lzx

Jonathan Forbes

COLLABORATORS

	TITLE : lzx	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Jonathan Forbes	January 13, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

lzx

1.1 LZX Documentation

LZX - The New Amiga Archiver

Version 1.21

1.2 Introduction

INTRODUCTION

LZX is a brand new data compression program developed on and for the Amiga, sporting mind-blowing compression rates, and brain-staggering speed.

LZX is a co-development between
Jonathan Forbes
(author of the LZ and LX
archivers, as well as the commercial Bulletin Board Software package
Xenolink), and
Tomi Poutanen
.

LZX uses compression technology that is absolutely state of the art, and offers a compression/performance ratio superior to that of all other programs for the Amiga, as well as to all programs for MS-DOS.

1.3 About Jonathan

ABOUT JONATHAN

Jonathan is 22 years of age, and is a fourth year Computer Engineering student at the University of Waterloo. The Computer Engineering curriculum is a continuous 5 year co-op programme (no holidays) including 6 four month terms of employment and the standard 4 academic years.

Jonathan has worked on contract for Communications Canada, where he developed a real-time head and eye tracking software library, and for Northern Telecom for 16 months, where he was involved with the development of Northern Telecom's Meridian IVR (Interactive Voice Response) Fax Response software.

Jonathan was born in the United Kingdom, but moved to Canada in 1986, where he abandoned his BBC Microcomputer 32K for an Amiga 1000. In 1988 he learned C, and in 1989 he learned 68000 assembly language, and he has participated heavily in the evolution of Amiga data compression programs ever since; see the
archiver chronology
for details.

Jonathan is the author of the
Lhunarc

LZ
and
LX
compression programs, as
well as Xenolink, the commercial Bulletin Board Software package.

Jonathan currently owns an Amiga 3000 with a 35 MHz 68040 card (and not much else), and his original Amiga 1000.

1.4 About Tomi

ABOUT TOMI

Tomi is a Finnish citizen currently residing in Canada. He was born into a Swedish speaking family in Finland in 1972. Besides Finland and Canada, Tomi has lived in Germany, Austria, Kuwait, Cameroon and the U.S. He emmigrated to Canada in 1985 and has resided in Toronto ever since.

Tomi is a fourth year Computer Engineering student at the University of Waterloo. As part of the co-operative work programme, he has worked for Reuters, Honeywell, and as a Software Design Engineer and a Program Manager for Microsoft.

Since its inception, LZX has been developed simultaneously on the Amiga and the PC. The compression related research was conducted by both Jonathan and Tomi in portable C, and once complete, was separated for the two platforms. Jonathan is responsible for development on the Amiga platform, and Tomi is responsible for the PC platform.

As of the release of this Amiga version, Tomi is trying to negotiate the hardships of the MS-DOS platform, such as the 64K segments and the lack of usable RAM. The Amiga has proved to be a much more efficient and flexible platform to develop under. As a result, the PC version is several months behind, requires the development of new technology, and as a result, may ultimately have to sacrifice performance to operate efficiently under the platform restrictions. Fortunately, the Win95 and WinNT platforms have overcome the aformentioned shortcomings and as a consequence, the archiver will first be built around Win32 technology, while providing a DOS decompressor. A DOS archiver will follow at a later date.

For development, Tomi uses a 486/50DX2 with 24 MB RAM running Windows NT 3.51.

1.5 System Requirements

SYSTEM REQUIREMENTS

LZX will run on an Amiga with any 680x0 CPU and AmigaDOS 1.2 or later, although it prefers AmigaDOS Release 2 or later. If running on AmigaDOS 1.2 or 1.3, LZX will attempt to open
arp.library
to emulate functionality
missing from the early versions of AmigaDOS; if arp.library is then not available, some
functionality
will be missing from LZX.

By default, LZX requires approximately 500K of RAM to archive files, and this includes the size of the LZX executable and the stack. To decompress files, LZX requires approximately 175K of RAM, and this also includes the size of the LZX executable and the stack. LZX's command line options allow it to be configured to use
less memory

, however.

Three versions of LZX are provided, to support the 6 different CPU's in the Motorola 680x0 series. Each version has been hand-crafted to provide optimal performance for two particular CPUs.
More CPU Info

1.6 More information on CPU dependent versions of LZX

CPU INFO

Versions are provided for the 68000/68010, the 68020/68030, and the 68040/68060, and are named LZX_68000EC, LZX_68020, and LZX_68040, respectively. In addition, only the 68000/68010 version supports AmigaDOS 1.3; the 68020/68030 and 68040/68060 versions do not.

Users with a 68000 or 68010 CPU can run only the 68000/68010 version, while users with a 68020-68060 CPU can run any version if they have AmigaDOS Release 2 or later. LZX has been carefully hand-crafted for each CPU; performance will be degraded somewhat if running a version not optimised for your CPU.

Unlike most programs which provide CPU depdendent versions, LZX was designed around the instruction set of the 68020-68060, which required that the 68000/68010 version be specially written to use only 68000/68010 instructions.

The 68000/68010 version has all of the functionality of the other versions, but receives the "EC" (economy) designation to identify it as being somewhat different internally. The 68000/68010 version runs a little more slowly than the other versions, but not much more slowly.

1.7 arp.library

ARP.LIBRARY

The arp.library was created by individuals as part of the AmigaDOS Resource Project (but not affiliated with Commodore in any way) for AmigaDOS Release 1 (versions 1.2 and 1.3). Its intent was to provide additional functionality not present in AmigaDOS at that time, including the developer-friendly wildcarding functions and date<->string functions that LZX uses.

All of these features (and more) were added in AmigaDOS Release 2, making arp.library redundant there. However, for users running AmigaDOS 1.2 or 1.3, arp.library provides very useful functionality for software developers.

1.8 Missing functionality due to lack of arp.library

MISSING FUNCTIONALITY

If running AmigaDOS 1.2 or 1.3 and arp.library is not present, LZX will not support wildcarding or directory recursion. For example, the following commands would not work:

```
lzx a cdir.lzh c:#?
lzx -x -r a test.lzh #?.info
```

1.9 Lower memory usage

LOWER MEMORY USAGE

The
-bi
and
-bo
options allow one to set LZX's input and output buffer sizes, respectively, and the
-w
parameter allow one to specify LZX's work directory, which defaults to "T:", which is usually assigned to "RAM:T".

Portions of LZX are currently being improved to use significantly less memory.

1.10 Glossary

GLOSSARY

Glossary of terms and archiving programs:

ARJ

Compression

Compression ratio

CRC

Decompression

File merging

Lharc (Amiga)

LHARC (DOS)

LharcA

LhA (Amiga)

LHA (DOS)

Lhunarc

LX

LZ

LZX

PKZIP

1.11 ARJ (DOS)

ARJ

ARJ is an archiving program for MS-DOS. Its main attractions are its huge array of features, and easy-to-use implementation of multi-volume archiving.

It is generally slower and less compressive than PKZIP.

There is no ARJ archiver for the Amiga, although there is an UNARJ program which can decompress ARJ archives.

Author: Robert Jung

1.12 Compression

COMPRESSION

The process by which data is represented in a more compact fashion by eliminating redundancies.

1.13 Compression ratio

COMPRESSION RATIO

There are two popular ways of representing a compression ratio as a percentage; either as the size of the input divided by the size of the output (times 100), or as 100% minus the aforementioned ratio.

LZX uses the former method, since it is more intuitive; if a file has a compression ratio of 35%, then it was compressed to 35% of its original size. A program which didn't compress at all will have a compression ratio of 100%.

Note that programs such as LZ and LhA use the second type of ratio, so a
file which compressed to 35% of its original size would be represented
by a compression ratio of 65%.

1.14 CRC

CRC

CRC stands for Cyclic Redundancy Check, and is generally superior to a
checksum for checking data integrity. LZX uses a 32-bit CRC for greater
error detection, rather than the 16-bit CRC used by the LHA programs.

1.15 Decompression

DECOMPRESSION

Decompression is the process of converting compressed data back into its
original form.

1.16 File merging

FILE MERGING

File merging is a feature of LZX which enables compression to be increased,
often very significantly, by allowing data from one file to be compressed
using the knowledge of previous files in the archive. This feature, unique to
LZX on the Amiga, often improves compression by 300% or more!
Example
This feature is very useful when compressing text files and source ↩
files,
where there is often a large amount of text common to many files.
Example
There is also a significant advantage to this feature when ↩
compressing
a large number of small files, since the data overhead of re-starting
compression for each file is now removed. This is particularly true when
compressing ".info" files.
Example
Even decompression time is improved, since there is now much less ↩
compressed
data to procress.

There are only minor disadvantages to file merging, which appear when deleting
files from a merged group, or extracting only some (but not all) files from a
merged group.

To see the advantages of file merging first-hand, try recompressing the LZX
distribution archive with LZX; LZX will compress the three executables (for
the 68000/68010, 68020/68030, and 68040/68060) down to a tiny size because
they are fairly similar.

