until September '94)-

csc280@cent1.lancs.ac.uk

1.11 North 'C' V1.3

A
public domain
version of 'C', not completely ANSI 'C' compatable,
but extremely good none-the-less, produced by S.Hawtin I believe.

Translates 'C' into Assembly Language.

1.12 a68k

A
public domain
680x0 machine code compiler by Charlie Gibbs.

Translates 680x0 Assembly Langauge into object code.

1.13 blink

A
public domain
linker, produced by "The Software Distillery."

Translates pieces of object code into a functional executable file.

1.14 Public Domain, Shareware, Freeware etc. Software

Public Domain (P.D.) Software is generally software written and
released to be freely distributed, unlike commercial software for
which monies must be payed.

Shareware/Freeware is generally software to be distributed freely,
but charges must not be made by any other third parties for further
distribution other than by the author. In Shareware, if the program is
used, a contribution is required to be given to the author; normally
a small and very reasonable fee, for which the author will
generally give back-up, support, assistance and details
of future versions. Much Shareware software available is
considerably better than thier commercial counterparts.

NOTE:
P.D. & Shareware are usually intended for non-commercial uses
only, and the authors of the programs often have clauses
to this effect. There are also other strict clauses in some
software packages, such that any further distrubition of the
software must not be charged for, and that the files must
not be changed any further etc.

1.15 UUE Encoding Decoding Algorithms

A technique for turning
binary
files using all 8-
bits
into
ASCII
files using only 7-bits, so that these files can
be easily transmitted, for instance, over
Electronic Mail
.

1.16 ASCII

(A)merican (S)tandard (C)ode for (I)nformation (I)nterchange.

An agreed standard for the computer alphabet. Each 'letter' being
one
byte
in size, but only using 7
bits
of the byte.

1.17 Bytes, Binary and Bits!

A 'byte' is a computer number, a piece of a computers memory. ↩
Generally
8 'bits' in size. A program would be made up of a number of 'bytes'.

A 'bit' is a single element of a piece of a computer memory, having
only two states, representing 'binary' 1 or 'binary' 0.

'Binary' is the number system that computers work with, base 2. We
work and think in base 10; we count from 0 to 9, then 10 to 19, then
20 to 29 and so on - computers count from 0 to 1, then 10 to 11, then
100 to 101, 110 to 111 etc - each '1' or '0' is represented by
a 'bit'.

'Binary' is also a term used to describe files where each
'byte' may use all the available 8-'bits'; as apposed to an
ASCII
file where only 7-'bits' are used.

A 'checksum' is a special number used for testing that a file has not
become corrupted somehow. Effectively, one adds up all the bytes in
a file, and takes a few of the lower (least significant) bits of the
total produced and uses this as a checksum byte.

1.18 GuiArc

An extremely good
freeware
program that allows the user to use
archivers, extracting/adding files from archives, all under a
graphical user interface, rather than having to operate the archivers
via a CLI. Written by Patrick van Beem.

1.19 arctypes

An a file for the
GuiArc
program that tells
it how to use various archiving systems.

1.20 Crohn's In Childhood Research Association

If you can afford to do so, donations to the following U.K. charity,
or your countries equivalent, would be very very much appreciated:-

CICRA,
Parkgate House,
356 West Barnes Lane,
Motspur Park,

```
  Surrey,
  KT3 6NB.
  ENGLAND.


I can't afford to pay them back in monetary means much for all the
help they gave me and my family when I was younger, so the next
best thing I can think of doing is to donate my computer programs
for them. :-)
```

1.21 Quick Usage Guide for UUArc

```
                Usage with CLI only.  No graphical user interface is
available by default; that is what
                GuiArc
                 is for!


Just type 'uuarc' on its own to get some brief instructions:-


---


UUArc Version 1.3 by Miss J.G.Brandon Oct 05 1993.


USAGE:  UUArc -<command>[p][m] <archive> [<filename> ... ]

Where <command> is one of-
  l = List contents of <archive>.
  t = Test contents of <archive>.
  a = Add <filename(s)> to <archive>.
  x = Extract <filenames> from <archive>.
      (All files if no <filenames> given.)
  d = Delete <filenames> from <archive>.
      (All files if no <filenames> given.)

If included after the archiver command, the 'p' option
specifies full path names to be considered; otherwise path
names will be ignored.

If included after an add/extract archiver command, the 'm'
option specifies files to be moved from source, i.e. source
files will be deleted; otherwise they are left as is.

If applicable, <archive> must include the '.uue' extension.


---


Some examples:-

  uuarc -apm fred.uue ram:harry.lha
```

```
  This would UUEncode the file 'harry.lha' from the RAM disk, into a
  UUEncoded archive called 'fred.uue' in the current directory, with the
  'p' option specifying that the the path 'RAM:' is to be left attached
  to filename in the archive.  The 'm' option specifies that when (and if)
  'ram:harry.lha' is successfully added to the archive, the file is
  deleted (i.e. the file is physically 'moved' into the archive.)

  uuarc -x fred.uue

  This would extract all files from the UUEncoded archive 'fred.uue',
  stripping any path-names, leaving the original files in the archive.
```

1.22 Electronic Mail (E-Mail)

```
                A computer letter made up of
                ASCII
                 characters, sent from
one person on a computer to another person, via a network,
just like normal Post-Service Mail, but existing within the memories
of computers rather than on paper - and much much faster!
```