

UUArc

COLLABORATORS

	<i>TITLE :</i> UUArc	
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>
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	2
1.5	acknowledgements	4
1.6	Cups of tea	5
1.7	Arthur Dent	5
1.8	The Hitchhikers Guide To The Galaxy	6
1.9	Marmite Sandwiches	6
1.10	Decent Music	6
1.11	B52's Rock Lobster	6
1.12	Hazel O'Connor	7
1.13	Will You?	7
1.14	Eighth Day	7
1.15	Eeby-Jeebies	8
1.16	Bed	8
1.17	Programming Work	8
1.18	bibliography	8
1.19	Reason For Writing This Program	9
1.20	Detailed Description Of The Program	9
1.21	Details Of Installation Of UUArc	11
1.22	Bug Reports, Suggestions etc.	11
1.23	North 'C' V1.3	12
1.24	a68k	12
1.25	blink	12
1.26	compiler	12
1.27	Object Code	13
1.28	High-Level Language	13
1.29	Low-Level Language	13

1.30 executable	13
1.31 680x0	13
1.32 Fred Fish	13
1.33 Public Domain, Shareware, Freeware etc. Software	14
1.34 The 'C' Language	14
1.35 Assembly Language	14
1.36 Portable	15
1.37 UUE Encoding Decoding Algorithms	15
1.38 ANSI 'C'	15
1.39 'C' Function Prototyping	15
1.40 Kickstart	16
1.41 Operating System	16
1.42 ASCII	16
1.43 Bytes, Binary and Bits!	16
1.44 Archive	17
1.45 archiver	17
1.46 Files	17
1.47 Internet	17
1.48 network	17
1.49 Electronic Mail (E-Mail)	18
1.50 Snail Mail	18
1.51 GuiArc	18
1.52 arctypes	18
1.53 Graphical User Interface	19
1.54 Script	19
1.55 Icons	19
1.56 Command Line Interface	19
1.57 Directory	19
1.58 Crohn's In Childhood Research Association	19
1.59 Bugs	20
1.60 Quick Usage Guide for UUArc	20

Chapter 1

UUArc

1.1 Index To UUArc Guide

UUARC Version 1.1 08/07/93
J.G.BRANDON
PUBLIC DOMAIN

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

V1.1 - 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 communication 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'
compatible 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-explanatory.

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 separate

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 collections
 . 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 Cups of tea

Yes 'Brits' really do drink that much tea, just like
 Arthur Dent
 .

1.7 Arthur Dent

A character from Douglas Adams' more-than-three-books-actually ←
trilogy

The Hitchhickers Guide To The Galaxy

.

1.8 The Hitchhikers Guide To The Galaxy

Hmmmm.....

If you are still lost, then you've missed a vast section of one of the few enjoyable parts of a good full-filing life.

Go out and borrow a copy from your local library, or buy a copy. It really is very good you know.

(And NO, I DON'T work for the publishing agency!)

1.9 Marmite Sandwiches

Yummy! I'm addicted to them.

1.10 Decent Music

Any of:

Jean-Michel Jarre

Kraftwerk

Suzzane Vega

Good Classical Stuff

Hazel O'Connor ('Will You?' and 'If Only' in particular.)

Bonnie Tyler (Especially good for a good cry.)

B52's (Especially Rock Lobster, just like on my circuit board!)

Led Zepplin

Pink Floyd

Marillion (When with Fish) (Back to crying a lot again.)

Fish (No, not Fred Fish, I don't think I've ever heard him sing.)

Madness

Early 80's

Brian Ferry/Roxy Music

Peter Gabriel

1.11 B52's Rock Lobster

Some of the earlier Amiga's had many people confused who opened up their machines, as along with all the other technical references inside the machine on the circuit board was inscribed...

"B52 Rock Lobster"

Much controversy surrounded this strange inscription, especially as to its reason.

The circuit board revision number was B52 I believe, the name of a punk era pop group (who have had a revival in recent years), whos most notable tune from the past was Rock Lobster... hence the jolly jape to add the word "Rock Lobster" next to B52 on the circuit board, so I am told anyway.

1.12 Hazel O'Connor

Very little heard of, my most favourite singer/artist of all. (Well why else would I place a dedication in a piece of computer documentation?)

Relatively famous in the U.K. during the late 70's/early 80's. A new-age singer, who starred in the film "Breaking Glass", and wrote the sound track to the film. Because she played a kind-of punk in the film, people assumed her to be a punk.

"Breaking Glass" is still available, now on CD, on which are two of her most well known songs,

Will You?

and

Eighth Day

. Highly

recommended, and rather pleasantly different to most stuff availble. If you can get hold of them, also very much recommended but much less 'punk/new age', are the albums "Sons & Lovers" and "Cover Plus" (also available on CD.)

She has started doing the occasional small concert or two over the past few years, so you might just see her around if you look out hard enough.

1.13 Will You?

Definitely a weepy one. Gets me crying every time.

1.14 Eighth Day

A story about artificial-intelligence computer systems being given more and more control, over time, over Governmental administration

The version I have come 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.

1.19 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
and
archives
, as E-Mail can only handle
ASCII
.

