lha

COLLABORATORS			
	TITLE : lha		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 23, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

lha

1.1 index

1.2 LhA User's Guide

October 1998

```
     ###      ###      ######
      ##       ##      ###  ##
      ##       ##      ##  ###
      ##      #####    #######
      ##      ## ###   ##   ##
     ### ##  ##  ##  ##   ##
    ####### ###  ### ###  ###
```

Written by Stefan Boberg
Currently Maintained by Jim Cooper

Copyright © 1991-94 by Stefan Boberg
Copyright © 1998 by Jim Cooper
All rights reserved

1.3 Introduction

LhA is a powerful archiver for the Commodore-Amiga computer. It is
fully compatible with LHA for MSDOS systems and LhArc for MSDOS, Amiga
and *NIX. It is also compatible with LhArcA, LZ and LhArc for the
Amiga. LhA sports fast compression and decompression and has several
commands and options not found in any other currently available archiver
for the Amiga.

1.4 About the manual

The manual is divided into two main sections, the first section (this
one) contains miscellaneous information related to the program. The
second section is a reference section, where all commands and options
are explained thoroughly. For information about registering read
'LICENSE.man' and 'Orderform'.

1.5 System requirements

LhA will run on any Amiga system with at least 512KB RAM and one
diskdrive, though 1MB RAM and two diskdrives or harddrive is recommended
to get the most out of LhA. LhA will run on any KickStart from version

1.2 and up. If LhA is used on a system with the new KickStart 2.x it
will adapt to this and utilize features new to this release. Special
care has been taken in the program design to make the program execute as
fast as possible on 68020, 030 and 040-processors without sacrificing
68000 compatibility.

1.6 Terminology

ARCHIVE - An archive is a file containing one or more files in a
compressed or non-compressed state and related information like file
names, last modification date/time, filenotes etc.

COMPRESSION - The process of encoding redundant information into data
requiring less storage space. There are a multitude of ways you can do
this in. LhA uses a modified version of Lempel-Ziv compression with
block-adaptive Huffman coding and a dictionary size of 4096 or 8192
characters.

COMPRESSION RATIO - The compression ratio figures reported by LhA are
calculated as follows: ratio = (1 - (CompressedSize) / (OriginalSize))
* 100. I.e how many percent of the file that was GAINED. Other
archivers may use other methods. LHA and ARJ for MS-DOS, for example,
calculates the ratio as: ratio = (CompressedSize) / (OriginalSize),
i.e. how large the compressed file is compared to the original file.
(MSDOSratio = 1-(ADOSratio/100)). The higher the LhA compression ratio
is, the better the compression. Most Amiga archivers use the same ratio
calculation method as I use.

CRC - CRC stands for 'Cyclic Redundancy Check'. It is a relatively
sophisticated method of checking data integrity. The version used in
LhA is a 16-bit CRC.

EXTRACTION or DECOMPRESSION - The process of recreating the exact
information that was previously compressed (file contents, modification
date, filenotes, protection flags, directory structure etc.)

SELF-EXTRACTION MODULE (SFX-Module) - This is an archive that is an
executable file capable of extracting self-contained files.

1.7 LhA - what is it?

   LhA is a file archiving program written especially for the Commodore
Amiga computer. The primary goal is to provide the Amiga community with
a fast, efficient and reliable file archiver. LhA creates and processes
archive files with the '.LZH' suffix, and is fully compatible with both
MSDOS LhArc and MSDOS LhA, as well as the Amiga LhArc, LZ, and LhArcA.
It handles both the old LhArc-style compression (-lh1-, -lh0-) and the
new LHA-style (-lh5-, -lh4-).

   LhA is at the time of release by far the fastest LhArc archive
handler, and is more reliable and robust than both LZ and LhArc.
Neither LZ nor LhArc even comes close to the speed of LhA. The

compression and decompression routines are written entirely in very
optimized 68000 assembly language. The routines were written directly
from scratch in pure 68000 assembler, and did not, as in the other Amiga
LZH/LHA-archivers, start as compiled C-code. Because of this, the
routines are both smaller and, more importantly, faster than they would
have been if I'd used compiler output as source material. You can get
an idea of how fast LhA is compared to other archivers by looking at the
speed test results in 'benchm.txt'. An even better way to see how fast
LhA is, is to try it yourself!

