LhA2LZX

Richard Burke

COLLABORATORS

	TITLE : LhA2LZX	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Richard Burke	August 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

LhA2LZX

1.1 LhA2LZX v3.2 AmigaGuide Documentation

```
LhA2LZX
~~~~~~~
The Archive Converter
© 1998 Richard Burke
```


1.2 Introduction

```
Introduction
```

When LhA came along, Amiga users were able to store files in one (generally) smaller file - an LhA archive . Then LZX came along - allowing much better compression rates for some files, so much more space could be saved.

When I first got LZX I found it a lot of work converting my old LhA archives to the new LZX format via the CLI, and sometimes found that after doing it that my original LhA files were smaller than the new LZX ones! Thus LhA2LZX was born. This script file allows the user to convert their archives from LhA to LZX or vice versa, or alternatively it will intelligently convert the archive to the smallest format! It is all done by requesters - so you don't even need to touch the keyboard!

And new for v3.0 is the ability to easily use it directly from Directory Opus"}, and also to repack each archive to make sure the maximum amount of space is saved! Just select the files you wish to convert, and click a button!

This program is an AmigaDOS script file, so you can load it into a text viewer/editor and see how it all works. Almost all of the lines have been commented to help you understand this.

LhA2LZX also supports LZH archives (they are very similar to LhA).

In a nutshell...

LhA2LZX can perform several different actions:

- It can convert archives between the two formats
  (e.g. give it an LhA archive and it will convert it to LZX, or vice versa)
- It can convert to the smaller of the two archives
  (e.g. give it an LhA archive and it will create the LZX version, compare the size with the LhA version and save the smallest, or vice versa)
- It can repack an archive
  (e.g. give it an LhA archive and it will de-archive it and re-archive it with LhA to make sure it is the smallest size for that particular format)

1.3 How to use LhA2LZX

How To Use

- To begin with,you can install LhA2LZX using the installer script provided. A full install will install everything in the archive, and a basic install will just install the latest version of LhA2LZX with this .guide. You will need the full install to access all functions in this document. Remember that one function of LhA2LZX as an on-going process is to help teach how to write AmigaDOS scripts, and so some of the previous versions are accessed from this .guide.
- Don't forget to copy 'inf' from the util/cli/mykinf.lha archive to C: , or at least to somewhere in your path
- To run from Workbench, Double-click on the LhA2LZX icon
- To run from the CLI, cd to the drawer LhA2LZX is in and type

```
LhA2LZX
```

  You may also follow this with an archive name:

```
LhA2LZX "dh1:Temp/new.lzx"
```

  If you do not CD to the drawer, you will get an error message telling you to do so. You may omit the quotes unless the archive path has spaces in.

- To run from Directory Opus, follow the specific instructions below.

- The first time you use LhA2LZX, you will be prompted for a path where the temporary drawer 'l2z' is to be created - this is used to de-archive the files in the archive during the operation. There should be at least 3-4 times the size of the original archive here. The path is then saved to a file 'LhA2LZXl2zpath' in ENVARC:,and will be used in all following sessions. Sometimes you may not have enough memory to use RAM: as the default path (which is recommended as it is much faster), and you will be prompted to choose another path. This new path will then replace the old. If you wish to ever change the path, just delete the file 'env:LhA2LZXl2zpath' and run LhA2LZX again
- Select the archive(s) from the file requester
- Choose whether to convert to the other archive type (i.e. LhA to LZX or LZX to LhA), to the smallest archive or to repack the archive.

Using LhA2LZX in DOpus

As of v3.0,LhA2LZX has now been modified for use as a Directory Opus button. I use v5.11, but it should work on any v5.0+. Please let me know if it doesn't.

- Create a new button, and set up the function as follows:

```
AmigaDOS  cd "LhA2LZXdrawer"
AmigaDOS  list {F} >RAM:LhA2LZX.exe LFORMAT="execute *"LhA2LZXdrawer/LhA2LZX*" *"% ↩
   F%N*""
AmigaDOS  execute ram:lha2lzx.exe
AmigaDOS  delete ram:lha2lzx.exe
```

where LhA2LZXdrawer is the path of the LhA2LZX program drawer e.g. dh1:temp

- Set the flags "CD source" and "Rescan source".
- To use, select the archive(s) and click the button.
- Each archive will be processed in turn
- You will know when each process has finished because the source lister will be rescanned, and you will get the LhA2LZX messages detailing which archive is being used. Do not worry if the lister stays busy for a moment - it is just LhA2LZX cleaning up after itself or processing the archive.

Notes

LhA2LZX can be stopped during execution by pressing Control-D or Control-C

If you know that you aren't going to change the location of LhA2LZX, you can copy the file ENV:LhA2LZXpath to ENVARC: to save LhA2LZX working out it's own location each time it is run. You will need to delete both these files if you decide to move LhA2LZX, but then you can copy the new ENV:LhA2LZXpath over to ENVARC: after running LhA2LZX from the new drawer.

If you want to see exactly how LhA2LZX works, get KingCON (an excellent Shell replacement which gives a scrollbar on the side of the CLI window) and type set echo on before running LhA2LZX from there. This will give you a line-by-line output of what is happening, and is very easy to read and

compare with the original LhA2LZX code to see how, amongst other things, the Environment Variables work, and how all the bits fit together. This is available on the AmiNET as util/shell/KingCON_1.3.lha. If you want to send a really good bug report, simply cut-&-paste the output from the KingCON window into an e-mail or attachment so I can see exactly where and what the problem is :^)

If the screen keeps flipping between Workbench and DOpus when requesters appear (if DOpus isn't the default public screen), try using ARQ, the requester improver, available on the AmiNET as util/cdity/Arq1_83.lha. This has the added bonus of adding small animations in the requester, and the possibility to add sounds to specific requester types.

I'd also recommend using a colour of 32,70,222 for the fourth (blue) colour in your palette for reading this .guide, and a proportional font.

From v3.1,LhA2LZX will invoke an AREXX script to handle various things that AmigaDOS cannot,so you should have the program Sys:System/RexxMast, and also the following commands somewhere in your startup:

```
ASSIGN REXX: S:
PATH Sys:system sys:rexxc ADD
RexxMast >nil:
```

The last one is optional, it just starts RexxMast so that it is already resident. The first two command lines above should already be in your startup-sequence. The installer script will automatically copy lha2lzx.rexx over to wherever REXX: is assigned to, as this is where the command that starts AREXX scripts, RX, looks for them.

1.4 Requirements - What you need to run LhA2LZX

Requirements

LhA2LZX requires several things to work:

```
[ BaseName        (1.3, arp component)    ]
  CD          (2.x/3.x internal)
  Copy          (2.x/3.x in C:)
  Delete        (2.x/3.x in C:)
  Echo          (2.x/3.x internal)
  Else          (2.x/3.x internal)
  Endif       (2.x/3.x internal)
  Eval          (2.x/3.x in C:)
  Filenote        (2.x/3.x in C:)
  Inf         util/cli/mykinf.lha on AmiNET, put in C:
  If          (2.x/3.x internal)
  Lab         (2.x/3.x in C:)
  List          (2.x/3.x in C:)
  MakeDir       (2.x/3.x in C:)
  Protect       (2.x/3.x in C:)
  Quit          (2.x/3.x internal)
  Rename        (2.x/3.x in C:)
  RequestChoice       (3.x only, located in C:)
  RequestFile       (3.x only, located in C:)
  Resident        (2.x/3.x internal)
```