1.17 File merging disadvantages

FILE MERGING DISADVANTAGES

When extracting only some files (but not all) from a merged group, all of the files in the merged group up to the last one to be extracted, must be decompressed internally. Luckily, LZX is obscenely quick at decompressing files, so this is not a significant setback. Example

When deleting files from a merged group, the whole group must be decompressed internally, and the files which are not to be deleted must be recompressed. Deleting a small file from a very large merged group can be time consuming, which is why LZX, by default, limits the maximum size of a merged group.

1.18 Deleting from a merged file group

EXAMPLE OF DELETING FROM A MERGED FILE GROUP

Assume that the following files were merged into a single group within the archive:

File1
File2
File3
File4

In order to delete File3, the whole group (File1, File2, File3, File4) would have to be decompressed internally, and then File1, File2, and File4 would be recompressed. The decompression and recompression (if necessary) is performed transparently by LZX when deleting from archives.

1.19 File merging example

EXAMPLE OF FILE MERGING

For example, in the archive "XPRZ31.LHA" (xprzmodem.library, version 3.1), 4 versions of the xprzmodem.library are distributed, comprising a total of 90932 bytes:

* One for the 68000/68010 with AmigaDOS Release 1.x
* One for the 68000/68010 with AmigaDOS Release 2.x
* One for the 68020-68040 with AmigaDOS Release 1.x
* One for the 68020-68040 with AmigaDOS Release 2.x

While LhA 1.50r compresses these four files to 52985 bytes, LZX compresses *all four* to an unbeliveable 17148 bytes, only marginally more than the 12758 bytes it took to compress just one of the four files. This is due to LZX's ability to transparently look for compressible data across multiple

files, before adding them to an archive.

1.20 File merging example

EXAMPLE OF FILE MERGING

For example, all of the header (".h") files in the SAS/C 6.5x include directory, comprising 980K of data, compress to 345K with LhA 1.50r, and to 254K with LZX.

1.21 File merging example

EXAMPLE OF FILE MERGING

For example, when compressing the icons as part of the Magic Workbench distribution, LhA 1.50r compresses 352K of icon data to 90K, while LZX compresses the data down to 62K.

1.22 PKZIP (DOS)

PKZIP

PKZIP is an archiving program for MS-DOS. Its main attractions are its speed and good compression. It has long been the market leading program for DOS.

InfoZip, a PKZIP-compatible archiver, is available for the Amiga, as is Unzip, a decompressor for ZIP files.

Author: Phil Katz (PKWare Inc.)

1.23 LHARC (DOS)

LHARC

LHARC is one of the original archiving programs for DOS. LHARC compression is supported on the Amiga in the form of the "-lh1-" compression format, which is supported by Lharc, LZ, LX, and LhA.

The second version of LHARC was renamed LHA.

Author: Haruyasu Yoshizaki

1.24 Lharc

LHARC (Amiga)

Lharc was the first program to support the -lh1- (LZH) format on the Amiga, and started the rapid move away from .arc and .zoo files. Although Lharc was later superseded by other compatible utilities, it deserves credit as the first LZH program for the Amiga.

Author: Paolo Zibetti

1.25 LhArcA

LHARCA

LhArcA is an archiving program supporting the -lh1- (LZH) format. When released, it was far faster than its only competition, Amiga Lharc. It was later supplanted in its position by LZ.

Author: Stefan Boberg

1.26 LHA (DOS)

LHA

LHA is the second version of LHARC for DOS. The LHA format is supported on the Amiga in the form of the "-lh5-" compression modes supported by LZ, LX, and LhA.

Most popular compression programs in use today were originally based on the source code or ideas of "AR002" (an LHA compatible program), although most programs have advanced since then.

Author: Haruyasu Yoshizaki

1.27 LhA

LHA

LhA is an archiver for the Amiga, supporting both the -lh1- (LZH) and -lh5- (LHA) compression formats.

Since its inception in late 1991, LhA has been the fastest archiver for -lh1- and -lh5- files, although it was supplanted by LX in 1993 for the fastest decompression of these files.

Author: Stefan Boberg

1.28 Lhunarc

LHUNARC

Lhunarc is a decompressor for -lh1- (LZH) files, and was the first program to provide high speed decoding (for that era) on the Amiga. Lhunarc was the precursor to LZ 0.80.

Author: Jonathan Forbes

1.29 LX

LX

LX is a decompression only utility, supporting both the -lh1- (LZH) and -lh5- (LHA) compression formats.

LX's claims to fame are a small code size (16K) (ideal for installation applications), asynchronous disk i/o (ideal for floppy drive applications), and fast decompression (the fastest available for LZH or LHA files).

Author: Jonathan Forbes

1.30 LZ

LZ

LZ is an archiver for the Amiga, supporting both the -lh1- (LZH) and -lh5- (LHA) compression formats. It was the first program for the Amiga to support the -lh5- format.

When it was first released in early 1990, LZ was far faster and more compressive than its competition, Lharc and LharcA, and retained this position throughout its development history, until the end of 1991.

Author: Jonathan Forbes

1.31 LZX

LZX

LZX is the latest and greatest archiver for the Amiga. LZX is the most compressive utility available, sporting some revolutionary new ideas in data compression technology. It also decompresses data faster than any other popular Amiga program, including LX and LhA.

Authors: Jonathan Forbes and Tomi Poutanen

1.32 Archiver chronology

ARCHIVER CHRONOLOGY

The chronology of the popular archivers is most interesting, consisting of programs continuously improving in performance and in compression. To date, all programs which became popular on the Amiga originated on the PC. It would be quite the irony if the PC port of LZX, an Amiga program, took over from the market leaders of the PC!

Program name	Platform	Date	Formats	Author(s)
PKZIP 1.x	DOS	?Q 1989	ZIP1	Phil Katz
Lharc 1.x	DOS	2Q 1989	LH1	Haruyasu Yoshizaki
Lharc 0.50	Amiga	3Q 1989	LH1	Paolo Zibetti
Lhunarc 0.90	Amiga	1Q 1990	LH1	Jonathan Forbes
LharcA	Amiga	1Q 1990	LH1	Stefan Boberg
LZ 0.80	Amiga	2Q 1990	LH1	Jonathan Forbes
LHA 2.02a	DOS	4Q 1990	LH1,LH5	Haruyasu Yoshizaki
ARJ 1.x	DOS	4Q 1990	ARJ	Robert Jung
LZ 1.80	Amiga	1Q 1991	LH1,LH5	Jonathan Forbes
ARJ 2.x	DOS	3Q 1991	ARJ	Robert Jung
LhA 1.00	Amiga	4Q 1991	LH1,LH5	Stefan Boberg
PKZIP 2.x	DOS	4Q 1992	ZIP2	Phil Katz
LX 1.00	Amiga	1Q 1993	LH1,LH5	Jonathan Forbes
LZX 1.00	Amiga	1Q 1995	LZX	Jonathan Forbes and Tomi Poutanen

1.33 Commands

COMMANDS

LZX supports commands standard on most Amiga archivers.

Entering "LZX" by itself from the command line provides a list of available commands and options (
Example
).

Each option will have one or two letters beside it in parentheses; either "(ax)", "(a)", or "(x)". These denote whether the option is used when archiving and extracting, only when archiving, or only when extracting, respectively. Note that in this case, archiving means adding, updating, freshening, or replacing.

Example
Command Descriptions

a
Add file(s) to archive

af
Fast add file(s) to archive

c

```
Concatenate archive(s)

d
Delete file(s) from archive

e
Extract file(s) from archive

f
Freshen file(s) in archive

l
List file(s) in archive

p
Print file(s) in archive to screen

r
Replace file(s) in archive

t
Test file(s) in archive

u
Update file(s) in archive

v
List file(s) in archive [Verbose]

x
Extract file(s) from archive [With full path]
```

1.34 Example command line display

```
LZX 1.00 (Evaluation) Archive/Extract utility - 68040/68060 Version
Copyright © 1995 Data Compression Technologies.  All rights reserved.
Commercial use of this unregistered program is prohibited

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

<command>:
  a  Add file(s) to archive              r    Replace file(s) in archive
  d  Delete file(s) from archive         t    Test file(s) in archive
  e  Extract file(s)                     u    Update file(s) in archive
  f  Freshen file(s) in archive          v[n] List file(s) [verbose]
  l  List file(s) [terse]                x    Extract file(s) with full path

<options>:
 -a (ax) Preserve file attributes       -q (ax) Configure console output
 -bi(ax) Set input buffer size (Kb)     -r (a ) Recurse into subdirectories
 -bo(ax) Set output buffer size (Kb)    -R (ax) Collect archives recursively
 -c (ax) Confirm files/archives         -s (a ) Add only files with no A bit
 -C ( x) Clear arc (A) bit on extract   -S (a ) Set A bit on added files
 -e (a ) Archive empty directories      -u (ax) Make file names upper case
```

```
 -E ( x) Touch extracted files          -U (ax) Set update rate (Kb)
 -f (ax) Ignore filenotes               -w (a ) Set work directory
 -F (ax) Fast progress display          -x (a ) Preserve path names
 -k ( x) Keep partial extractions       -X (ax) Control .LZX suffixing
 -l (ax) Make file names lower case     -y (a ) Store files with ratio >= x%
 -m (ax) Disable interactivity          -Y (a ) Compress archives
 -M (a ) Set merge group size limits    -0 (a ) Store files
 -o (ax) On or after date (yyyy/mm/dd)  -1 (a ) Fast compression
 -O (ax) On or before date (yyyy/mm/dd) -2 (a ) Default compression
 -p (ax) Pause after loading            -3 (a ) Maximum compression
 -P (ax) Set task priority              -- (ax) Stop further option parsing
```

1.35 Example

For example, the "-C" option (Clear arc (A) bit on extract) has "(x)" beside it, indicating that the option is meaningful only when extracting (or testing) files.