1.8   What is a file archiver anyway?

A file archiver, as the name implies, archives files. It collects
the files you specify and stores them all in a single archive-file.
Almost all file archivers (including LhA) also compress the files before
putting them in the archive-file, so that they occupy less diskspace.
When you wish to retrieve some file from the archive, the archiver
decompresses the file and restores it's file attributes (Last
modification date, time, file comments, protection status etc.). A file
archiver can usually also process archive files in different ways, for
instance delete files, freshen files, print files etc. See the
'ARCHIVER COMMANDS' section of this manual for an explanation of the
different actions LhA is capable of performing.

The most common use for a file archiver is for transferring several
related files via modem. It would be a very tedious and cumbersome task
to transfer for instance every single source file of a big project
separately, so why not put them all into one single file? This is where
the file archiver comes in, we simply feed the archiver with all the
files we wish to transmit, and then just transfer the single
archive-file the archiver then creates! After the transfer the receiver
just has to use the archiver to extract all files from the archive-file
onto his/hers harddisk (or floppy). Also, if the files were compressed
by the archiver, it would take less time to transfer the files as well,
which means the phonecall will cost us less. There are of course other
uses for a file archiver, you can use it as a harddisk-backup program
for example (if you have another harddisk partition to put the archive
file on..), and you can use it to stuff away files you don't use very
often, and then when you want to use them you simply extract the files
from the archive, and then delete them when you're finished (saves disk
space). Personally I use LhA a lot to make automatic backups of my
source codes for various projects.

The file compression methods vary from very simple, less effective,
and fast (Run-Length Encoding, RLE, for instance), to complex, effective
and relatively time-consuming methods (Lempel-Ziv-Huffman, LZHUF, as
used in LhA). The method used in LZH-Archivers (LZHUF) is to date
probably the best algorithm used in an archiver. There are other
similar methods, like ZIP, but they are not as good. Even though the
files become smaller you don't lose any information when compressing
them, the information is just stored in a different way. Basically,
redundant (repeated) information is replaced with a pointer to some
other part of the file, where this information is located. For example
in this text the word 'archive' appears at several places, this is an
example of redundant data. Simply put, if a file compressor was to

compress this file it would let the first occurence of 'archive' remain
unchanged, and then it would replace all other occurences of 'archive'
with a pointer to the first one. When decompressing the file, the
archiver uses these pointers to restore the file to its original state.

    Files which have already been compressed with one technique can
generally not be compressed any further by feeding them to the same file
compressing program again (If that had been possible, modem transfers
would have been a lot cheaper :), since the redundant information has
already been eliminated. It is possible though to compress files output
by certain compressors (RunLength–Encoders for example) further by
feeding them to a program that uses another method (like LZHUF), since
they eliminate different kinds of redundant information.

1.9 Compatibility and Amiga-specific features

    LhA is aimed at full compatibility with LHA V2.13 for MS–DOS, which
is an improvement of the original LhArc V1.13. LhA is also compatible
with LhArc, LhArcA and LZ for the Commodore Amiga computer. However,
LhArc and LhArcA cannot process any archives with headers of level 1 or
2, or files compressed with the new LHA compression (–lh5–). LZ 1.92
cannot process archives with headers of level 2. LHA V2.13 can process
all archives created by LhA.

1.10 About the author, program history and future

                        (Stefan's original data)

    I, Stefan Boberg – the author of the programs in the LhA family, am
19 years old and currently studying 'applied physics and electrical
engineering', first year, at the Linköping Institute of Technology. I
started working on LhA mainly because I thought there was no real good
archiver for the Amiga, the ones that existed at the time (June 1991) I
began work on it were either too slow, had loose compression ratios or
were bugged/crippled so that they could not do what I needed an archiver
to do. I use archivers mainly to back up sources for my various
programming projects automatically, and I also use it a lot to just
decompress archives from bulletin board systems and computer networks.
Another reason for doing it was to earn a little extra money, which I
badly need, being a poor student with a _small_ allowance.. :)

                        (Jim Cooper's blurbage)

    I, Jim Cooper, started asking Stefan about the LhA source code since
he stopped working on it long ago. He finally sent it to me, probably
just to stop my nagging! :–)

    In any case, I have decided to release it to the Amiga community for
free. No more registrations necessary. This is the _complete_ version
of LhA, equivalent to the former "registered" version. I may or may not
release the source, as well, at a later date. Not decided, yet.

If you find bugs, let me know, and I'll try to fix 'em. Maybe not
immediately, since I do have a "day job," but I'll try to keep it
working as best I can.