Now that the rather fabulous
GuiArc
program is available, all
my
archiving
can be done via a nice
graphic 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 their 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!

1.20 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 compatible 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.

1.21 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!)

1.22 Bug Reports, Suggestions etc.

Contact over internet would not be too easy at the moment; I'm likely to go through quite a few more e-mail address changes as time goes on, so its best to stick with snail mail for now-

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

1.23 North 'C' V1.3

A
public domain
version of
'C'
, not completely
ANSI 'C'
compatible,

but extremely good none-the-less, produced by S.Hawtin I believe.

Translates 'C' into
Assembly Language
.

1.24 a68k

A
public domain

680x0
machine code
compiler
by Charlie Gibbs.

Translates 680x0
Assembly Language
into
object code
.

1.25 blink

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

Translates pieces of
object code
into a functional executable file.

1.26 compiler

A computer program that translates a computer language into a
lower level language. (See
High-Level Languages
and
Low-Level Languages
.)

1.27 Object Code

A type of file produced by a compiler from a high level language
,
somewhere between a high level language and executable code,
though still requires work to be performed on it by a linker before it can be 'run'.

(This isn't a very good description, but if you are reading this then giving the full technical explanation probably wouldn't help much.)

1.28 High-Level Language

A form of computer program that is intended to be fairly comprehensible to humans; which without translation a computer itself would not understand.

1.29 Low-Level Language

A form of computer language that, although not very comprehensible to humans, is much more understandable to a computer.

1.30 executable

A type of file that can be 'executed' or 'run' by a computer; a program!

1.31 680x0

The central-processing unit, the 'thinking' (so-to-speak) part of a computer, inside the Amiga is based around the 68000 series of chips.

1.32 Fred Fish

Fred Fish puts together one of the best known and regarded collection of
Public Domain
and

Shareware
software available
for the Commodore Amiga series of computers; the Fred "Fish" disk
collection.

1.33 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.

1.34 The 'C' Language

A very good, extremely popular,
portable
,
high-level language
used throughout the computer world. Although it is a high-level
language, it can be relatively easily converted into a

low-level language
quickly, efficiently, producing
executable
code that will generally run much faster than code produced by
many other high-level languages.

1.35 Assembly Language

A very very
low-level language
, the nearest step you can get to
the language that the computer understands - but still retaining

some human 'understandability'.

1.36 Portable

'Portability', with respect to computing, means the ability to translate a program designed for use on one computer system to be usable on another computer system.

1.37 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.

.

1.38 ANSI 'C'

Agreed standard for the 'C' programming language, to attempt to remove the differences between various implementations of the language, and to increase the languages portability.

.

1.39 'C' Function Prototyping

In good high-level languages it is possible to define exactly how various functions of the program should be used, to help stop bugs being designed into the program. In ANSI 'C' the method used is 'C' Function Prototyping.

1.40 Kickstart

The name of the operating system used on the Commodore Amiga series of computers. Many versions have been in regular use, most commonly from around 1.1 to 3.0.

1.41 Operating System

The special set of computer programs working within a computer to perform the basic required tasks of running the computer; also supplying various commonly used tools for other programs to run on the computer.

1.42 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.

1.43 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' uses 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.

1.44 Archive

A single file, in which a collection of other files is stored. An archive is also generally compressed, to use less bytes to be smaller in length - to make transmitting/receiving the program cheaper/faster/easier. ←

1.45 archiver

A program for manipulating archives, generally allowing the user to add, extract, remove, examine/list files in an archive.

1.46 Files

A named collection of bytes, representing either a computer program, or data (information) of some sort.

1.47 Internet

A popular world wide network, allowing all sorts of different forms of links and communications between computers and thier users, including electronic mail. Allows computers to 'talk' to each-other.

1.48 network

Computers linked up together to be able to communicate with each-other.

1.49 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!

1.50 Snail Mail

An 'in' computer term to describe normal Postal-Service mail.

1.51 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.

1.52 arctypes

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

1.53 Graphical User Interface

A recent step away from the traditional means of operating computers via a CLI . On computer system using an operating system based around a graphical user interface (such as the Commodore Amiga), commands are given to the computer by the user using common-sense/intuition; by selecting appropriate Icons that represent your desired instructions to the computer on the screen.

1.54 Script

A set of computer commands, generally CLI commands.

1.55 Icons

Pictorial representation of a command/instruction/object.

1.56 Command Line Interface

A system of instructing the computer your wishes by entering a series of textual commands.

1.57 Directory

A place on a computer where files are stored.

1.58 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.
UNITED KINGDOM.

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. :-)

1.59 Bugs

Faults in computer programs!

1.60 Quick Usage Guide for UUArc

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

UUArc Version 1.1 by Miss J.G.Brandon Jly 08 1993.

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

Where <command> is one of-

- l = List contents of <archive>.
- t = Test contents of <archive>.
- a = Add <filename> to <archive>.
- m = Move <filename> to <archive>.
- x = Extract <filename> from <archive>.
(All files if <filename> is missing.)
- d = Delete <filename> from <archive>.
(All files if <filename> is missing.)

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

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