```
  RX          (2.x/3.x in Sys:REXXC)
  Search        (2.x/3.x in C:)
  SetDate       (2.x/3.x in C:)
  Skip          (2.x/3.x internal)
  Type          (2.x/3.x in C:)
  Which       (2.x/3.x in C:)
```

As indicated, you will already have these either in ROM or on your Workbench disks in the C: drawer (or already on your hard disk).

A hard disk is greatly recommended,and some spare RAM (depending on the size of the archives you are converting).

Both the LhA and LZX tools will also be needed - preferably in C:
[these are both available on the AmiNET, and from most PD libraries]

LhA can be found at util/arc/LhA_e138.run and LZX is at util/arc/lzx121r1.lha on the AmiNET.

Since v2.6 the registered version of LZX is needed - and the registered keyfile is now freely available and part of the LZX distribution.

IconX will also be needed in C: if you want to double-click the icon to start LhA2LZX.

From v2.3+ of LhA2LZX, BaseName is no longer used.

From v2.7a+, a 68020 or faster CPU is needed.

Versions of RequestChoice & RequestFile for 2.x are available on the AmiNET.

Tested on:

A1200, 25Mhz '020 + FPU (Apollo Turbo 1220), 170 MB HD, 2/4 MB RAM, KS 3.1
A1200, 25Mhz '030 + FPU (Blizzard), 580 MB HD, 2/4 MB RAM, KS 3.1
A1200, 40Mhz '030 + FPU (Apollo 1230), 2.1 GB HD, 2/14 MB RAM, KS 3.1

Untested on KS 2.x.
If you can test it on any of these, please mail me with the results :^)

1.5 Writing AmigaDOS Scripts

Writing AmigaDos Scripts Read LhA2LZX

AmigaDOS scripts are very useful because everyone who has Workbench has the necessary software to write and run them. They are written in a language which is easy to learn and easy to understand, and can be used to do many things. The startup-sequence is the most well known of these scripts, and shows just what they can do.Programs can be called and started,and arguments can be passed to them, which is essentially how LhA2LZX works - LhA and LZX are called from the LhA2LZX script and the arguments that the user has selected are passed to them. An important aspect of LhA2LZX is the use of Environment Variables.

Scripts are written using a text editor (such as Ed), and do not need to be

compiled, so you can simply run them from Workbench or the CLI (which is great when you're writing them - just the click of a button to test it!).

Comments (anything written after a semi-colon ";") are ignored when the script is running, so don't be afraid to comment your file so you clearly know what bit does what. They will not affect the actions of the script.

If you do not have an AmigaDOS manual, detailing what all the keywords do (there wasn't one in the A1200 distribution), there are many available in the Public Domain - the one I use is the fantastic DOSMan by Peter Bagnato which details the usage of commands up to and including the new WB 3.1 ones, and gives examples of how to use them. Also get your hands on other script files such as the other ones I have done , or DiskSqueeze, a DMS alternative that crunches entire disks with LZX! That file helped greatly in creating LhA2LZX.

The best way to learn script writing techniques is to read loads of other scripts, making sure you understand them, and experimenting if you do not (and even if you do!). Trying things out for yourself is a Good Thing(TM). If you read LhA2LZX and everything in the Program History page, you can't go far wrong :^)

1.6 Arguments

Arguments

Arguments are the keywords that follow the program's name if started from the CLI,and tell the program (for example) which file(s) to use, any special command or options to use and so on.

For example, if you typed

```
  LZX ?
```

in a CLI, you would get a list of the arguments you can use:

```
  LZX [-<options>] <command> <archive> [<file>...] [<destdir>]
```

where options are any options (such as make filenames uppercase)
command is what action you are performing (e.g. archiving)
archive is the name of the archive e.g. to be saved
file is the name(s) of the file(s) to be e.g. archived
destdir is the name of the destination drawer where the archive is
e.g. to be saved.

e.g.
if you typed at a CLI

```
    LZX -F a Test.lzx dh0:Games/#? dh1:
```

you would be archiving all (#?) the files in the dh0:Games/ drawer,
with a Fast progress display (-F) and the archive would be saved as
Test.lzx in the dh1: drawer.

1.7 Environment Variables

Environment Variables

Environment Variables are exceedingly useful in script files as they allow numbers or text to be saved to a small file in ENV: (in RAM:) and used by any program. What makes them so useful is that if a dollar sign ($) preceeds the filename when it is called, the contents of the file are used instead. So if for instance the text dh0:Games/ was saved as the file ENV:zxc, and you typed

```
  cd $zxc
```

in a CLI, the current drawer would change to dh0:Games/ - just as if you had typed

```
  cd dh0:Games/
```

Here is a relevant part of LhA2LZX that shows this:
Click to find in LhA2LZX

```
if $suf EQ lha
  cd $drw1
  cd l2z
  $wlha x -a -F -M -x $ldfl
```

Previously in the script, List was used (found in C:) to get the suffix (all the letters after the final full stop in the filename) of the filename to determine if it was an LhA or LZX archive, and the suffix was saved as the file ENV:suf. So if the file is an LhA archive, the line would be the same as "if lha EQUALS lha". It would then cd to the drawer the user had chosen to put the temporary storage drawer l2z, and would then cd into l2z itself.Next it would call LhA(with the location of the LhA program saved as the file ENV:wlha) and extract all the files, preserving their attributes (-a) from the chosen archive (the filename of the archive is saved as ENV:ldfl), with a Fast progress display (-F), preserving the path names(-x) with no "autoshow" files (which automatically display themselves when LhA de-/archives them). So if LhA was in C: and the archive was called dh1:Test.lha(which would have been selected by the user via a requester),the final line would be the same as

```
  C:LhA x -a -F -M -x dh1:Test.lha
```

Environment Variables can be used also to store numbers:
Click to find in LhA2LZX

```
  eval >env:flmem $flsz * 4
```

More info
This line calls Eval (in C:), and tells it to multiply the size of the original archive (found with List and saved as ENV:flsz) by 4, and to save the answer as ENV:flmem instead of printing it to screen. This is used in LhA2LZX to determine if the user's machine has enough memory to perform the action of de-/archiving (LZX archives can de-archive sometimes to 3-4 times the size of the archive) if they have chosen RAM: to temporarily store the files of the archive, and warns them if there may be insufficient memory - note here that file redirection is used (the > sign), which enables user's choices to be saved in files rather than be printed to the screen.

When creating your own script files, it is a sound idea for the very first line to be

```
set echo on
```

as this will give a display of all the actions being performed, so you can see exactly what is happening and where things need to be changed to fix any problems. This can be deleted when the script is finished and everything works as you would want it to.

1.8 Lisa Loeb - Cutest Rock Chick Queen in The World!

It’s hidden page number 3!

No explanation needed here about the wonderful Lisa Loeb! Just go buy one of her albums - Tails or Firecracker - and prepare to be transfixed forever! Her albums are so good that you can leave them in your CD player for months without tiring of them! So go on then. Treat yourself. In fact, I’m just off to go put Firecracker on...again...

[muffled sounds of inept teenager fiddling with a beautifully pink CD case...]

Clue Four: How best would I describe my beautiful girlfriend? In a word of 9 letters? That’s on this page?

1.9 Bugs in LhA2LZX

Problems

Apparently the clear screen command (echo "*ec") doesn’t work for KS 2.0, so I changed it back to "*e[1;1H*eJ" in v2.6a (this moves the cursor to column 1 row 1 and clears all that follows it).

When de-archiving, if it encounters an empty drawer, LZX may display the error message

```
*** LZX: Error writing to disk; disk full?
```

but still continues to de-archive regardless. This seems to be a bug in LZX, and seems to have no effect on the process. When the files are archived by LhA seconds later, the empty directories are also archived as normal.

e.g. Here is a copy of the output from one of my tests:

```
LZX 1.21 (Registered) Archive/Extract utility - 68020/68030 Version.
Copyright © 1995 Data Compression Technologies. All rights reserved.
Registered to the Amiga community for non-commercial use.