Enjoy.

1.11   Reference guide

This section of the manual is intended to be used mostly as a
reference guide when you want to know exactly how a certain option or
command works. If you haven't used LhA before (but used other
archivers), you should at least glance through the descriptions of all
the commands and options to get an idea of what LhA can do.

1.12   Command line syntax

The command line syntax is as follows:

LhA [-options] <Command> <Archive> [[HomeDir] FileSpec] [@file]
[destdir]

The items in square brackets are optional, and the items in angle
brackets are mandatory. Read the following sections for exact
information on the various items.

1.13   Specifying options

Unlike other archivers, LhA lets you specify options anywhere on the
command line. The option specifier is '-' (dash), any items on the
command line that begins with this character are considered to be option
switches. If you want to specify a filename or something else that
begins with a '-' character, enclose the name in double quotes or use
double dashes. For example, to specify a filename of '-minus', you
could write either '"-minus"' or '--minus'.

If you write '-o' the option 'o' is enabled regardless of its initial
state.   If you want to disable an option, append a '0' (zero) after the
option, like in '-o0'. If an option is followed by any other numeric
character than '0', the option is enabled.

You can specify multiple options without having a dash in front of
every option character. An example would be '-ox0m', which would enable
option 'o', disable option 'x' and enable option 'm'. The only
exception is options taking multi-digit numeric arguments, which must be
followed by whitespace and another dash if you want to specify more
options (like in '-b32 -ox0m').

1.14 Specifying commands

    The first non-option argument on the command line MUST be the command
specifier.   The commands are case-insensitive ('l' means the same thing
as 'L'), and only the first character of the argument is considered
(except for the 'vv' command), so you may use verbose commands such as
'list' or 'add' instead of 'l' and 'a', respectively.

1.15 Specifying archives

    The archive specification must be the second non-option argument (the
first being the command specification). In most cases you can specify a
pattern here.   The exception being the 'm' (move files to archive)
command.

1.16 Specifying action files

    The action files are specified after the archive specification. The
action file specifications may include pattern matching tokens. Note
that, as all other file specifications in LhA, action file
specifications may contain wildcards for directory names as well –
'hd:*/*/dir/*.h' is valid, for example.

                          NOTE

    If you do not specify any action files, LhA assumes
    that you wish to act upon all files in the archive or
    in the current directory.

1.17 Home directories

    Home directories is a new concept introduced with LhA, it provides an
easy way of specifying what parts of pathnames that should be preserved
in the archive.   It can also be used to simplify specifications of
mutiple files in the same directory. It is perhaps best explained with
a couple of examples:

 EXAMPLE

  Example 1:

  lha   -x   a newarc dh0:files/ file1 dir1/file2 dir2/file3
  dh0:files2/ *.c

  This would add the following files to 'newarc.lha':

   Added file(s)   Stored as
   ----------------   -------------
   dh0:files/file1   file1

```
dh0:files/dir1/file2   dir1/file2
dh0:files/dir2/file3   dir2/file3
dh0:files2/*.c         *.c
```

Example 2:

lha -r a newarc hd:tmp/ *.c *.h hd:px/ *.s *.snd *.iff

This would add all '.c' and '.h' files in 'hd:tmp' and it's
subdirectories, storing pathnames, but excluding the 'hd:tmp' part.
For instance, the file 'hd:tmp/src/foo/arargh.c' would be stored in
the archive with the name 'src/foo/arargh.c'. Additionally, all '.s',
'.snd' and '.iff' files in 'hd:px' and its subdirectories will be
added, excluding the 'hd:px/' part of the name.

   Homedir specifications must end in '/' or ':', otherwise they won't
be recognized as such.

   Homedir specifications may contain wildcards and other pattern
matching tokens.

                              NOTE

     You are not supposed to include the home directory name
     in the action file specifications after the home
     directory spec. I.e. you should not enter
     'devs:printers/   devs:printers/*HP*',   but
     'devs:printers/ *HP*' is correct.

     The home directory remains active for the rest of the
     command line or until the next home directory
     specification. If you want to set the home directory to
     the current directory (as it is from the beginning),
     use a single slash ('/') as a home directory
     specification. This means you cannot use a single slash
     to specify the parent directory, to do this you will
     have to add an additional slash ('//' means parent
     directory, '///' the parent's parent directory and so
     on).