The "-1" option (Fast compression) has "(a)" beside it, indicating that it is meaningful only when compressing files.

The "-P" option (Set task priority) has "(ax)" beside it, meaning that it is meaningful both when extracting and when compressing.

1.36 Option Usage

OPTION USAGE

LZX options may be specified anywhere on the command line, and begin with the dash (-) character. The option letter(s) and any optional parameter(s) should follow the dash character.

Example	Comment
LZX a testfile.lzx dh0:test	No options
LZX -x x testfile.lzx	-x option
LZX x -x testfile.lzx	-x option (same effect as above)
LZX x testfile.lzx -x	-x option (same effect as above)
LZX -bi32 x testfile.lzx	-bi option with parameter "32"
LZX -x -a0 a testfile.lzx dh0:test	-x and -a0 options
LZX a -x testfile.lzx dh0:test -a0	-x and -a0 options (same as above)

Since LZX allows options to appear anywhere on the command line, the following command could be considered ambiguous: "LZX a testfile.lzx -testfile2". In

this case one is actually trying to add a file which has a name starting with the dash (-) character. LZX would take "-testfile2" to be the "-t" option with "estfile2" as a parameter.

To solve this problem, the double dash (--) option exists; LZX will not parse any options on the command line after a double dash. To correctly perform the above command, one would use: "LZX -- a testfile.lzx -testfile2". The double dash, like any other option, can be placed anywhere on the command line, but must appear before the "-testfile2" parameter.

1.37 Options

COMMAND LINE OPTIONS

Option usage

Option descriptions

1.38 Option Descriptions

-a
Preserve file attributes

-bi
Set input buffer size

-bo
Set output buffer size

-c
Confirm files/archives

-C
Clear arc (A) bit on extract

-e
Archive empty directories

-E
Touch extracted files

-f
Ignore file notes

-F
Fast progress display

-k
Keep partially extracted files

```
-l
 Make file names lower case

-m
 Disable interactivity

-M
 Set maximum merge group size

-o
 On or after date (yyyy/mm/dd)

-O
 On or before date (yyyy/mm/dd)

-p
 Pause after loading

-P
 Set task priority

-q
 Set quiet mode

-Qf
 Faster compression

-r
 Recurse into subdirectories

-R
 Collect archives recursively

-s
 Add only files with no arc (A) bit

-S
 Set arc (A) bit on added files

-u
 Make file names uuper case

-U
 Set console update rate

-w
 Set work directory

-x
 Preserve path names

-X
 Control '.LZX' suffixing

-y
 Store files with ratio > x%
```

```
-Y
 Compress archives

-0
 Store all files

-1
 Fast compression

-2
 Default compression

-3
 More compression

-9
 Maximum compression
```

1.39 Preserve file attributes

```
'-a' - PRESERVE FILE ATTRIBUTES

If this option is enabled, LZX will store (when archiving) and restore (when
extracting)
 file attributes
.

This option is ENABLED by default.  To disable it, enter '-a0' on the command
line.  When this option is disabled, files added to, or extracted from
archives will have the '----rw-d' attributes.

In order to ensure that file attributes are properly preserved, preservation
of file attributes must be active both when the files were archived and when
they are extracted.

File attributes will be properly preserved and translated across all
platforms for which LZX is available.

Note that by default LZX will clear the archive attribute ('a') for any file
extracted; enter
'-C0'
on the command line to disable this feature.


VALIDITY

This option is valid with the following commands:

   a (add)
   e (extract)
   f (freshen)
   r (replace)
   u (update)
   x (extract)
```

1.40 Set input buffer size

'-bi' - SET INPUT BUFFER SIZE

This option sets LZX's input buffer size. The input buffer size affects decompression only.

The input buffer size determines the amount of compressed data which LZX will read from disk at one time. For example, if the input buffer size were 8K, then LZX would read compressed data from disk 8K at a time.

Larger buffer sizes yield faster performance, but require more memory. The default buffer size is 64K, which is quite sufficient for almost all systems. Users with exceptionally fast CPU's and hard drives may wish to increase the buffer size further, while users with much slower CPU's and hard drives can probably lower the buffer size without significantly affecting performance.

VALIDITY

This option is valid with the following commands:

- e (extract)
- t (test)
- x (extract)

1.41 Set output buffer size

'-bo' - SET OUTPUT BUFFER SIZE

This option sets LZX's output buffer size. The output buffer size affects both compression and decompression performance.

DECOMPRESSION

For decompression (extraction), the output buffer size determines the amount of data which LZX will buffer in memory before writing to disk; for example, if the output buffer size were 8K, then LZX would write decompressed data to disk in 8K pieeces.

Because of the way in which decompression is done, using a larger output buffer will always improve the speed of decompression (up to the point where the buffer size equals the size of the largest file or solid file group; increasing the input buffer size beyond this size will have no effect on speed).

Buffer sizes as low as 8K are generally quite inefficient and should be used only when the amount of available memory is very low. Buffer sizes above 256K are highly excessive. LZX's default output buffer size is 64K.

COMPRESSION

For compression (archiving), the output buffer size determines the amount of compressed data which LZX will buffer in memory before writing to disk. For example, if the output buffer size were 8K, then LZX would write compressed data to disk 8K at a time.

Larger buffer sizes yield faster performance, but require more memory. The default buffer size is 64K, which is sufficient for most systems. Increasing the output buffer size can have a significant positive effect on compression time, if you can spare the extra memory.

VALIDITY

This option is valid with the following commands:

- a (add)
- e (extract)
- f (freshen)
- r (replace)
- t (test)
- u (update)
- x (extract)

1.42 Confirm files

'-c' - CONFIRM FILES

If this option is enabled, LZX will ask for confirmation for operation on any file or archive.

For example, if extracting files with a command such as 'lzx -c x archive.lzx', then LZX will prompt you for each file in the archive 'archive.lzx', asking whether you wish to extract the file; for example:

```
  Extract 'testfile'? (Yes/No/All/Quit):
```

In response to the question, you may enter 'Y' (extract the file), 'N' (do not extract the file), 'A' (extract all further files in this archive), or 'Q' (quit extracting from this archive).

If operating on multiple archives, LZX will also prompt for which archives should be operated on; for example, if the command 'lzx -c x *.lzx' was used (which tells LZX to extract files from all archives ending in '.lzx') then you might be confronted with something similar to the following:

```
Extract files in ARCHIVE 'testarc.lzx'? (Yes/No/All/Quit): n
Extract files in ARCHIVE 'johns_archive.lzx'? (Yes/No/All/Quit): y
  Extract 'file001'? (Yes/No/All/Quit): n
  Extract 'testing'? (Yes/No/All/Quit): n
  Extract 'hello.txt'? (Yes/No/All/Quit): y
```

Note that when extracting from multiple archives, there are two levels of prompts; the prompts which ask whether any files should be extracted from a particular archive (the 'archive prompt'), and the prompts which ask which files to extract from that archive (the 'file prompt').

If at a 'file prompt', the 'A' (extract all) and 'Q' (quit) commands affect only the current archive; these responses would cause all files from the current archive to be extracted, or quit processing of the current archive, respectively.

If at an 'archive prompt', the 'A' (extract all) prompt will cause LZX to assume a 'Y' (yes) response for all further archives. However, you will still receive the 'file prompt' for files in these archives. Entering 'Q' (quit) will cause LZX to terminate immediately.

This option is disabled by default.

VALIDITY

This option is valid with the following commands:

- a (add)
- e (extract)
- f (freshen)
- l (list)
- r (replace)
- t (test)
- u (update)
- v (view)
- x (extract)

1.43 Clear arc (A) bit on extract

'-C' - CLEAR ARCHIVE BIT ON EXTRACT

If this option is enabled, LZX will unset the archive ('a') protection bit for all files extracted. This feature is provided for use with hard drive backup programs, which assume that any file with the archive protection bit has not been changed since the last backup.

This option is ENABLED by default; therefore, by default, the archive protection flag is not preserved on extraction even if the '-a' option is used.

To disable this option, enter '-C0'.

VALIDITY

This option is valid with the following commands:

- e (extract)
- x (extract)

1.44 Archive empty directories

'-e' - ARCHIVE EMPTY DIRECTORIES

If this option is enabled, LZX will archive empty directories. An empty directory is a directory with no files or subdirectories in it.

This option is disabled by default; empty directories will not be added to archives.