Extracting files from archive 'AQ2:DiskSalv/Executive.lzx':
 (       0 /       0) executive_v2.10/empty/ (       0 /       0) executive
```

```
*** LZX: Error writing to disk; disk full?
 (       0 /       0) executive_v2.10/c/empty2/empty3/ (       0 /       0)
*** LZX: Error writing to disk; disk full?
  ** 30 file(s) extracted, all files OK

Operation successful

LhA Evaluation V1.51 - Copyright (c) 1992 by Stefan Boberg.
All rights reserved. Not for commercial use.

Creating new archive 'Executive.lha':
  Frozen: ( 52.4%)    3604 =>   1713 : executive_v2.10/.Product-Info
  Archiving empty directory 'executive_v2.10/c/empty2/empty3'
  Frozen: (  6.0%)    3576 =>   3360 : executive_v2.10/data/DashboardPrefs.l
  Stored: (  0.0%)   38661 =>  38661 : executive_v2.10/data/Register.lzx
  Archiving empty directory 'executive_v2.10/empty'
30 files added, all files OK.

Operation successful.
```

As can be seen above, the empty directories I added ("c/empty2/empty3" & "empty") are still archived,so the message does not seem to affect anything. For clarity,the disk was not full.

The Tilde (~) Bug (fixed in v3.1)

Since having access to the AmiNET, I have found an additional bug, and that is AmigaDOS's and LZX's inability to handle tilde's (~) in filenames.I found this when copying files from the PC's here at University to my disks to take back to my trusty Amiga, when if a filename is greater than 8 characters long (suffix not included), it is truncated: so for instance an archive called "longarcname.lha" would be renamed "longar~1.lha" when saved to disk. When I tried to use these in LhA2LZX, LZX would not do anything with them, but gave an output of Operation successful. I have tried getting round this by detecting a tilde in a filename and getting the user to input a new filename, but AmigaDOS treats tilde's as something to do with negation - it is used in pattern-matching. If you want to get a list of all the files in a drawer, you would type

```
  list #?
```

If you wanted a list of all the files except for the icon files, you would type

```
  list ~(#?.info)
```

which would list all the files apart from those ending in info (icon files). So it seems if an archive has a tilde in it's name, AmigaDOS and LZX try to operate on it in a pattern-matching context.

Using the Type command was the closest I got (e.g.

```
  Type FROM arc~2.lzx TO arc2.lzx
```

but the command only copied part of the file. I tried also Copy, Join and Rename, but these didn't change the filename either. The only way to change the filename via the CLI would be to use the apostrophe (') before the tilde e.g. if your archive was called my~arc.lzx:

```
  Rename my'~arc.lzx TO myarc.lzx
```

This bug was fixed in v3.1, by use of an AREXX script. I was able to get the above command line by finding the position of any tilde's and inserting

an apostrophe before them.

Thanks to Frank Bunton for the help here, both with how to rename such files by the CLI and by writing the fabulous "AREXX For Beginners".

There was a bug found in the 68000 version of LZX (which is why you now need to use the '020 or higher version), and this is discussed fully in the program history for v2.7a.

Another bug which may appear has come about in v3.0. The size of the file listing of the RAM: disk is found as a check to see if any files have been selected for conversion - if the file listing of your archives is exactly the same as the entire RAM: listing, LhA2LZX will think you have not selected any files, although the chances of this are remote. To get around it, just put your archives in a different place and convert them from there, or copy a file over to RAM: (which will then change the listing size) and try again.

This AmigaGuide has been tested only on WB3.1, so it is possible not all of the keywords will work under 3.0 or below systems. If anyone can read this on any other WB version, please contact me (you can load this into a text viewer and read what all the missing words,if any, are). If all is not well, I will change it.

Bug reports and suggestions of improvement are welcome.

Feel free to try out the first versions of LhA2LZX (i.e. 1.x), but only use copies of your archives as they have bugs in - hence you can see where I improved it... :^)

1.10 Aw....

I'm putting in my final secret page dedicated to my wonderful girlfriend Leann. Everyone else seems to thank their girlfriends in the credits, so I thought I might as well do it too. So...

...Leann, thanks for being so great in all the time we've known each other,
and an extra big thanks for all the time we've been together.

"She can't tell me that all of the love songs have been written,
'cause she's never been in love with you before..."
- Lisa Loeb, "Sandalwood"

1.11 Program History & Development

Program History

Versions 1.1, 1.2, 2.0 - v3.2 are included in this distribution so you can see how it developed. Files are commented from 2.0+.

Version No.

~~~~~~~~~~~~
1.0 Initial incarnation

1.1 Added filesize difference to show which archive was smallest

1.2 Fixed bug of 'l2z' not being deleted if the original archive was smaller. The following extract is from the original docs:

"- DO NOT QUIT WHEN DE-ARCHIVING otherwise ONLY THE FILES CURRENTLY DE-ARCHIVED WILL BE PLACED IN THE NEW ARCHIVE. The new archive will not contain all the original files and will overwrite the ORIGINAL archive.

So DON'T DO IT!

This happens because it is an AmigaDOS script file, and will perform the next action immediately after completing the previous one. Thus if it is de-archiving and it is interrupted by a CTRL-C, it will perform the next action of archiving everything it has (so far) de-archived."

I found that if I aborted either LhA or LZX whilst they were de-/archiving then some of the original files were lost and, due to the nature of LhA2LZX's workings, were irretrievable unless a backup was kept. LhA2LZX was such that after it had de-archived and archived, the new file would replace the old, so files would be lost if this process was interrupted - some of the original files would not be archived and then the new would replace the old. Not a good thing. This was fixed in v2.1.

2.0 Added the extra option to convert to the other filetype. This was done for 2 main reasons. Firstly in case the user merely wished to convert between the filetypes, and secondly so I could convert my .lzx files to .lha,allowing me to enter them interactively via DOpus5 and add to them - my version of LZXDir 26.6.95 v1.1 (modified by Stone-D) doesn't allow me to add to .lzx files interactively.

Note
Directory Opus is a filemanager which allows you to quickly and easily perform actions on files. LhADir & LZXDir are two add-ons ("modules") which allow you to enter an .lha/.lzx file as if it were a normal drawer and perform actions on the files within, e.g. viewing a picture, as well as deleting and adding files to the archive as if you were merely transferring files from one drawer to another. If I want to add a file to an .lzx archive however I have to convert it to an .lha archive due to the version I have.If anyone has an updated version of LZXDir, please feel free to send it to me.

This version of LhA2LZX also checked if the file was already of the type that the user wished to convert to.