1.18   Recursive file collection

   When collecting files recursively (by using the -r option with a or u
commands), action file specs are treated somewhat differently. Home
directories work the same way as usual. In recursive file collection
mode, the last node of the action file specification (i.e. the file
name part) is used as a pattern that is compared to all files in the
specified directory and its subdirectories. Some examples to hopefully
clarify the somewhat fuzzy description:

EXAMPLE

Example 1:

lha -r a myarc *

This will add all files in the current directory and its
subdirectories to 'myarc.lha'.

Example 2:

lha -r a myarc *.c *.cpp

Will add all '.c' and '.cpp' files in the current directory and its
subdirectories to 'myarc.lha'.

Example 3:

lha -r a myarc ram:work/* hd:tmp/*.c

Will add all files in 'ram:work' and its subdirectories - as well as
all '.c' files in 'hd:tmp' and its subdirectories - to 'myarc.lha'.
The full pathnames will be stored (excluding the device specification
of course).

Example 4:

lha -r a myarc ram:work/ * hd:tmp/ *.c

Will do exactly the same as example 3, but LhA will not store the
'ram:work/' and 'hd:tmp/' parts of the filenames in the archive.
(Because of the home directory specifications).

Example 5:

lha -r a myarc ram:dir1 ram:makefile

Will archive all files in the directory 'dir1' and its subdirectories,
as well as the file 'ram:makefile'.

Example 6:

lha -r a myarc ram:dir1 ram:(makefile)

Will do almost the same as example 5, but will archive ALL 'makefile's
in ram: and all it's subdirectories (because of the parentheses - see
note below).

<div align="center">NOTE</div>

Explicitly specified directories (explicitly = without
pattern matching) will be treated as 'dirname/*', i.e.
all files in the directory and it's subdirs will be
archived. Explicitly specified files will only be
looked for in the current home directory, unless the
filename is enclosed in parentheses, in which case the
file will be looked for recursively. I have chosen to
implement it this way because LhA can then be used
better together with directory utilities such as
Browser or DirectoryOpus.

1.19   Specifying destination directory

You can optionally specify a destination directory for the files
written by the extract commands by writing the desired directory name
anywhere after the archive name on the command line. If no destination
directory is specified, LhA will use the current directory as the
destination. The destination directory specification must end in ':' or
'/', just like home directory specifications, or LhA would not be able
to distinguish directory names from action file specs.

 EXAMPLE

  'lha x corpus ram:' would extract the contents of 'corpus.lzh' to
  ram:.

  'lha   x   project   *.c   dl:tmp/' would extract the contents of
  'project.lzh' to the 'dl:tmp' directory.

  and so would 'lha x project dl:tmp/ *.c'.

                              NOTE

     You can specify a directory that does not already exist
     as the destination, LhA will automatically create the
     directory for you (without asking first).

1.20   `@'-files

   '@'-files are files that are treated as if their contents were
written on the command line. They can be used to specify files, options
commands and anything else can be specified on the command line. An
example would be the command 'lha -r e arc.lzh *.[chas] @filelist ram:',
which would extract all files matching '*.[chas]' or the files listed in
'filelist' to ram:.   Carriage returns and linefeeds in '@'-files are
treated as whitespace.

1.21   LhA limitations

   LhA has been written to be as flexible as possible, but there are
some limitations that you should be aware of as a user.

o LhA pathnames are currently limited to 255 characters. If you exceed
   this limit behaviour is undefined. User reports indicate that
   AmigaDOS does not handle pathnames with more than 180-190 characters
   properly.

o When headers of level 0 are used, filenotes may not be longer than
   230-{filename length (including path)} characters. With header level
   1 or 2 filenotes may be up to 255 characters (AmigaDOS currently only
   supports filenotes of max 80 characters so this should not be any
   problem except with exceptionally long filenames and paths).

o   The number of files in an archive files are only limited by available
    disk space.   The size of an archive must not exceed 2.147.483.648
    bytes (2 Gigabytes); LhA will get VERY confused.

o   The number of arguments on the command line is only limited by
    available RAM and the used shell.

o   The allowed number of wildcard-matched files is only limited by
    available RAM.   Any number of files may be extracted or added to an
    archive in one go.

o   Level 2 headers must not be longer than 1024 characters, or LhA will
    not be able to process them.

o   Currently LhA only handles multivolume archives with a maximum of 100
    volumes. If you create archives with more than this number of
    volumes, behaviour is undefined.