VALIDITY

This option is valid with the following commands:

- a (add)
- f (freshen)
- r (replace)
- u (update)

1.45 Touch extracted files

'-E' - TOUCH EXTRACTED FILES

If this option is enabled, LZX will set the file modification date of all extracted files to the current date and time.

This option is useful if you perform hard drive backups using the file date as an indicator of whether has been changed, as opposed to using the archive ('a') protection bit.

This option is disabled by default.

VALIDITY

This option is valid with the following commands:

- e (extract)
- x (extract)

1.46 Ignore filenotes

'-f' - IGNORE FILENOTES

If this option is enabled when compressing, then LZX will not store filenotes in the archive. If enabled when extracting, then LZX will not restore filenotes to the extracted files.

There is no need to disable filenotes, even if archiving for other platforms; if the other platform does not support filenotes, LZX (on the other platform) will simply ignore the filenote.

This option is disabled by default.

VALIDITY

This option is valid with the following commands:

- a (add)
- e (extract)
- f (freshen)
- r (replace)
- u (update)
- x (extract)

1.47 Fast progress display

'-F' - FAST PROGRESS DISPLAY

If this option is enabled, then LZX will not output a line feed (ASCII 10) on the console after each file has been compressed or decompressed. Since scrolling the display can take a substantial amount of time (often more than the amount of time required to compress or decompress the file, for small enough files or fast enough CPU's!) this option can greatly decrease the time required for LZX to perform its operations.

LZX will still emit line feeds when listing the archives processed, so that all processed archives can be seen.

If errors occur on any file, LZX will emit a line feed for that file so that its information remains visible.

This option is disabled by default, but is highly recommended.

VALIDITY

This option is valid with the following commands:

- a (add)
- e (extract)
- f (freshen)
- r (replace)
- t (test)
- u (update)
- x (extract)

1.48 Keep partially extracted files

'-k' - KEEP PARTIALLY EXTRACTED FILES

This option causes LZX to keep as much as it can of files which are only

partially extracted. A file may be partially extracted because of a corrupted archive, a disk full error, or a user abort.

VALIDITY

This option is valid with the following commands:

- e (extract)
- x (extract)

1.49 Make file names lower case

'-l' - MAKE FILE NAMES LOWER CASE

This option causes all file names to be converted to lower case.

If adding files to archives, the names of all files added will be stored in lower case. If extracting from archives, all files will be extracted using lower case file names. If viewing or listing archives, all files will be shown as having lower case file names even if this is in fact not the case.

VALIDITY

This option is valid with the following commands:

- a (add)
- e (extract)
- f (freshen)
- l (list)
- r (replace)
- t (test)
- u (update)
- v (view)
- x (extract)

1.50 Disable interactivity

'-m' - DISABLE INTERACTIVITY

This option causes LZX to not prompt for the response to a question; instead, LZX will default to a sensible response.

When LZX would normally prompt asking whether a file should be overwritten, this option will cause the answer to automatically be "yes".

This option is enabled automatically if LZX detects that standard input is not interactive (for example, if it is run in the background).

VALIDITY

This option is valid with the following commands:

   a (add)
   e (extract)
   f (freshen)
   l (list)
   r (replace)
   t (test)
   u (update)
   v (view)
   x (extract)

1.51 Set maximum merge group size

'-M' - SET MAXIMUM MERGE GROUP SIZE

This option sets the maximum size of a merged file group when adding files to an archive. This sets an upper limit on the amount of file data that can be merged together.

The only reason a maximum merge group size exists is to prevent the situation where one may wish to delete or update one file in a very large merged group (e.g. 10 MB of merged files). Since all undeleted data must be compacted (recompressed) after a deletion, it is advantageous to make the merged group size not too large. On the other hand, if the merged group size is too small, then compression suffers.

The default merged group size is 260K. The minimum allowable is 8K, and the maximum allowable is just over 8000K (8 Megabytes).

The parameter to this option is specified in 1000's of bytes (not 1024); for example, -M500 would specify a maximum merged group size of 500,000 bytes.

VALIDITY

This option is valid with the following commands:

   a (add)
   d (delete)
   f (freshen)
   r (replace)
   u (update)

1.52 On or after date (yyyy/mm/dd)

'-o' - ON OR AFTER DATE

This option puts a date constraint on the files on which LZX will operate; LZX will ONLY operate on any files which have a modification date equal to or later than that specified as a parameter to this option, in the form

"yyyy/mm/dd".

For example, if extracting files and using the option "-o 1994/04/20", then only files which have a modification date on or after April 4, 1994 in the archive will be extracted.

Similarly, if adding files to an archive, only files with a date on or after the one supplied as a parameter wlil be added.

This option works in conjunction with other constraints, so the command "LZX -o 1994/01/01 x test.lzx *.c" would extract from the archive "test.lzx" all files which had the extension ".c" AND which also had a modification date of January 1, 1994 or later.

If updating, freshening, or replacing files already existing in an archive, then the "on or after" date refers to the date of the physical file, rather than the date of the file in archive.

As a sidenote, the format of the date parameter was chosen as "yyyy/mm/dd" because this format is unambigious; European countries all use different date formats, which are often different from those used in North America and the rest of the world.

VALIDITY

This option is valid with the following commands:

- a (add)
- d (delete)
- e (extract)
- f (freshen)
- l (list)
- r (replace)
- t (test)
- u (update)
- v (view)
- x (extract)

1.53 On or before date (yyyy/mm/dd)

'-O' - ON OR BEFORE DATE

This option puts a date constraint on the files on which LZX will operate; LZX will ONLY operate on any files which have a modification date equal to or before than that specified as a parameter to this option, in the form "yyyy/mm/dd".

Otherwise, this option behaves in an analagous fashion to the -o (On or after date) option.

VALIDITY

This option is valid with the following commands:

```
   a (add)
   d (delete)
   e (extract)
   f (freshen)
   l (list)
   r (replace)
   t (test)
   u (update)
   v (view)
   x (extract)
```

1.54 Pause after loading

```
'-p' - PAUSE AFTER LOADING

This option causes LZX to pause and wait for the user to press a key, after
it has loaded.  This feature is provided for users with floppy drive systems
only, who may wish to swap disks before allowing LZX to start operating.


VALIDITY

This option is valid with the following commands:

   a (add)
   d (delete)
   e (extract)
   f (freshen)
   l (list)
   r (replace)
   t (test)
   u (update)
   v (view)
   x (extract)
```

1.55 Set task priority

```
'-P' - SET TASK PRIORITY

This option sets the priority of the LZX process for the duration of its
running.  The priority may be any value from -127 to +127, although only
values between -5 and +5 should be used.

The higher the priority, the more CPU time will be used by LZX.  Because
of the way in which the Amiga's process priority system works, and because
LZX is very processor intensive, setting LZX to a high priority will probably
stop (or at least slow to a very slow crawl) any lower priority processes.

The one time LZX does not use much CPU is when it is waiting for disk i/o to
complete (from an actual physical disk; RAM drives do not count).
```

By default LZX inherits its task priority from the shell it was run from.

VALIDITY

This option is valid with the following commands:

```
a (add)
d (delete)
e (extract)
f (freshen)
l (list)
r (replace)
t (test)
u (update)
v (view)
x (extract)
```

1.56 Set quiet mode

'-q' - SET QUIET MODE

This option tells LZX to not output anything to standard output. Two modes are supported:

'-q 0' - Normal output (the default)
'-q 1' - Disable all output

Entering '-q' as an option is equivalent to entering '-q 1', which disables all LZX output to the console.

1.57 Faster compression

'-Qf' - FASTER COMPRESSION

[REGISTERED VERSION ONLY]

The '-Qf' option causes LZX to increase the speed of compression, without affecting the degree of compression. Of course, this option has a price; it also causes LZX to use 32K more RAM.

The '-Qf' option doesn't improve compression speed for all files; text files, for example, generally see little improvement in speed, while Amiga executables compress about 5% faster.

1.58 Recurse into subdirectories

'-r' - RECURSE INTO SUBDIRECTORIES

This option is used in conjunction with wildcarding, and causes LZX to search

all subdirectories (in addition to the current directory) for matching
file names (and directories, if applicable).

Note that the '-x' (preserve path names) option is automatically enabled when
the '-r' option is used. If you do not wish to preserve path names, enter
the '-x0' option after '-r' on the command line.

VALIDITY

This option is valid with the following commands:

- a (add)
- f (freshen)
- r (replace)
- u (update)

1.59 Collect archives recursively

'-R' - COLLECT ARCHIVES RECURSIVELY

This option causes LZX to recurse into subdirectories (in addition to the
current directory) to look for matching archives.

For example, to view all files ending in ".h" in all LZX archives in the
current directory and its subdirectories, one would type:
"LZX -R v #?.lzx #?.h".

To add the file "MYBBS.ADV" to all LZX archives in the current directory and
its subdirectories, one would type: "LZX -R a #?.lzx MYBBS.ADV".