2.1 Added routines to check if the FULL path of the archive was chosen from the file requester by checking the size of the path name. Also did the same to check that a file had been chosen.Altered the "Convert to. . ?" routine to automatically check the filetype as this made things more logical (previously you could try to convert an .lha archive to an .lha archive . . .), and also saved me from trying to remember what type I'd just converted the file to. The routine allowing the checking of the filetype was already being used in LhA2LZX, so it was really just a
~~~~~~~~~~~~

cut-&-paste job once I'd realised it.

* MAJOR BUG FIX *
Stumbled across the means to add error detection if something went wrong in the de-/archiving process! BIG thanks to Peter Bagnato and his DOSMan creation - an AmigaGuide detailing the usage of AmigaDOS commands, with examples - which is a boon to those of us that haven't got an AmigaDOS reference manual. Fixing this bug meant that if anything went wrong when de-/archiving (such as a user abort or a corrupt file), then LhA2LZX would stop immediately and so all archives, the new and the original, would remain intact. And all by adding only 6 lines of code . . . I could now sleep at night.

2.2 Deleted all the 'deletes' at the start of the file as I decided that they were unnecessary. If LhA2LZX had been quitted and then re-started, the env: files would have already been deleted when it was quit the first time. If it was being started for the first time, chances are the files wouldn't be there anyway as env: is a temporary drawer in RAM: and therefore erased with each reset. If the user chose to convert more archives(the final requester of the program), then the env: files would be overwritten with the new variables. This also allowed me to use the drawer of the last archive chosen as the default path for the next archive - something I had wanted to do from the start - as it saves on searching through all of your directories if you have a drawer with several archives in which you wish to convert.

I also changed the new archive copying routine to copy the new archive to the selected destination BEFORE deleting the old in case of mishaps, but then altered it back for two reasons - there may not be enough space in the destination, especially if it is a floppy, to contain both the archives (new & old) before the old is deleted, and also it would now make little difference as this part of the script would never be reached if something DID go wrong when it was de-/archiving, now that the error detection was in place.

The "Convert to . . ?" requester was changed to show the filename in the title bar. The output messages detailing which archive is smaller were put into requesters to lessen the output in the CLI window - but I had to sacrifice the ANSI colouring to do this. Now the only CLI output to screen is the coloured closing message, which is displayed whilst all the temporary files created are deleted.

Amount of free memory is checked if the 'l2z' is selected in RAM:, to make sure you won't run out, and a requester pops up advising you if this might happen. I would also like to find the amount of free space of the destination drawer (in case RAM: isn't chosen) to see if there is enough, but I haven't yet found a way to do this using the AmigaDOS files in the Workbench distribution.

A clever trick is also employed here - that of using the backtick `. You can surround a command by these and pass this as an argument to another command. The command in the backticks is evaluated first, and the result is then passed to the first command

e.g.
echo `eval 2+2`

would call Eval (which performs numerical functions), pass 2+2 to it (which would give the answer 4), and pass the anwser 4 to Echo, which would then print the answer to the screen. It would be exactly like typing echo 4, and that is how echo would see it.

2.3 BaseName now is no longer used, as a function of List is used instead, to give the filename without a suffix and the suffix on its own.Deleted a couple of deletions of environment variables as they were overwritten on the following line, and so were unnecessary. Added a small routine just to check that the archive name does have an lzx/lha/lzh suffix(lzh is de-archived by LhA). Added detection to check firstly if the alternate path for 'l2z' exists and secondly that it is the full pathname.

2.4 LZH archives now processed fully as if they were normal LhA archives. Found a much nicer line to clear the screen for the final Thankyou message (from an old Amiga Power coverdisk!). Extra options added to the LhA/LZX command line to give faster progress display and suffixing. Added extra ANSI line "*e[0 p" which supposedly turns off the cursor to make text output twice as fast, so LZX etc. can now work faster. Empty directories now dealt with and de/-archived as usual, although there seems to be a bug in LZX here - luckily this seems to in no way affect the running of LhA2LZX.

Also added a similar routine to that of the default path of the archive when using LhA2LZX for multiple archives (see v2.2) to have a default path for where 'l2z' is stored, in case the user doesn't have much RAM: (and so won't want to go searching through drawers each time they want to have 'l2z' somewhere on their hard-drive).

2.5 Added 2 small routines just to check that both the path and the archive exist when selected, in case the user decides to type in either. Also improved the "Convert to . . ?"requester to show exactly what filetype you are converting and what you can convert it to. This makes things much easier to see (in terms of options), and also shows off the ability of filetype detection more.

Also fixed a small bug in the Insufficient Memory area to do with quotes and volume names with spaces in, such as Ram Disk:.Filenames or volume names with spaces need to be surrounded by quotation marks, like this:

"Ram Disk:"

so they can be accessed -if you just type Ram Disk: in a Shell,AmigaDOS will try to run the command Ram. Now the path name of the archive can have spaces in.

2.6 First official release (on my web page)
As the keyfile for LZX is now freely available on the AmiNET, extra features of LZX are enabled, such as de-archiving of LhA/LZH files,with a 25-30% speed increase. Also is the option of faster archiving at the cost of 32k of RAM. I have included both of these here to increase the overall speed of LhA2LZX. I tried here to fix the tilde bug, but without much success. Expect better handling in the next release!

2.6a First AmiNET upload
All I did here was shorten the length of the program by renaming the variables with the prefix "nl2z", which is unique enough for me to use

list to generate a list of the variables at the end and use this in a script created by LhA2LZX to delete them all. I got this idea from reading the startup-sequence (it uses the same principle but runs monitor drivers). I also changed the command that clears the output screen at the end to make sure it will work on KS2.0.

2.7 This version now contains filename error trapping, and fixes the tilde bug for the moment. If a tilde is detected in the filename, an error message is produced and the program quits. I am currently trying to think of a way round this without quitting the program or have the user typing an apostrophe before the tilde themselves, but I really don't know of a way to get AmigaDOS to scan filenames and insert specified characters before other characters :^(, and I don't really want to use AREXX or an actual program to do this as LhA2LZX is not just a file conversion utility, it is an exercise in AmigaDOS,showing its potential and all that it can do.

LhA2LZX now skips to the end and deletes all variables currently made if LhA or LZX cannot be found, instead of just quitting.

As the registered version of LZX is now needed, a quick check is now performed to see if the lzx.keyfile exists,as otherwise LZX won't work.

2.7a It seems that the change to de-archiving with LZX instead of LhA has revealed a bug in the 68000 version of LZX, which seems to ignore some of the command line options for de-archiving, or at least not process them correctly as the 68020 version does. The filename of the archive is given to LZX,but the '000 version adds the suffix ".lzx" to the file even if is specifically told not to. This means it cannot find the file (because it is looking for "myarc.lha.lzx" instead of "myarc.lha"), and so it returns a Return Code of 10. If there is a user-abort, a Return Code of 20 occurs (Return Codes are values passed back from programs to indicate the result of an operation - such as successful, fail, user - abort and so on),which is what LhA2LZX checks for,as I found previously nasty things happened if a user aborted the program (ie the archive was deleted). I tried to also check for this RC of 10, but found that when checking a Return Code only one could be checked. Knowing already that a 0 is returned if the operation is successful, I altered the Return Code checking line to inversely check - ie check that the operation was not successful, and act accordingly. This would mean that if anything went wrong (ie something that had not happened during my tests),LhA2LZX should compensate and Do The Right Thing, even if I had not anticipated the problem or even known about it.Now LhA2LZX pops up an error message requester telling you of the problem, and then skips to the "More...?" requester. Your archive should remain intact, although not converted, and will not be deleted as it would previously have been. If you are using an '020 or higher, you will probably never see this requester.

I also changed part of the LZX command line as a result of this (the LZX command line refers to the line of code in LhA2LZX in which LZX is started and various commands and options are passed to it -for instance what the archive name is, whether to archive everything in sub - directories and so on). I changed the -X option to -X0, which would not automatically append the suffix ".lzx" to the archive filename - this seems to change nothing if using the '020 version (as long as you don't delete the suffix in the requester, but you wouldn't do this anyway as you can just click on the filename), but if using the '000 version it

will return an error. Strangely, the error returned is "Bad LZX archive header" - not surprising because it is operating on an LhA archive! The '020 version handles LhA files the same as LZX files. It seems that the '000 version in the LZX distribution doesn't have the registered features...

I had a cunning idea to use the Version command to determine which version of LZX is being used, but the only version information given by LZX is the version number.There are other ways of finding which version (like checking the size of the LZX executable using List or something), and changing the command line accordingly(ie using LhA to de-archive on an '000), but then how many people with '000s nowadays would be using this program? Thus I have decided that LhA2LZX now requires an '020 or faster CPU, as this is what most have nowadays.If you're still using an '000, write to me (snail-mail or e-mail)and I'll send you a version for it.

Another small change was just putting the path of LhA and LZX in speech marks, just in case your path has a space in (like if you keep them in "Workbench 3.0" or something).

I shall also note that LhA2LZX supports filecomments and filedates and so on of the files in the original archive -this means all comments etc will stay intact when converted between the two archive types. I am told not all the programs similar to LhA2LZX do this (although that may change after the other authors' read this...).

2.7b Not much change here in terms of code, but a bit in ideas. The suggestion came from Mosquito, and it seemed like a good idea to implement. The file comment, creation date and time of the new archive are replaced by the old - so the only noticable difference between your archives is the file type and the file size (and the name slightly, obviously!). Thus all file comments of the archives remain intact, and the old date used makes it look as if it was the new archive you downloaded/created in the first place! This was all done by using features of the List command - showing just how versatile it is!

Another tiny detail was changed - after converting to the other archive type, I used the suffix .(lha|lzx) (lha or lzx) to copy the new archive over and get its size:I used this format because you were converting to the other file type, which could have been lha->lzx or lzx->lha. Since writing that line I had added another variable somewhere along the way which would give the other file type suffix ($nsuf), so I altered the line to give the other suffix, instead of either because I could now determine which you had converted it to.

2.8 Well, I seemed to change a lot around in this release! Not much difference in operation, but in efficiency. I noticed that much of the code was repeating itself throughout (mainly the area where LhA and LZX were used, and the Insufficient Memory bit, and also the size comparisons of the old and new archives, as well as a few other areas). I was able to cut a lot of this out and make much of the code common to whichever operation was being carried out, thus making the whole thing much more efficient, and it looks a lot more streamlined too! It is a worthwhile job to compare this version with v2.7b, and see exactly how and where I have steamlined it. Part of the fact that there was a lot of similar code was that before v2.6, LhA was used to de-archive LhA

archives, whereas now LZX is used to de-archive both file types - meaning this bit is common to both de-archiving processes and can thus be put into a single process. Using the same code like this to do several processes (the same code is used whether you are converting to the other file type or to the smaller archive, and this is only taken into account at the end when deciding whether or not to replace the old archive) is a GOOD THING when writing code...of course, I never should have programmed it like that in the first place...:^)...but, hey, that's what I'm at University to learn...:^)

I managed to get the file size of LhA2LZX down to around 10k from 15k - which shows just how well I did in optimising it! I then added a few more comments though, which increased the size. Without comments, LhA2LZX is about half the size. I decided to use this as a separate release to show how effectively you can sometimes improve a script by making very small changes in the way it works,but large ones in the way it is written.

The only differences you may notice about this new version (apart from maybe some things being executed in a different order) is firstly that you no longer have to enter the path of the drawer the archive is in - you can just select the archive itself - and secondly that the destination of 'l2z' is now saved forever in ENVARC: so that whenever you run LhA2LZX after the first time, you won't have to choose where to create it, which saves time and hassle, unless you get the Insufficient Memory requester, in which case you get to choose a new location for it, which is then saved in place of the original. I changed this because that damn "Where should I store..." requester popped up every single time I wanted to convert an archive, and I always chose RAM:, so it seemed pointless and time-wasting to have to choose each time the same drawer. If you ever want to change the destination for 'l2z', just delete ENV:LhA2LZXl2zpath and run LhA2LZX again to choose another drawer. With the "Select the drawer the archive is in..." bit: I always found selecting the drawer and then getting a requester to select the file was a bit long-winded, but I thought it wasn't too bad if you were converting several files from the same drawer, as the drawer of the first archive was used as the default for the following archives. I found that using another option in List I was able to get the path of the drawer by simply stripping away the filename, which made things easier and less time-consuming. Is there nothing that this command won't do?

Another minor alteration: I changed the output in the Insufficient Memory part to tell you how much memory in Kilobytes instead of bytes you may need, as it's much easier to read and relate to like this. If you ever get the requester, the archive must be fairly big and so the large number telling you how many bytes of memory you may need won't mean much (I always found myself mentally inserting the commas!).

I changed a few of the 'Skip's to skip to different places I think, just so it would make a bit more sense when using it ... however, as I was working on this version I was also working on a parallel version for use within DOpus which would take an archive name at the command line so that you could select an archive in a DOpus lister and then click on an LhA2LZX button...and before I knew it, it started rapidly blossoming into the magnificently improved...

3.0 Where to begin?! LhA2LZX now accepts multiple files for input, can repack an archive to ensure minimum size for an archive type and can successfully be used in DOpus to select multiple files and work on them sequentially! It now creates several scripts of its own whilst running, and executes itself from these scripts! I shall explain: LhA2LZX will check to see on startup if any archives have been passed to it for processing (which it will have been if you are using it from DOpus), and if none have been given then a requester pops up asking you to select one or more archives for processing. It then lists these archives into another script - for example this new script would read:

```
execute "DH1:Toolsdaemon/lha2lzx/LhA2LZX" "AQ2:Temp/AGHTW.LZX"
execute "DH1:Toolsdaemon/lha2lzx/LhA2LZX" "AQ2:Temp/AIFFdt.LhA"
execute "DH1:Toolsdaemon/lha2lzx/LhA2LZX" "AQ2:Temp/AnimGIF.LZX"
```

and then this new script would be executed. As you can see, each line gives LhA2LZX an archive to process, and once it has processed the first it will move onto the second, and so on. If LhA2LZX is given an archive on startup like this, it will process only this archive before quitting. This script is created using a neat little function of the multi-talented List - telling it exactly how you want it to output the filenames it is given. You can see also that the path of LhA2LZX is given - this is saved as an env variable using the new installer script and so will change to wherever LhA2LZX is stored.

This version also saw a neat little trick concerning LZH archives - renaming them with an .lha suffix, as they're basically the same anyway, and can be processed as if they are LhA archives.

I've also made sure that every time a drawer name or file name is passed as an argument it is surrounded by quotes just in case it has a space in, like "Ram Disk:".

Another new neat feature is that of "Repack" - an archive will be de-archived and then re-archived to ensure that, if you want to keep the archive as the same file type, it is of the minimum size. This was achieved simply by adding about 5 lines of code! All I did was to swap the contents of the variables concerning the suffixes (so that LhA changed to LZX and vice versa) before the file is de-archived, so that LhA2LZX thinks the file is of the other type when re-archiving, and hence re-archives it with the appropriate program! The suffixes are then swapped back. Simple but effective! I added this option because when using DOpus to select files to put in an archive, the individual filenames are passed to LhA/LZX, and if you have many files it often creates the archive with the first few files, then adds the next few in a separate command, and then the next few and so on. This rarely affects LhA archive creation, but LZX archives are often reduced in size if all files are added in one go - in other words by a wildcard pattern. This is achieved by LhA2LZX doing what it has always done - de-archiving to a newly created temporary drawer and then re-archiving everything inside to a new archive.

DOpus archive creation:

```
lzx -e -3 -F a "RAM:lha2lzx2" "lha2lzx/LhA2LZX.guide.info"
"lha2lzx/LhA2LZX_v2.6" "lha2lzx/LhA2LZX_v2.6a" "lha2lzx/LhA2LZX_v2.5"
```

```
"lha2lzx/LhA2LZX_v2.4" "lha2lzx/LhA2LZX_v2.3" "lha2lzx/LhA2LZX_v2.2"
"lha2lzx/LhA2LZX_v2.1" "lha2lzx/LhA2LZX_v2.0" "lha2lzx/LhA2LZX_v1.2"
"lha2lzx/LhA2LZX_v1.1" "lha2lzx/LhA2LZX.guide" "lha2lzx/LhA2LZX_v2.7b"
"lha2lzx/LhA2LZX_v2.7" "lha2lzx/LhA2LZX_v2.7a" "lha2lzx/LhA2LZX_v2.8"
"lha2lzx/LhA2LZX.install" "lha2lzx/LhA2LZX" "lha2lzx/LhA2LZX.info"
lzx -e -3 -F a "RAM:lha2lzx2" "lha2lzx/LhA2LZX.install.info"
"lha2lzx2.readme"

LhA2LZX archive repacking:
lzx -e -3 -F a l2z/#?
```

You can see that the archive is first created by giving all the filenames to add, but the command line is too long and so LZX is called again to add the rest of the files. Once the archive has been created, it can be easily repacked by a wildcard pattern, which is often better due to the nature of how LZX works over LhA.

LhA2LZX is extensively commented now - almost every line in fact! You can now follow exactly what is going on line by line, and if you want to get the full benefit then read it as it is executing - so you can see what is being created, what it says and so on.

The final line "Quit" was removed, otherwise the whole process would quit after processing the first archive, when the script reached that line of execution. I moved the "Thanks for using..." farewell message too (as otherwise it would appear after every archive conversion) and it is now only reached at the end of the entire process, when all archives have been processed. If using from DOpus, all output like this is suppressed, as it looks better without it there. If you still want to see the files being de-archived, set the "Output to window" flag.

Almost all of the 'Skip's have been altered to - instead of skipping back for you to enter another filename if the one you gave doesn't exist, it skips to the end. This is because the filename you gave it is now integrated into LhA2LZX, and so after telling you it doesn't exist it will check to see if a filename has been entered already before asking you for another - and since one has been entered, you'll get stuck in a loop. The only time you will ever encounter this is if you enter a file's name that doesn't exist, and the only way you could do this would be to enter it by the CLI. But seeing as the whole point of LhA2LZX is for you to use it by the requesters or now from DOpus, it'll never happen :^) . The same would have happened if the file was not an LhA or LZX archive, but I changed that too so that if the file you give it is not an archive, LhA2LZX skips to process the next one.

The backtick is once again put to clever use. The executable archive listing file RAM:LhA2LZX.exe is created by passing the filenames given from the file requester into List, which then outputs the filenames as you saw above, ready for processing. You can see this here.

The four programs originally loaded into memory at the start are now checked to see if they are already in memory, in which case they won't need to be reloaded, and so aren't. This just makes things a little tidier, and also uses the backtick effectively.

3.0a I found a much neater way of getting the path of LhA2LZX, meaning you

don't need to install it now - it can be easily run from anywhere. In fact, I found 2 ways: the second way was much more efficient and did exactly the same job. Here were my notes for the first way:

```
echo `cd` >env:LhA2LZXpath NOLINE           ;This gets the current directory (i. ↩
   e. the drawer of LhA2LZX), e.g. "dh1:Temp"
echo "$LhA2LZXpath" FIRST 256 LEN 1 >env:nl2zpathtmp NOLINE ;This will get the  ↩
   last character of the drawer - if it is ":" then it is a root directory...
if $nl2zpathtmp EQ :                ;...and if it is...
  echo "LhA2LZX" >>env:LhA2LZXpath NOLINE       ;...we only echo "LhA2LZX" onto  ↩
     the end of the directory name...
else                        ;...otherwise...
  echo "/LhA2LZX" >>env:LhA2LZXpath NOLINE      ;...we have to make sure we go  ↩
     into the directory where LhA2LZX is
endif                       ;This first section joins the drawer of where LhA2LZX  ↩
   is found to the actual LhA2LZX file e.g. "dh1:Temp/LhA2LZX". This first part  ↩
   of the program merely gets the command line of LhA2LZX, and stores it in 'env ↩
   :LhA2LZXpath'
```

At the beginning, the current directory is echoed to a file, and then "LhA2LZX" is appended to that, depending on whether it a root directory or not (just read the comments for more info). I decided to submit LhA2LZX for inclusion on an Amiga Format cover CD and found it would be a little clumsy having to specify the path on there of LhA2LZX, and then suddenly had an idea of how to get the command path on startup instead, meaning you can run it from anywhere without having to install it first. Below is a paragraph which was intended after v3.0 for the next version of LhA2LZX, but is now no longer needed:

"As the path of LhA2LZX is now known, a small check is performed at the start just to make sure it is there. I kept forgetting to change the path back to my original LhA2LZX drawer when testing the install script, and after I deleted the newly installed version it wouldn't work from my original drawer (as the path specified by the installer had just been deleted)."

This also helps to explain why the new process is better. The check is still there, but you will now only experience it if you run LhA2LZX from the CLI and don't CD to the drawer LhA2LZX is in beforehand.

The "echo "$LhA2LZXpath" FIRST 256 LEN 1" line will give the last character of the current directory by starting at the last character and echoing it out - it takes the first character to start echoing as character number 256, but as this is the maximum length of a filename anyway you won't in all probability have this many characters in the filename and so it starts echoing at the last character in the filename. This will either give us ":" (for a root directory, e.g. RAM:) or the last character of the directory (e.g. "p" for dh1:Temp). This is important, because when we echo "LhA2LZX" we need to know if we have to descend into a directory first (i.e. echo "/LhA2LZX"). The filename of LhA2LZX is concatenated (joined) onto the directory to give the full command path of LhA2LZX. This is done by using the double redirection symbol >>. This means that the text to be echoed is echoed into the end of the file specified without replacing the file with the new text:

e.g.