1.22   Environment variables

    LhA supports both local and global environment variables. Upon
startup LhA looks for the environment variable 'LHAOPTS' and includes
this as if it had been typed on the command line directly after the
'LhA' command name.   If you don't want to use the settings from the
environment variable, use the '-I' switch.

 EXAMPLE

  If you set LHAOPTS to '-N -b64' with the following command:

   1> setenv LHAOPTS -N -b64

  LhA would not display any file-progress indicator and use a
  64K I/O buffer for all following invocations until the machine
  is reset or LHAOPTS is changed. If you want to set some
  default options that should survive reset and power off, use
  the environment variable name 'ENVARC:LHAOPTS' instead, like
  in:

   1> setenv ENVARC:LHAOPTS -b64

  This would cause the environment variable LHAOPTS to be set to
  '-b64' whenever the machine is rebooted.

1.23   Pattern matching

    This section describes how LhA handles pattern matching and file
collection. For a discussion on what commands will accept file
patterns, please refer to section 2.1 (Command line syntax).

    Pattern matching in LhA is always case-insensitive.   (i.e. it
doesn't matter if you write names in upper- or lowercase, 'a' will match

both 'a' and 'A'.

1.24   Exactly what is pattern matching anyway?

   Pattern matching is a means of specifying several files in an elegant
and   relatively   straightforward   manner.   Instead of just lining up all
the   file names you would like to work on on the command line (which can
be   very   tedious   when   a   lot   of   files   are   involved) you can use a
technique   called   'pattern matching'.   With this technique you - as the
name implies - use the fact that the names of the files you wish to work
on often share certain characteristics.   For example, the names of files
containing   C-source   almost always end in '.c', so if you would like to
add all C-source files in the current directory you could take advantage
of   this   fact   by   specifying a pattern to that matches these files (in
this   case   such   a   pattern would be '*.c').   Exactly how these patterns
are   built   up   are   explained   in section 2.3.1 forward.   Also read the
sections   explaining   'how   to specify action files' and 'how to specify
archive files'.

1.25   Accepted pattern tokens

   LhA accepts   all valid KickStart 2.x+ pattern tokens.

   In the explanations that follow, the term 'expression' means either a
single   token or character (such as 'x' or '?'), or an alternation (such
as '(ab|cd|ef)'), or a character class (such as '[a-z,A-Z]').

1.26   Question mark (?)

   The   question   mark matches any one _single_ character.   The question
mark is sometimes also referred to as the 'wildchar'.

 EXAMPLE

  'd?' : matches   all   two-letter   names   beginning   with   a 'd'
        character.   For example 'dm' or 'd8'.

  'ab?d' : matches all four-letter names beginning with 'ab' and
        ending   in   'd'.   For example 'abcd', 'abad' and 'ab_d'
        but not 'abd' or 'acid'.

  'f??' : matches   all three-letter names beginning in 'f'.   For
        example 'foo', 'fel', 'fan'   but   not 'ab', 'fuga' or
        'fini'

1.27   Star/Asterisk (*)

The star matches any sequence of any length, including sequences with length zero (i.e. the null string). The '*' character is often called the 'wildcard' character.

EXAMPLE

'a*' : matches all names starting with an 'a', for example 'abba', 'anette'.

'a*z' : matches 'auugaz', 'awacz' and 'az' and any other names starting with an 'a' and ending in 'z'.

's*f*n' : matches 'stefan', 'staffan', 'steffen', 'sfn' or any other name starting with an 's', followed by any number (including zero) of arbitrary characters, followed by an 'f', and ending in 'n'.

'*.lzh' : matches all names ending in '.lzh'

1.28   Hash mark (#)

The hash mark matches a subsequent expression (pattern) 0 or more times. The simplest example of this is '#?' which will match any filename (equivalent to the '*' token).

EXAMPLE

'#a' : matches any name consisting of the 'a' character only. For example 'aaaa' and 'a'.

'b#ad' : matches any name beginning in 'b', followed by any number (including 0) of 'a' characters, and ending in 'd'. For example 'bad', 'bd' and 'baaaad'.

'#(ha)#(hi)urgh' : of 'ha':s followed by any number of 'hi':s followed by 'urgh'. For example 'hahahahahihiurgh' matches, and so does 'haurgh' and 'hahiurgh'.

1.29   Square brackets ([])

The square brackets enclose a set of characters to match. They are a bit like the parentheses but match single-characters only. You can either specify just the letters you would like the expression to match, as in '[abcx]' (this would match 'a', 'b', 'c' and 'x'), or you can specify ranges, like '[a-c,x-z]' (which would match 'a', 'b', 'c' and 'x', 'y', 'z').