Similarly, to delete the file "zzzendpad.foo" from all LZX archives in the
current directory and its subdirectories, one would type:
"LZX -R d #?.lzx zzzendpad.foo"

VALIDITY

This option is valid with the following commands:

- a (add)
- d (delete)
- e (extract)
- f (freshen)
- l (list)
- r (replace)
- t (test)
- u (update)
- v (view)
- x (extract)

1.60 Add only files with no arc (A) bit

'-s' - ADD ONLY FILES WITH NO ARC (A) BIT

This command causes LZX to add only files without the arc (A) protection bit.

This option is useful for performing incremental backups, and is most often used in combination with the -S (Set A bit on added files) option.

VALIDITY

This option is valid with the following commands:

a (add)

1.61 Set arc (A) bit on added files

'-S' - SET ARC (A) BIT ON ADDED FILES

This option causes LZX to set the A protection bit on all files which are added to an archive.

This option is used primarily in incremental hard drive backups, and is most commonly used with the -s (add only files with no arc (A) bit set) option.

VALIDITY

This option is valid with the following commands:

a (add)

1.62 Make file names upper case

'-u' - MAKE FILE NAMES UPPER CASE

This option causes all file names to be converted to upper case.

If adding files to archives, the names of all files added will be stored in upper case. If extracting from archives, all files will be extracted using upper case file names. If viewing or listing archives, all files will be shown as having upper case file names even if this is in fact not the case.

VALIDITY

This option is valid with the following commands:

a (add)
e (extract)
f (freshen)

```
   l (list)
   r (replace)
   t (test)
   u (update)
   v (view)
   x (extract)
```

1.63 Set console update rate

```
'-U' - SET CONSOLE UPDATE RATE

This option sets the update rate of the console when compressing data
(archiving files, or compacting a merged file group when deleting files);
LZX will update the console each time the supplied number of Kilobytes is
compressed.  Due to LZX's file merging, it currently uses a percentage
display to update the console, but the calculation of when to update the
console is still perform in bytes.

LZX checks the CPU on your system to set the default rate to something
sensible for your system, be it a 68000 or a 68060.

As mentioned previously, the parameter should be specified in Kilobytes.
For example "-U4" will cause LZX to update the display for each 4K of
data compressed.

This parameter only affects the update rate for encoding; the update rate for
decoding is determined solely by the output buffer size.


VALIDITY

This option is valid with the following commands:

   a (add)
   d (delete)
   f (freshen)
   r (replace)
   u (update)
```

1.64 Set work directory

```
'-w' - SET WORK DIRECTORY

This option specifies the work directory of LZX.  The work directory is used
to hold temporary files when adding, updating, freshening, replacing, or
deleting files from archives.

The default work directory is "T:", which is usually assigned logically to
"RAM:T".  For systems with a small amount of memory, there may not be enough
memory to compress data using "RAM:T" as the work directory.  This can also
be the case if compressing very large files.
```

The name of the work directory supplied must end in either a colon (:) or a slash (/); if it does not, LZX will complain.

VALIDITY

This option is valid with the following commands:

- a (add)
- d (delete)
- f (freshen)
- r (replace)
- u (update)

1.65 Preserve path names

'-x' - PRESERVE PATH NAMES

This option tells LZX to preserve the full path name (excluding the device name) of any file added to (if adding) or extracted from (if extracting) the archive. This is useful if one wishes to preserve a complete directory structure.

By default this option is not set. In order to preserve the directory structure, this option must be used both when creating the archive, and when extracting it.

This option is set automatically when the '-r' (recurse into subdirectories) option is used; to use '-r' without activating this option (preserve path names), enter '-x0' after the '-r' on the command line; i.e. "-r -x0".

VALIDITY

This option is valid with the following commands:

- a (add)
- e (extract)
- f (freshen)
- r (replace)
- u (update)
- x (extract)

1.66 Control '.lzx' suffixing

'-X' - CONTROL '.LZX' SUFFIXING

This option tells LZX to append '.lzx' to an archive name if it does not already end in '.lzx'.

This option is enabled by default; that is, the following command will add the file 'somefile' to 'myarc.lzx': "lzx a myarc somefile".

To disable this behavior, use the -X0 option. For example, the following command will add 'somefile' to 'myarc': "lzx -X0 a myarc somefile".

VALIDITY

This option is valid with the following commands:

- a (add)
- e (extract)
- f (freshen)
- l (list)
- r (replace)
- t (test)
- u (update)
- v (view)
- x (extract)

1.67 Store files with ratio > x%

'-y' - STORE FILES WITH RATIO > X%

This option causes LZX to store files which have a compression ratio of greater than the specified percentage. For example '-y95' would cause LZX to store all files which compressed to 95% or more of their original size.

Specifying a number greater than or equal to 100 makes no sense; LZX will always store any file which compressed to more than its original size.

This option is a carry-over from the other Amiga archivers which often decompress poorly compressed files very slowly. LZX is very fast at decompressing any type of file, but it must be said that decompressing a stored file is quicker (it's all disk i/o and very little CPU usage; for a stored file, the disk i/o will be the bottleneck).

Note that merged file groups are treated as one file for all intents and purposes; therefore, the whole merged group will be stored if the compressed group is greater than the group's combined original size.

VALIDITY

This option is valid with the following commands:

- a (add)
- d (delete)
- f (freshen)
- r (replace)
- u (update)

1.68 Compress archives

'-Y' - COMPRESS ARCHIVES

This option forces LZX to compress files which are known by their extension to already be compressed. By default, LZX will simply store these files without attempting to compress them, the reason being that attempting to compress one of these types of files yields very little (invariably 0-3%) in the way of additional compression.

The following file extensions are treated as indicating an archive:

AIN ARC ARJ DMS GIF HAP JPG LHA LHW LZH LZX PAK PP RAR UC2 WRP ZAP ZIP ZOO

Additional file extensions will be added as more archive types become available.

VALIDITY

This option is valid with the following commands:

- a (add)
- u (update)

1.69 Store all files

'-0' - STORE ALL FILES

This option causes LZX to store (i.e. not compress) all new files added to an archive. This option is useful if one wishes to make use of LZX only as an archiving tool, rather than as a compressive tool.

If updating an archive, existing files will be recompressed with whatever compression mode they were originally compressed.

VALIDITY

This option is valid with the following commands:

- a (add)
- u (update)

1.70 Fast compression

'-1' - FAST COMPRESSION

This option causes LZX to use its "fast compression" mode; this mode is completely output compatible with LZX's other compression modes, except that the compression parameters have been tweaked to favour speed over compression.

The loss in compression is not very significant; usually about 1-2%.

VALIDITY

This option is valid with the following commands:

   a (add)
   u (update)

1.71 Default compression

'-2' - DEFAULT COMPRESSION

This option causes LZX to use its "default compression" mode; this mode is completely output compatible with LZX's other compression modes. In this case, the compression parameters have been selected to provide a fairly optimal balance between compression speed and degree of compression.

VALIDITY

This option is valid with the following commands:

   a (add)
   u (update)

1.72 More compression

'-3' - MORE COMPRESSION

This option causes LZX to use its "more compression" mode; this mode is completely output compatible with LZX's other compression modes. In this case, the compression parameters have been tweaked to provide a greater degree of compression at the expense of speed.

This option provides the maximum amount of compression available on the evaluation version of LZX; however, the registered version's
-9
(Maximum compression) option provides still better compression.

VALIDITY

This option is valid with the following commands:

   a (add)
   u (update)

1.73 Maximum compression

```
                ’-9’ - MAXIMUM COMPRESSION

[68020/68030 and 68040/68060 REGISTERED VERSIONS ONLY]

This option causes LZX to use its "maximum compression" mode; this mode is
completely output compatible with LZX’s other compression modes.  In this
case, the compression parameters have been tweaked to provide the maximum
possible compression, at the complete and utter expense of speed.

The improvement in compression from the -3 to the -9 mode is invariably
1.5 to 3 times larger than the improvement from the -2 (the default) to
the -3 mode.

However, the -9 mode is between 2 and 8 times as slow as the default
compression rate, depending on the type of file being compressed.  The
decompression speed is unaffected, and is as fast as ever for files
compressed with the -9 mode.

The -9 version is provided for distribution-type applications, where the
data is compressed once (usually on a fast machine!) and is then
distributed online or on disk.  The -9 mode is slow enough that it is
impractical for casual day-to-day use, but is useful when the absolute
maximum degree of compression must be attained.

Typical applications for the -9 mode are recompressing BBS file
directories to save disk space and make files smaller for users to
download; to distribute online magazines; to distribute FidoNet
nodelists; or to cram as much software as possible onto a disk.

The
                 Benchmarks
                 section contains test data using the -9 option.


VALIDITY

This option is valid with the following commands:

   a (add)
   u (update)
```

1.74 Add

```
                ’a’ - ADD FILES TO ARCHIVE(S)

The add command adds one or more files to one or more archives, although
typically it is used to add files to only one archive.

If the archive specified does not already exist, it will be created.
Otherwise, files will be appended to the archive.  LZX will not add files to an
archive if they are aleady present in the archive; attempting to do so will
```

elicit a warning message from LZX, although it will continue to add any other files you may have specified.