```
echo "Hello" NOLINE >env:new
echo " World" >>env:new
```

The file env:new now reads thus:

```
Hello World
```

The NOLINE keyword tells Echo not to put a carriage return at the end of the line of input text, so we can keep everything on one line.

I then realised that there was a much easier way to get the command line of LhA2LZX using just one line instead of 7:

```
list LhA2LZX LFORMAT=%F%N >env:LhA2LZXpath
```

As you will already be in the current directory of LhA2LZX (there is a check that warns you if you are not, before quitting), I was able to just use List to list the full pathname of LhA2LZX. %F gives the disk and directory name, and %N gives the filename.

I have not included v3.0 in this archive because it only differs from v3.0a by not having the first 3 lines of the actual program (which includes the List line above), and also it has a small check at the beginning to see if LhA2LZX has been installed correctly:

```
if NOT EXISTS env:LhA2LZXpath     ;This checks to see if the path of the ↩
   location of where LhA2LZX is stored on disk has been saved to the file 'env: ↩
   LhA2LZXpath', which is done in the installation process
  RequestChoice >nil: "LhA2LZX Error Message" "LhA2LZX not installed using the ↩
     installer script!*nSee docs for details!" "Quit"
  quit              ;If the file isn't there, there's no point in carrying on...
endif
```

Now, if the path file does not exist then it is created.

3.1 This version sees much better handling of the tilde bug, by use of an AREXX script. AmigaDOS checks for a tilde in the filename as before, but if it finds one then it starts the AREXX script lha2lzx.rexx which searches through the filename and inserts an apostrophe before each tilde - so that the AmigaDOS program Rename can be used to rename the file with underscores instead of tilde's. The file will be renamed even if you do not perform any actions on it. I'm still learning AREXX, so in the future I will probably use this for checking if the new file will fit on the destination disk. lha2lzx.rexx will fail if the path of the archive contains spaces, but then you shouldn't really have them there in the first place :^)

I also found that I'd got 2 env variables which contained the same information (nl2zldfl1 and nl2zl), so I got rid of the former by using the latter in its place.

3.2 The docs always take longer to write than the actual program, so I'm going to keep this short :^). On the AmigaDOS side of things, all I did was get the protection bits of the old archive and transfer them to the new, use a small program 'inf' to get the remaining amount of disk space on the destination (so now if your new archive is too big to fit back over the old archive, a requester pops up telling you this and

asking you where you want the new archive to be saved to), and check if LhA and/or LZX are resident and use the resident ones if they are. The big improvement to this version (apart from the remaining destination size)was done by adding to the ever-growing AREXX script, lha2lzx.rexx. This does 3 new things: it alters the filecomment if it has any asterisks in (it changes them to plus signs, because previously if your filecomment ended with an asterisk then in the command line it would have read "<comment>*", and AmigaDOS interprets the asterisk followed by a quote character to mean "include the following quote character, don't treat it as the end of the argument"), it removes the unwanted Line Feed and Carriage Return characters from the end of the file created by 'inf' which gives the size of the remaining disk space on the destination disk, so that the file can be used as an env variable for use with Eval), and it takes the protection bits of the original archive and gets rid of any hyphens that it may contain so that only the actual protection bits remain - e.g. if you type

```
list ram:disk.info LFORMAT=%A
```

you will get an output of the protection bits of the file, like

```
----rw-d
```

- the protection bits tell you whether the file is deletable, executable and so on - just select the icon and choose "Information" from the "Icons" menu, and look at the check boxes in the upper right of the newly-appeared window. If you want to set these bits from the CLI, you have to use the Protect command:

```
protect ram:disk.info arwe
```

This would set the bits for "Archived", "Readable", "Writable" and "Executable". As you can see, there are no hyphens as List gives, and if you try the following command you will get an error:

```
protect ram:disk.info --arwe-
```

You cannot have hyphens in the Protect command line, which means that if you want to set them using the output from List, you have to get rid of the hyphens in some way - all I did in the AREXX script was to go through each character in the output individually and check if it was a protection bit like "a" or "d", and add that letter to an output string which was then written to a file.

The protection bits are:

H - Hidden
S - Script
P - Pure
A - Archived
R - Readable
W - Writable
E - Executable
D - Deletable

Frank Bunton once again deserves a mention here for his help with the AREXX - firstly for his excellent "AREXX For Beginners", and secondly

for giving some advice on parts of the procedures and Good AREXX Programming (TM)!

Didn't do much of a good job keeping this bit short, did I? ;^)

This .guide has also been extensively added to...although you may have to search for some of the hidden pages...as a treat for everyone who has got this far,why don't you have a small break and go on my treasure hunt by double-clicking on the word "hidden" above...

1.12 Hidden pages? Surely not...

So you think there are some hidden pages, eh? I always liked treasure hunts...

Clue One: The power of AmigaGuides is detailed in Frank Bunton's "How To Write An Amigaguide" ...the most comprehensive document on the subject...isn't he on the "Thanks to..." page...?

1.13 For the Future

For the Future

- Even safer archive handling by keeping backups.
- Erm...anything else you can think of!

1.14 Legal Stuff - Disclaimer

Legal Stuff

Standard disclaimer:

THERE IS NO WARRANTY FOR THE SOFTWARE TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHERE OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE SOFTWARE "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY REDISTRIBUTE THE SOFTWARE AS PERMITTED BELOW, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE SOFTWARE TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

1.15 What I Used

What I Used

BED - Black's Editor, to write LhA2LZX and also this AmigaGuide doc. It is not your average PD editor, so get it soon! It allows you to edit text/binary files, has a user-configurable menu bar, has a multitude of AREXX macros (and you can add your own) and can do just about anything you require - justification, clipboard support, encryption & viewing individual AmigaGuide pages as you write them are just a few of the options!

DiskSqueeze -an AmigaDOS script file which uses LZX to compress entire disks and so giving better compression than DMS! It also allows a filelist to be added to the archive, so that you know exactly which files are on the disk. DiskSqueeze helped me to understand how to structure a script file (although originally it was by messing around with DMSMaker that I learnt how to write scripts) and gave me a few ideas to improve LhA2LZX (for instance using the Which command, checking memory size and most notably about the use of Environment Variables).
Available on the AmiNET as util/arc/DiskSqueeze.lha

DOSMan - An AmigaDOS reference manual (in AmigaGuide and MUI format) which details the usage of commands up to and including the new WB 3.1 ones, and gives examples of how to use them.Invaluable if you don't have one, and essential to find out what all the CLI commands do and how to use them. Available on the AmiNET as util/wb/DosMan121.lha

Vimto - not anything to do with computers at all in any way, but the Fruit Juice Drink of Champions. Just thought it deserved a mention.

And, of course ...

LhA Evaluation V1.51
LZX 1.21 (Registered) Archive/Extract utility - 68020/68030 Version.

1.16 Thanks to . . .

"LhA2LZX has been brought to you today by ..."
Thanks to

Marco Negri for Black's Editor v1.02
Dirk Vael for DiskSqueeze v1.22

Peter Bagnato for DOSMan v1.21
Stefan Boberg for LhA v1.51
Data Compression for LZX v1.21r
Technologies
Jotes of Fraxion for his Ansi.Manual v0.9, allowing text clearing and coloured text!
Frank Bunton for his help with ways around the tilde bug
http://www.acay.com.au/~bunton/

Bill Duxbury for his help in finding the ’000 LZX bug
Jerry Sandstedt for the extra ideas implemented in v3.2, namely keeping the protection bits and pointing out the filecomment bug, and also finding if LhA and LZX are already resident

And finally...

to Sarah McLachlan, for singing to me for countless hours as I toiled away at my keyboard. Not personally, mind... (her piano wouldn’t fit in my bedroom).

1.17 Frank’s very own secret node!

Wahey! Told you I’d put it in. The only way to read this is either to double-click on your name in the "Thanks to..." node, or to read this .guide in PPMore or something. I wonder if I’m the first to ever do this? You can click on the Browse buttons all you want, but you’ll never get to this page!

I wrote the above as a secret page for Frank, to show the only practical use he could find for the "double-click on a word instead of having a button" thing. So far the only practical use seems to be this treasure hunt! Were Commodore anticipating this eventuality when designing the Amigaguide...?

Clue Two: Who’s piano wouldn’t fit in my bedroom?

1.18 Contacting the Author

Contacting the Author

If you have any suggestions, comments or bug reports, you can contact me at the following addresses:

e-mail: R.Burke-97@student.lboro.ac.uk

Snail-mail: c/o 37 Woodside Gardens
RAVENSHEAD
NOTTINGHAM
NG15 9GF
UK

E-mail is preferred if possible (but don’t expect a reply during the Summer months ... :^) ... and that means June - September for all you Southern - Hemispherists...).

If writing by snail-mail, please include a stamped addressed envelope.
As I was once told..."We don't get paid for writing this, you know!"

LhA2LZX is GIFTWARE. That means if you use it, send me a little something. It doesn't matter if it's a nice picture by e-mail, a link to a web site I might like, or even a 20p bar of chocolate (Mint's my favourite...). I don't care, as long as I know you are appreciating my small gift to you :^)

You can download the latest version from my LhA2LZX web page:
http://www-student.lboro.ac.uk/~corb1/lha2lzx.html

1.19 A secret page dedicated to the wonderful...Sarah McLachlan!

Sarah McLachlan - she's great!

Behold the second hidden page! Just thought I'd add a bit of info about one of my absolute favourite singer/songwriters. Bet you weren't expecting this!

Sarah McLachlan is a Canadian songstress with the most beautiful voice this side of the Moon. In the 1998 Grammy Awards she was nominated in 3 categories:

Best Female Pop Vocal Performance for Building A Mystery
Best Pop Instrumental Performance for Last Dance
Best Pop Album for Surfacing

and she won the first 2. Go Sarah!

She has released 7 albums so far:

Touch 1988 Recorded when she was only 19!

Solace 1991 Has some wonderful tracks, especially for those quiet moments.

Fumbling Towards Ecstacy 1993 Loads of the songs from here have been on "Due South", an excellent TV series about a Canadian Mountie. This is great to listen to on those dark, cold, long winter nights!

The Freedom Sessions 1995 Sarah and her crew decided one night, after drinking loads of red wine, to go into someone's basement and re-record "Fumbling..." as it originally sounded - much more acoustic. The sound of Sarah's vocals with just her guitar or piano is just breathtaking! And if that isn't enough, it's an enhanced multimedia CD as well as an album, so if you can run Quicktime movies you can see video interviews, and all the videos to all the songs from her first 3 albums! Bargain!

Rarities, B-Sides, 1996 I only got this Canada-only release recently, so I'm still listening to it. Some of the songs are great!

and Other
Stuff

Surfacing 1997 I've got this one playing at the moment! It's great. And
        it's an enhanced multimedia CD too, with more interviews
        and live versions of some of the songs and stuff.

Live 19?? Haven't got this yet. It's one of her earlier concerts,
        put on CD. Go to my life.html page to listen
        to a live RealAudio concert lasting an hour!

So now you know a lot more about Sarah and her albums than you used to. If anybody remembers what songs she performed on "Later...with Jools Holland" last time she was on ('93/'94 I think), I'd be grateful for any info (it's probably only people in the UK that have seen this, though...).

I've told you about Sarah, but should I even start on the marvellous Lisa?

If you want any more info about Sarah and others, go to my Music Emporium:
http://www-student.lboro.ac.uk/~corb1/music.html
where you can find links, audio files, movie files, an hour-long RealAudio concert and more!

Clue Three: Which wonderful singer have I not even started telling you
      about?

1.20 Also by the Author

Also by the Author

All the programs below are AmigaDOS script files created purely to show what you can achieve,using the software you already have,to make your life easier. Most of them are simply "front-end" programs to CLI executables- so you just click requesters instead of typing out CLI commands like in LhA2LZX.

AllowBad.req
  Do you have floppies with hard errors on? Do you find that you simply can't format them or use them properly? Then wipe away your tears, for AllowBad.req is here! A simple requester interface to Mikolaj Calusinski's AllowBad means you too can now live with the peace of mind that never again will you have a Bad Floppy! ®
DMSMaker
  A fab little AmigaDOS script file allowing you to use all the standard functions of the DiskMasher System - including read, write, view, test, repack, and even encrypt your DMS files with an invisible password!
DSQView
  A small file which finds the filelist in a DiskSqueeze archive and displays it.
FillDisk
  Helps fill disks to their maximum, using Torbjörn Hultén's "Fill v2.0." Choose your files by requester, and Fill calculates which combination of files to use to fill the disk to its maximum. Useful for storage.
HappySearch

I once had a virus which adds itself to every file you open, and is recognisable by finding "Happy New Year 96!" in your files. HappySearch will search specified files to see if you have the virus.

LhAMaker
This creates LhA archives from specified input files, via requesters.

UnLZXer
This de-archives LZX archives to a chosen destination.

You can download these from my LhA2LZX web page.

1.21 Archiving

Archiving

Archiving means to backup files. In the case of LhA & LZX, all the files you wish to backup -store a copy somewhere in case you need them in the future to, for example, replace lost data -are merged into one archive file and compressed, so that in almost all cases the final archive file takes up less space than the files which were archived did.

See also De-archiving

NB "De-/archiving" in this document refers to when LhA2LZX de-archives and then archives an archive.

1.22 De-archiving

De-archiving

De-archiving means to restore files from a backup archive. Files are decompressed and separated back to individual files.

See also Archiving

1.23 arexx

AREXX is an interpreted programming language - which means it is not compiled like normal programs but is instead read in as a script and each line is executed as it is reached - in a similar way to an AmigaDOS script. AREXX can be used to provide a 'back-door' into many programs, allowing you to easily give the program instructions for an operation - for instance loading a drawer of GIF pictures into an image processing program, telling it to convert them to greyscale and to make them half the size, and then saving them as JPEGs. It can also be used to create standalone programs, such as getting the user to input the year and then calculating and printing out a simple calendar.

AREXX comes with the Workbench distribution.

1.24 The Command Line Interface

CLI

The Command Line Interface allows the user to communicate to the computer via the keyboard. The Shell is the Amiga's CLI.

1.25 LhA & LHZ

LhA is a utility which allows you to compress and archive files. It was a very popular program for backup and distribution purposes (i.e. the AmiNET) but is now being surpassed in terms of decompression times and archive size by LZX.

LhA archives can be compressed further. If you archive all your files into an LhA archive with no compression, then archive this with LhA with compression, you can often save quite a lot more space, although you will have to de-archive it twice. I have not included this facility in LhA2LZX, because the same does not work with LZX,and it would have no real indication how many times the archive had been re-archived, and may get too complex to be useful. But it's nice to know anyway :^)

LHZ is similar to LhA and files archived with this are de-archived by both LhA and LZX.

1.26 LZX

LZX is a utility which allows you to compress and archive files with often a great degree of compression. It also decompresses data faster than any other popular Amiga program. It gives greater compression because it uses the knowledge of previous files in the archive,using an ability to transparently look for similar compressible data across multiple files before adding them to an archive.

See also LhA/LZH

1.27 RequestChoice

RequestChoice brings up a requester in which the user is prompted to make a choice.The title-, body- and button-texts are configurable so that it may be used in any situation. The value of the button pressed by the user is then returned so that the relevant process is started.

See also RequestFile

1.28 RequestFile

RequestFile brings up a filerequester in which the user is prompted to choose a file.The title- and button-texts are configurable so that it may be used in any situation. The chosen file name is then returned so that it may be used later.

See also RequestChoice

1.29 Index

Program History -
v1.0 v1.1 v1.2
v2.0 v2.1 v2.2 v2.3 v2.4 v2.5 v2.6 v2.6a v2.7 v2.7a v2.7b v2.8
v3.0 v3.0a v3.1 v3.2