EXAMPLE

'prg.[1-9]' : matches any five-letter name beginning with 'prg.', followed by a non-zero digit. For example

'prg.1', 'Prg.8'.

'Ver_[1-2].[0-9].[a-z]' :   matches   any   nine-letter   name
        beginning with 'ver_' followed by either a '1' or a '2'
        character, followed by a dot ('.'), a digit and finally
        a character between 'a' and 'z' (i.e. all letters in
        the   english   alphabet).   For example 'Ver_1.2.a',
        'Ver_2.9.d'.

'#[a-z 0-9]'   matches   any   name   containing   any   number   of
        alphanumeric characters (i.e.   either in the alphabet
        or numeric).   For example 'ados' or 'PDP11'. It does
        not   match   'AXE.dat'   however,   since it contains a '.'
        which is not in the specified character range.

'*.[chas]' : matches any name ending in '.c', '.h', '.a' or
        '.s'.

1.30   Parentheses and the vertical bar

Parentheses can be used to achieve several things. The first way of
using them is just like in mathematics – to group several individual
expressions into one single expression. The other way is to provide a
list of acceptable expressions separated with '|' chars. The entire
parenthesized expression is treated as one token by other tokens (like
'#' and '~'). These two are actually the same, since the first is just
a special case of the second use. This is easier to explain with a
couple of examples:

EXAMPLE

'(abc|def|xyz)' : will match names 'abc', 'def' and 'xyz' and
        no other.

'*.(doc|prf|man)' : will match all names ending in '.doc',
        '.prf' or '.man'.

'~(pfile)' : will match all names except 'pfile'. (NB: this
        expression is NOT the same thing as '~pfile', see
        section 2.3.1.6 for details)

'(*.c|*.h|*.doc|ab*)' : will match all files ending in '.c',
        '.h' or '.doc' and all files beginning with 'ab'.

Typing   an   action-file   spec   of   '(xxx|yyy|zzz)'   is functionally
equivalent to writing 'xxx yyy zzz' (xxx, yyy, zzz can be any valid
patterns, including patterns with parentheses).

Parentheses can be nested.

1.31   Tilde (~)

The tilde negates the immediately following expression. It negates
ONLY the immediately following token or paranthesized expressiom, not
the entire following expression as some people think.

 EXAMPLE

  '~x?' : matches any two-letter name except those starting with
        'x'.   For example 'ah', 'ko' or 'ba' but not 'x0' or
        'xi'.

  '~(x?)' : matches anything except two-letter names starting
        with 'x'.   For example 'xaa' or 'ab' but not 'xa' or
        'x9'.

  '~(#?)' : matches nothing at all.   (The tilde negates the
        '#?', which matches all names).

  '~lha' : matches all strings that doesn't begin with 'l', and
        ends in 'ha'. For example 'uha', 'why_lha' but not
        'lumbha' or 'lha'.

1.32   Percent sign (%)

    The percent sign represents the empty string. i.e. it matches 0
characters always.   It is only useful in parenthesized expressions and
must not follow a the '#' token ('#%' would be a rather pointless
pattern, since the % always matches exactly 0 characters).

 EXAMPLE

  'lha(.doc|.man|%)' : matches 'lha.doc', 'lha.man' and 'lha'.

  'l%u%a' : matches 'lua' only; the percent signs are totally
  irrelevant here and may just as well be omitted.

1.33   National characters

    LhA correctly converts national characters to lower- or uppercase
when the '-Qn' option is enabled. This causes some trouble on older
versions of the KickStart since these did not correctly convert national
characters to uppercase when calculating the filename hash value. The
'-Qn' option should not be used on OFS or FFS disks.

1.34   Commands

    This section describes the commands for archive manipulation and
maintenance LhA provides. See section 2.1.2 (Specifying commands) for
details on how to specify commands on the command line.

1.35 `a' Add files to archive

Obviously, this command adds a number of files to one or more
archives. If the specified archive does not already exist, then it will
be created. You cannot add files to an archive if these already exist
in the archive. If you attempt to do so, a warning will be issued, but
LhA will continue adding the other files you have specified.

Only the filenames are stored by default, if you want to preserve
some disk structure and directory names, you will have to use the -x
option to turn path preservation on. If you want to archive entire
subdirectories recursively you can use the -r option, which will turn on
the -x option automatically. These options are explained in section
2.5.