By default, only the file name component of the file is stored in the archive; that is, any preceding path name is not stored. In order to preserve the path name as well, use the -x option. If you specify the -r option to recurse into subdirectories, then the -x option is set automatically.

To preserve empty directories (directories containing no files or subdirectories), select the -e option.

The -x, -r, and -e options can be used to preserve a directory structure.

Examples

1.75 Examples of the ADD command

EXAMPLES OF THE ADD COMMAND

'LZX a myarchive.lzx testfile.txt' would add the file 'testfile.txt' to the archive 'myarchive.lzx'. However, if 'testfile.txt' were already present in the archive, then it would not be added. If 'myarchive.lzx' didn't exist at the time the command was invoked, then 'myarchive.lzx' would be created as a new archive.

'LZX a myarchive newfile.txt' would perform exactly the same operation as the above, since LZX would automatically append the '.lzx' extension onto the archive name 'myarchive'. To alter this behavior, the -X option can be used.

'LZX a myarchive *.txt *.doc test.tmp' would add to the archive 'myarchive.lzx' all files ending in '.txt', '.doc', and the file 'test.tmp'.

'LZX a myarchive.lzx subdir/newfile.txt' would add the file 'subdir/newfile.txt' to the archive 'myarchive.lzx'. The file would be stored as 'newfile.txt' within the archive; that is, the 'subdir/' prefix would not be preserved. In order to have the file stored as 'subdir/newfile.txt' within the archive, the -x option would be required: 'LZX -x a myarchive.lzx subdir/newfile.txt'

1.76 FastAdd

'af' - FAST ADD FILES TO ARCHIVE(S)

[REGISTERED VERSION ONLY]

The fast add command is almost identical to the add command in that it adds one or more files to one or more archives.

The only difference is that the fast add command does not check the existing archive (if it does already exist) for duplicate files. Therefore,

if one is not careful, one can add the same file to the archive multiple
times.

For example: lzx af test.lzx testfile
             lzx af test.lzx testfile

The above case is allowed, and will result in two "testfile" files being
in the archive. The add ('a') command, on the other hand, will not allow
a file to be added twice.

The fast add command is useful if you know that there will be no duplicate
files in the archive. This command would typically be used in a FidoNet
mail setup, where new mail packets are added to existing mail archives and
the mail packets are all guaranteed to have unique names. In this
situation, the fast add command can speed up mail processing dramatically.

1.77 Concatenate

'c' - CONCATENATE ARCHIVE(S)

[REGISTERED VERSION ONLY]

The concatenate command either appends one or more archives to an existing
archive, or creates a new archive by appending one or more archives together.

For example: lzx c dest.lzx arc1.lzx arc2.lzx

If "dest.lzx" exists, the archives "arc1.lzx" and "arc2.lzx" will be appended
to it. "arc1.lzx" and "arc2.lzx" will be unchanged.

If "dest.lzx" does not exist, it will be created, and will consist of
"arc2.lzx" appended onto "arc1.lzx". "arc1.lzx" and "arc2.lzx" will be
unchanged.

This command is provided to append compressed data from one archive to
another without having to recompress the archive.

NOTE: LZX performs no checking for duplicate files; it simply concatenates
      the archives together blindly.

1.78 Delete

'd' - DELETE FILES FROM ARCHIVE(S)

The delete command deletes files from one or more archives, although typically
it is used to delete files from only one archive.

LZX requires temporary storage in the current directory in order to condense
the remaining files in the archive.

If any of the files to be deleted were part of a merged file group, then LZX
will recompress the undeleted files in the affected groups. The files will

be recompressed using the default compression mode, or the selected one if one is specified via one of the options '-1', '-2', '-3', or '-9' on the command line.

1.79 Extract

'e' - EXTRACT FILES FROM ARCHIVE(S) [NO PATH PRESERVATION]

This extract command is used to extract files from an archive. It differs from the 'x' extract command in that path names and directory structure will not be preserved by default.

For example, if one were to extract files from an archive, and a file named "mydir/subdir/myfile" were present in the archive, then the file 'myfile' would be created in the current directory.

Note that using the -x1 parameter with this command causes this command to behave in exactly the same was as the 'x' extract command.

1.80 Extract

'x' - EXTRACT FILES FROM ARCHIVE(S) [PATH PRESERVATION]

This extract command is used to extract files from an archive. It differs from the "e" extract command in that path names and directory structure are preserved by default.

For example, if one were to extract files from an archive, and a file named "mydir/subdir/myfile" were present in the archive, then the file "mydir/subdir/myfile" would be created. If the "mydir/subdir" directory did not already exist, then it would first be created.

Note that using the -x0 parameter with this command causes this command to behave in exactly the same was as the 'e' extract command.

1.81 Freshen

'f' - FRESHEN FILES IN ARCHIVE(S)

The freshen command freshens one or more files in one or more archives, although typically it is used to freshen files in only one archive. The freshening process replaces older files which already exist in the archive. When replacing files, the 'last modification dates' of the files in the archive are compared with those of their counterparts specified at the command line, and the earlier modification date is assumed to represent the older file.

As with the add command, only the file name component of the file is stored by default. The -x or -r options should be used to preserve path names.

The freshen command is almost identical to the update command, except that the freshen command does not add any new files to the archive; it will only freshen existing ones. The replace command is similar to the freshen command, except that it always replaces files in the archive, regardless of their modification dates.

1.82 Update

'u' - UPDATE FILES IN ARCHIVE(S)

The update command updates one or more files in one or more archives, although typically it is used to update files in only one archive. The updating process adds files which are not yet in the archive, and replaces older files which already exist in the archive. When replacing files, the 'last modification dates' of the files in the archive are compared with those of their counterparts specified at the command line, and the earlier modification date is assumed to represent the older file.

As with the add command, only the file name component of the file is stored by default. The -x or -r options should be used to preserve path names.

Two commands similar to update are the freshen and replace commands. The freshen command is almost identical to the update command, except that no new files will be added to the archive; only existing files in the archive may be updated, depending on their last modification date. The replace command is almost identical to the freshen command, except that all specified files are always updated, regardless of their last modification date.

1.83 Replace

'r' - REPLACE FILES IN ARCHIVE(S)

The replace command replaces one or more files in one or more archives, although typically it is used to replace files in only one archive. The replacing process replaces files which already exist in the archive with those specified at the command line.

As with the add command, only the file name component of the file is stored by default. The -x or -r options should be used to preserve path names.

The replace command is similar to the freshen command.

1.84 Test

't' - TEST FILES IN ARCHIVE(S)

The test command is used to test the integrity of files in archives. This is done by extracting the file data and throwing it away (i.e. no files are created).

If no file names are specified on the command line, then all files in the archive(s) are tested; otherwise, only the specified files are tested.

If any of the tested files are corrupted, or a problem is encountered in any archive(s), LZX will exit with an error code.

1.85 List

'l' - LIST FILES IN ARCHIVE(S) [TERSE]

The list command is used to display the contents of archives. The list command displays the following information for each file in an archive:

File name
Original file size
Packed file size
Compression ratio (packed/original as a percentage)
File modification date and time

In addition, similar statistics for the whole archive itself are tallied and displayed.

Sample output might be:

Viewing archive 'testarchive.lzx':

```
Original  Packed  Ratio   Date      Time   Name
-------- -------- ----- --------- -------- -----------
  156492    78826 50.3% 10-Mar-94 22:29:14 test
  419328   164352 39.1% 19-May-93 09:25:12 book1
-------- -------- ----- --------- --------
  575820   243178 42.2% 09-Nov-94 17:17:06 2 file(s)

1 archive(s) viewed
```

The information under the second dotted line is for the archive itself. For example, the archive contains a total of 575820 bytes worth of data, compressed down to 243178 bytes, for an average compression ratio of 42.2%, and the modification date of the archive is 09-Nov-94 17:17:06. Two files were present the archive.

Note that one can list only some of the files in the archive, if desired; these files can be specified on the command line, either explicitly, or as wildcards. For example, in the above case, if one were to enter 'lzx l testarchive.lzx book1', the following would be displayed:

Viewing archive 'ram:blah.lzx':

```
Original  Packed  Ratio   Date      Time   Name
-------- -------- ----- --------- -------- -----------
  419328   164352 39.1% 19-May-93 09:25:12 book1
-------- -------- ----- --------- --------
  419328   164352 39.1% 09-Nov-94 17:17:06 1 file(s)

1 archive(s) viewed
```

Note that in this case, the information tallied under the second dotted line is only for the files listed, and not for the complete archive; that is, 419328 bytes worth of uncompressed file data was listed, and this data was compresed to 164352 bytes, for a compression ratio of 39.1%, and 1 file was listed.

When solid file groups are present in the archive, the situation is slightly different, as multiple files have been compressed together as if they were one large file. In this situation, LZX estimates the compressed file size of each of the solid files by giving it a compressed size proportional to that of its actual size when compared to the solid group.

An asterisk will appear between the original file size and compressed file size of any file which is a member of a solid file group, to inform you that the packed file size is an estimated value.