 EXAMPLE

  'LhA a myarchive dict.txt' would add the file 'dict.txt' to the
  archive 'myarchive.lha'.

  'LhA a arc.lzh *.c *.h' would cause all files in the current
  directory ending in '.c' or '.h' to be added to the archive
  'arc.lzh'

  'LhA -r -0 arch *.c' would cause all '.c' files in the current
  directory and all it's subdirectories to be added to the archive
  'arch.lzh' using the -lh1- (LhArc 1.x) compression method.

  'LhA -r archive src:(lharca|lha)/*.[cha] asrc:*.asm' would cause all
  '.c', '.h' and '.a' files in the 'src:lharca' and 'src:lha'
  directories and subdirectories, as well as all '.asm' files in the
  'asrc:' directory, to be added to the archive 'archive.lha'.

1.36 `c' Concatenate/Append archives

With this command it is possible to concatenate several archives to
one or to append several archives to the end of another. Currently, LhA
does not check for duplicate files, so if two archives contain a file
with the same name there will be two entries with the same name in the
resulting archive.

Concatenating and appending works just as if you had extracted all
files from the archives and then moved them all to the destination
archive - except that there is no decompression/compression involved in
the operation.

To combine (concatenate) several archives into a new archive you
specify a non-existant or empty archive as the working archive - this
file will then contain the resultant archive.

To append archives to the end of an archive, specify the archive to
append to as the working archive - the remaining archives will then be
appended to this archive.

EXAMPLE

  'LhA c ram:new arc:csrc arc:csrc2' would combine the two archives
  'arc:csrc.lha' and 'arc:csrc2.lzh' into one archive named 'ram:new.lha'.

  'LhA c arc:csrc arc:csrc2' would yield the exact same result as the
  above command but the resulting archive is in 'arc:csrc.lha' instead
  ('csrc2.lzh' is appended to the end of 'arc:csrc.lha').

   Note that you can use wildcards to specify the files to append /
concatenate.

1.37 `d' Delete files from archive

   This command removes one or more files from an archive. Please note
that the message about 'packing' does not mean that LhA compresses the
files once more.. Just that it removes the unused files from the
archive (packs the remaining files closer).

                              NOTE

    Files deleted with the 'd' command cannot be recovered
    from the archive file in any way. Once a file is
    deleted from an archive it is gone forever.

1.38 `e' Extract files from archive

   This command is used to extract files from an archive. It works just
like the 'x' command, except this command takes the '-x' option into
consideration (the 'x' command assumes it is set). If the '-x' option
is disabled, files are extracted without their pathnames, and if it's
enabled LhA will extract all files with the pathnames and create the
needed directories if they do not already exist.

EXAMPLE

  'lha -x0 e foo.lzh ram:' will extract all files from the archive
  'foo.lzh' to ram:, without paths (all files will be put in the ram:
  root directory).

  'lha x foo.lzh *.c ram:' will extract all files ending in '.c' to
  ram:, with paths – i.e. it will recreate the original directory
  structure.

  See the tutorial section for more examples

1.39 `f' Freshen files in archive

This command is used to freshen files in an archive. I.e. replace
older files in the archive with new files from the current directory.
Pathnames are considered unless the '-x' option is disabled explicitly
(with '-x0'). This command never adds any files to an archive, it just
replaces those files that have older modification dates than the
corresponding files in the current directory.

EXAMPLE

  'lha f /aab/lha' will freshen all files in the archive '/aab/lha.lha'.

  'lha f /aab/fsys *.[ch]' will freshen all '.c' and '.h' files in the
  archive '/aab/fsys.lha'.

This command automatically enables the '-x' option unless it is
explicitly disabled on the command line with '-x0'.

1.40 `h' Hunt for diffs arc <-> filesys

This command is used to see what files in an archive have been
changed since the files were archived. The '-D' (display type) option
has a special meaning with this command; The listing format is as
follows:

'-D0' (default) : Each differing file is listed with the name on the
right and a 'checklist' on the left with x-es in the appropriate
positions indicating what differs between the archive and the
filesystem. 'Tm' means the last modification date differs, 'Sz' means
the size is different, 'Pr' means the protection bits has changed, 'Fn'
means the filenote has changed, and 'Del' means the file does not exist
anymore.

'-D1' : Every differing file is listed with the name on the left
followed by a brief description of what differs. If more than one thing
differs a new line will be used for each differing attribute.