1.86 Print

'p' - PRINT FILES IN ARCHIVE(S) TO SCREEN

[REGISTERED VERSION ONLY]

The print command is used to display text files stored in archives, to the screen. For example: "lzx p archive.lzx readme.txt". This features allows one to read "readme" and ".diz" files in archives without having to extract the files first.

To read any "readme" type files in a newly obtained archive, one can use a command similar to: "lzx p archive.lzx #?read#?".

If no file names are specified on the command line, then all files in the archive(s) are printed; otherwise, only the specified files are tested. A short banner appears before each file listed, providing its name and file length.

This command should be used to view text files only; viewing binary files will result in garbage being displayed to the screen. If this should happen, a ^C (control-C) will abort the operation.

If any of the tested files are corrupted, or a problem is encountered in any archive(s), LZX will exit with an error code.

1.87 View

'v' - LIST FILES IN ARCHIVE(S) [VERBOSE]

The view command is similar to the list command, in that it is used to display the contents of archives. The difference is that the view command displays more information than the list command.

There are in fact two view commands (not counting the 'list' command); the

ordinary view command ('v') and the verbose view command ('v1'). The 'v1' command displays even more information than the 'v' command, as is explained below.

The list command ('l') displays the following information:

File name
Original file size
Packed file size
Compression ratio (packed/original as a percentage)
File modification date and time

The view command ('v') displays the following additional information:

File attributes

File CRC

Compression mode
If the verbose view command ('v1') is used, then the following additional information is also displayed:

Host operating system
Note that the information in the verbose view ('v1') command will not fit on an 80 column display.

The sample output of the view command ('v') might be:

```
Viewing archive 'test.lzx':

Original  Packed  Ratio   Date      Time   FileAttr  CRC-32  M Name
--------- -------- ----- --------- -------- -------- -------- - -----------
  151496    76854 50.7% 15-Nov-94 02:17:15 --p-rwed 8CA65049 1 testfile
--------- -------- ----- --------- --------
  151496    76854 50.7% 15-Nov-94 02:17:21 ----rwed 1 file(s)

1 archive(s) viewed
```

The sample output of the verbose view command ('v1') might be:

```
Viewing archive 'test.lzx':

Original  Packed  Ratio   Date      Time   FileAttr  CRC-32  Host OS M Name
--------- -------- ----- --------- -------- -------- -------- ------- - -----------
  151496    76854 50.7% 15-Nov-94 02:17:15 --p-rwed 8CA65049 Amiga   1 testfile
--------- -------- ----- --------- --------
  151496    76854 50.7% 15-Nov-94 02:17:21 ----rwed 1 file(s)

1 archive(s) viewed
```

1.88 Host operating system

HOST OPERATING SYSTEM

In order to ensure easy transportation of files across platforms, LZX records a Host OS (Operating System) for each file added to the archive, so that file attributes can be translated compatibly across platforms.

LZX currently is aware of the following operating systems:

Amiga
MS-DOS
Windows
OS/2
UNIX

1.89 Compression Modes

LZX COMPRESSION MODES

LZX currently supports only one compression mode ("1"); files in LZX archives are either compressed using this compression mode, or stored as uncompressed data ("s").

The "-1" "-2" "-3" and "-9" compression options simply alter LZX's search depth parameters to trade speed for degree of compression, or vice versa; the output of the four compression modes is completely compatible.

1.90 File attributes

AMIGADOS FILE ATTRIBUTES

The following file attributes are supported by AmigaDOS and LZX:

h

s

p

a

r

w

e

d

1.91 Hidden

THE AMIGADOS 'HIDDEN' ('h') FILE ATTRIBUTE

The 'hidden' attribute is not used under AmigaDOS; setting it does not cause files to be hidden from directory listings. In fact, the AmigaDOS 3.1 'protect' command does not recognise the 'h' bit.

1.92 Script

THE AMIGADOS 'SCRIPT' ('s') FILE ATTRIBUTE

The 'script' attribute indicates that the file is an AmigaDOS shell script file.

1.93 Pure

THE AMIGADOS 'PURE' ('p') FILE ATTRIBUTE

The 'pure' attribute indicates that the file is residentable and re-entrant, and can be made resident with the AmigaDOS 'resident' command, or equivalent command if operating under a different shell.

1.94 Archived

THE AMIGADOS 'ARCHIVED' ('a') FILE ATTRIBUTE

The 'archived' attribute is used by most backup programs as a way of determining which files have been changed since the last backup; any write to a file with the 'a' attribute set will unset the 'a' attribute.

1.95 Readable

THE AMIGADOS 'READABLE' ('r') FILE ATTRIBUTE

The 'readable' attribute is used to indicate that a file can be read; a file without this attribute cannot be read. Almost all files have the 'readable' attribute set.

1.96 Writeable

THE AMIGADOS 'WRITEABLE' ('w') FILE ATTRIBUTE

The 'writeable' attribute is used to indicate that a file can be written to; a file without this attribute cannot be written to, although it can be deleted (the 'delete' attribute must be unset in order to prevent deletion).

1.97 Executable

THE AMIGADOS 'EXECUTABLE' ('e') FILE ATTRIBUTE

The 'executable' attribute is used to indicate that the file is a binary load file, which can be executed from the shell, for example.

1.98 Deleteable

THE AMIGADOS 'DELETE' ('d') FILE ATTRIBUTE

The 'deletable' attribute is used to indicate that a file can be deleted using the AmigaDOS 'delete' command (or any other command or operating system call which provides the same functionality). If the 'deletable' attribute is not set, then the file cannot be deleted.

1.99 Limitations

LIMITATIONS

LZX's limitations are as follows:

* Path names are limited to a maximum 255 characters in length.

* The size of a single archive must not exceed 2 Gigabytes.

* The maximum number of files that can be compressed in one invocation is limited only by available memory.

1.100 Suggestions

SUGGESTIONS

If compression speed is your number one priority and you have large amounts of memory to spare, then increasing the output buffer size (-bo) to a very large size (256K+) will dramatically down on disk usage.

On the other hand, if you have a low memory situation, set the output buffer size to a small value, such as 8K. You should also set the work directory to something other than the default, which is "T:", since T: is generally assigned to "RAM:".

1.101 Environment Variables

ENVIRONMENT VARIABLES

The evaluation version of LZX does not support environment variables, or a set of environment-specified default options.

1.102 Registration and The Future

REGISTRATION AND THE FUTURE

LZX was developed over the period of approximately one and a half years. A lot of thought went into every component of LZX, and countless brainstorming sessions kept us up until 5 AM for weeks on end, while our university marks nose-dived due to all of our missed classes.

A PC version of LZX is being developed in parallel with the Amiga version, although the PC version is several months behind the Amiga version, due to MS-DOS's numerous architectural problems. We dearly hope that LZX, an Amiga program, will oust the market leaders on the PC, but it will be a tough battle; PKZIP is heavily entrenched in the PC market.

The LZX project needs your support! LZX is fairly inexpensive as shareware programs go, especially when you consider what you get:

The current (1.21r) registered versions of LZX support the following:

- Faster compression
- New "maximum compression" (-9) option
- New "faster compression" (-Qf) option
- Decompression of LHA and LZH files (25-35% faster than LhA!)
- Asynchronous disk i/o
- More options (fast add, archive concatenation)

Multi-volume archiving is under development, and will probably be included in version 1.30r.

If you have a feature you would like added, feel free to let us know, and drop us a line either by

Email
,
Snail-mail
,
fax
, or even

telephone
.

PAYMENT DETAILS

The registration fee for LZX is US$ 25, which can be paid by either cheque or money order if in the U.S. or Canada, or by postal money order if elsewhere. The address to send the above to is:

Data Compression Technologies
383 Lawrence Avenue West
Toronto, Ontario
M5M 1B9
Canada

Alternatively, you may send cash; the cash may be either US$ 25, or its

equivalent at the current exchange rate to the currencies of the following countries:

   Canada
   U.K.
   Germany
   Sweden
   Denmark

This alternative payment scheme is available so that one does not have to make the trek down to the post office and pay an extra fee for the creating of a money order; now all one has to do is hide cash in the envelope and send it off.

For example, if you wished to send Canadian dollars, you would look up the current exchange rate (currently 1 US$ ~= CDN$ 1.39) and then send the equivalent amount (CDN$ 34.96, or CDN$ 35). Note: Only bills/notes are accepted. Foreign coins are not accepted, because they cannot be exchanged at a local bank.

A note about sending cash in the mail; make absolutely sure that it is disguised; if it is not, you can bet that along the way someone will open it and take the money out - it happens. Enclose the cash in two pieces of non-white paper to disguise it. We cannot take responsibility for cash lost in the mail; if you think this may happen, send a money order instead.

To check the status of your order, you can contact the authors as described above.

You will receive a disk with the registered versions of LZX, as well as your personalised keyfile, which will enable you to use any of the publicly distributed registered version archives.

1.103 Contacting the authors via Email

CONTACTING THE AUTHORS VIA EMAIL

Please Email us with your comments, questions, and suggestions!

Jonathan Forbes:
   jonathan.forbes@canrem.com (always)
   jadforbe@electrical.watstar.uwaterloo.ca (Jan 4 - Apr 18, 1996)

Tomi Poutanen:
   tjpoutan@elecom2.watstar.uwaterloo.ca (Jan 4 - Apr 18, 1996)

1.104 Contacting the authors via Fax

CONTACTING THE AUTHORS VIA FAX

The authors can be faxed at the number below, which is located at the address
of Jonathan Forbes. The fax machine resides on a dedicated line, and is
available 24 hours a day. If appropriate, please indicate who the fax is for.

FAX: (416)-781-1502 (Toronto, Ontario, Canada)

1.105 Contacting the authors via Telephone

CONTACTING THE AUTHORS VIA TELEPHONE

Jonathan Forbes can be reached at the number below:

Voice: (416)-781-1501 (Toronto, Ontario, Canada: EST)

1.106 Contacting the authors via Snail Mail

CONTACTING THE AUTHORS VIA SNAIL MAIL

Please mail us your comments, questions, and suggestions!

The authors can be contacted via snail mail at the following address:

Data Compression Technologies
383 Lawrence Avenue West
Toronto, Ontario
M5M 1B9
Canada

1.107 Benchmarks

BENCHMARKS

The benchmarks were performed on the following setup:

Amiga 3000/25
35 MHz 68040 Accelerator
8 Megabytes of 60 ns FAST RAM
4 Megabytes of 80 ns FAST RAM
2 Megabytes of CHIP RAM

The following programs were benchmarked; LZX, LhA, Shrink, and InfoZip.
Details on the programs are as follows:

Program	Version	Comments
LZX	*1.20	Best compression of all, very fast
LhA	*1.38e	Poor compression, but very fast
Shrink	1.1	Excellent compression, but very very slow
InfoZip	2.0.1	Good compression, slow, but portable

* The evaluation version of LZX 1.20 for the 68040/68060 was used
* The evaluation version of LhA 1.38 was used. Since only a 68000 version of LhA is provided for evaluation, that was used.

All tests were performed in the RAM: drive, and all programs and files were loaded into RAM: beforehand.

Two sets of tests were run; a single file test, and a multiple file test. As one might guess, the single file test involves compressing only a single file; while this nullifies the effect of LZX's file merging capability, the results clearly demonstrate that LZX, even on single files, is far superior to all other compression programs on the Amiga.

The multiple file test involves compressing many files at once. Here, LZX's file merging capability allows it to compress data far more effectively than the other programs.

One interesting result of note is borne out; not only does LZX compress better than all other Amiga archivers, but it also decompresses faster, and, in many cases, compresses faster!

Single File Test

Multiple File Test

1.108 Single File Test

SINGLE FILE TEST

Two of the files tested ("book1" and "obj2") were taken from the Calgary Corpus, a popular benchmark for compression programs. These are a text file and a binary object file, respectively.

Also tested was "xenomsgs" (a capture file from a BBS, consisting of highly compressible information).

All files are sorted in order of best compression to worst compression.

[================================ "book1" ================================]

(From the Calgary Corpus - text taken from a book)

Original size: 768771 bytes

Program	Compressed size	Compression time (s)	Decompression time (s)
*LZX-2	265870	N/A	N/A
LZX 1.20r -9	287230	33.55	0.75
LZX 1.20 -3	297568	12.11	0.77
LZX 1.20 -2	298232	11.64	0.77
LZX 1.20 -1	306744	8.75	0.77

Shrink 1.1	307320	33.61	18.63
Zip -9	312588	18.35	2.52
LhA 1.38e -2	338934	10.42	1.15

*available late 1995

[================================ "obj2" ================================]

(From the Calgary Corpus - object file)

Original size: 246814 bytes

Program	Compressed size	Compression time (s)	Decompression time (s)
*LZX-2	70290	N/A	N/A
LZX 1.20r -9	73431	14.24	0.26
LZX 1.20 -3	75223	2.91	0.26
LZX 1.20 -2	75959	2.44	0.26
LZX 1.20 -1	78549	2.20	0.27
Shrink 1.1	79462	6.00	5.47
Zip -9	81170	7.04	0.72
LhA 1.38e -2	84943	3.55	0.32

[============================== "xenomsgs" ==============================]

(Capture file from a BBS)

Original size: 876135 bytes

Program	Compressed size	Compression time (s)	Decompression time (s)
*LZX-2	178434	N/A	N/A
LZX 1.20r -9	206909	79.07	0.67
LZX 1.20 -3	212423	9.97	0.67
LZX 1.20 -2	213681	8.99	0.67
LZX 1.20 -1	217239	7.45	0.67
Shrink 1.1	224354	19.13	15.03
Zip -9	241759	12.07	2.20
LhA 1.38e -2	301480	8.65	1.09

1.109 Multiple File Test

MULTIPLE FILE TEST

The files in "lha_e138.run" (the distribution archive of LhA 1.38e) were extracted and recompressed with various archivers.

Also tested were compressing the complete INCLUDE directory of SAS/C 6.51 (C and ASM header files), and compressing the C directory of SAS/C

6.51 (containing all the compiler's command-line tools).

The appropriate commands were used for each archiver in order for it to preserve path names and recurse into subdirectories. For LZX, the "-M5000" option was also used to allow large merged file groups.

These results demonstrate that not only is LZX by far the most compressive utility for the Amiga, but that it is, astoundingly, also one of the fastest!

[========== "LhA 1.38 Distribution Archive" ==========]

Original size: 240262 bytes

Program	Compressed size	Compression time (s)	Decompression time (s)
LZX 1.20r -9	87524	13.52	0.28
LZX 1.20 -3	89024	2.99	0.28
LZX 1.20 -2	89330	2.70	0.28
LZX 1.20 -1	90720	2.32	0.28
Shrink 1.1	97294	8.29	6.80
Zip -9	99905	6.53	0.84
LhA 1.38e -2	102399	3.64	0.38

[========== "SAS/C 6.51 Include Directory" ==========]

Original size: 993828 bytes (361 *.h files)

Program	Compressed size	Compression time (s)	Decompression time (s)
LZX 1.20r -9	258611	81.82	1.20
LZX 1.20 -3	265233	22.74	1.20
LZX 1.20 -2	267633	20.90	1.20
LZX 1.20 -1	271915	19.46	1.20
LhA 1.38e -2	365269	23.35	2.18
Shrink 1.1	368966	40.26	29.16

NOTE: Zip's "-r" option was unable to recurse into all directories and collect only "*.h" files, so Zip has been excluded from this test.

[========== "SAS/C 6.51 C Directory" ==========]

Original size: 1580029 bytes

Program	Compressed size	Compression time (s)	Decompression time (s)
LZX 1.20r -9	717950	72.32	1.95
LZX 1.20 -3	733342	20.33	1.95
LZX 1.20 -2	738318	18.08	1.95

LZX 1.20 -1	751036	16.54	1.97
Shrink 1.1	865286	55.06	61.58
Zip -9	900980	38.20	6.63
LhA 1.38e -2	909071	25.57	2.69

1.110 Development Plan

DEVELOPMENT PLAN

Over the past 4 months we have developed a compression algorithm far superior to that currently in LZX. This new algorithm is appropriately called "LZX-2", and is currently undergoing optimisation and fine tuning. The Amiga release date of LZX-2 is 2 to 3 months away. However, preliminary compression figures appear in the single file benchmarks section.

We elected to not release a PC version using the currently available LZX algorithm, but to instead wait for LZX-2 before doing so, to ensure that when LZX debuts on the PC, it is far ahead of its competitors. The PC version will therefore be released close to the end of the year.

1.111 Compatibility

COMPATIBILITY

Almost without exception, the output of any new compression algorithm will be incompatible with that of existing algorithms. In some cases it is possible to retain output compatibility by simply using a faster search algorithm or more intelligent parser while retaining the older output format, but the savings from doing so are generally very slight (less than a percent). In other words, it is not possible to improve the LZH (-lh1-) and LHA (-lh5-) algorithms to anywhere near the degree of compression provided by LZX - in this respect, compatibility cannot be preserved.

To truly advance the state of the art, a new, incompatible compression format must be used. LZX advances the state of the art by providing a combination of degree of compression and speed not seen on either the Amiga or MS-DOS platforms.

Rather than let LZX languish as an Amiga-only niche product, it is our intent to push LZX to be the new cross-platform compression standard by providing both a PC version and a UNIX version. (It should be possible for a third party to port the UNIX version to the Macintosh platform, since neither Tomi nor Jonathan has Macintosh programming experience).

1.112 Decompression of LHA and LZH files

Decompression of LHA and LZH files

[REGISTERED VERSION ONLY]

As of version 1.10r (the registered version of 1.10), LZX can decompress ".lha" and ".lzh" archives. In fact, due to its faster decompression algorithm and asynchronous disk i/o, LZX decompresses ".lha" and ".lzh" files 25-35% faster than LhA.

As always, the speed improvement is dependent on the CPU and hard drive speed; your mileage will vary.