'-D2' : As '-D1' but all differing attributes are listed on the same
line.

'-D3' : Only the differing filenames are listed, one on each line.

If no directory is specified on the command line, LhA assumes you
want to compare the archive to the current directory. The directory to
compare to is specified the same way as the destination directory with
the 'e' and 'x' commands.

EXAMPLE

  'lha h arc:utils.lha sys:utilities/' would compare all files in the
  archive 'arc:utils.lha' to the corresponding files in the
  'sys:utilities' directory, reporting all differences.

  'lha -x0 h src:misc #?.c misc:' would compare all files with names

ending in '.c' in the 'src:misc.lha' archive to the corresponding
files in the 'misc:' directory.

'lha h dl:backup' would compare the files in archive 'dl:backup.lha'
to the files and directories in the current directory.

1.41 `l` List archive contents (terse)

This command gives a terse list of the contents of an archive file,
including file names (without paths), original and compressed length,
last modification date and compression ratio.

Files with pathnames are indicated by having a '+' character in front
of the name. See example below.

Filenotes are NOT displayed when using this command, use the 'v' or
'vv' command to display those.

The action file specification is used to determine what files to
list. If no filespecs are given, all files will be listed.

 EXAMPLE

  1> lha -N l dl:c64new

  Listing of archive 'dl:c64new.lzh':
  Original Packed Ratio   Date   Time   Name
  -------- ------- ----- --------- -------- -------------
     36098   26979 25.2% 20-Oct-91 22:40:16 +Stormlord
       482   293 39.2% 20-Oct-91 22:41:36 +Stormlord.info
     23016   12100 47.4% 21-Oct-91 08:28:18 PlaySID
  -------- ------- ----- --------- --------
     59596   39372 33.9% 25-Oct-91 21:22:48   3 files

The '+' (plus) signs in front of the first two names indicate that
the file has a path which is not displayed with the 'l' command (use
the 'v' or 'vv' command to display pathnames as well). The '-N'
suppresses the copyright notice.

1.42 `lq` List archive (terse-quick)

This command works just like the 'l' command, but the only
information listed is the filenames without paths. Empty directories
are displayed as an empty line.

1.43 `m` Move files to archive

This command works just like the 'a' command, but the source files
are deleted after successfully adding them to the archive.

EXAMPLE

  'lha m includes.lzh src:*.[hi]' will move all files in directory
  'src:' having filenames ending in '.h' or '.i' to the archive
  'includes.lzh'.

  'lha m myarc.lzh lhb_log.911012 lhb_idx.911012' will move the two
  specified files ('lhb_log.911012' and 'lhb_idx.911012') to the archive
  'myarc.lzh'.

1.44 `p' Print files to stdout

   This command works just like the extract ('e', 'x') commands, but
sends the extracted output to stdout (normally the console or output
redirection file).

1.45 `r' Replace files

   This command works just like the update/add commands but replaces the
files that already exist in the archive regardless of the file
modification time. (Using the 'u' command together with the '-Qr'
option is equivalent to using this command).

1.46 `t' Test archive integrity

   This command tests the specified archive's integrity by extracting
the files they contain to nowhereland, i.e. the data is decompressed
only, not written to any file. This command only works on entire
archives, i.e.   you cannot just test one file in an archive. If this
command fails, the archive is corrupted, and a warning return code is
returned.

EXAMPLE

  'lha t work:arcs/*' will check the integrity of all archives in
  directory 'work:arcs'.

  'lha t s:envarc.lzh' will check the integrity of 's:envarc.lzh'

  'lha -R t dh0:*' will check the integrity of all archives on the
  'dh0:' volume ('-R' = Collect archives recursively).

1.47 `u' Update archive

   As the command name implies, this command updates archives. It adds
files that are not yet in the archive and replaces existing but older
files. The last modification date for files are used to determine which

file is the newest one.

  EXAMPLE

  'lha u /aab/lha.lzh *.c' will update archive '/aab/lha.lzh' with all
  '.c' files in the current directory.

1.48 `v' List archive (verbose)

    This command works just like the 'l' command, but displays the full
pathname of the file, while 'l' only displays the name node without
path. Another difference between 'l' and the 'v'/'vv' commands is that
the 'l' command does not show filenotes. Filenotes are displayed on a
separate line with a colon (':') in front of it, just like the AmigaDOS
'list' command.

    The action file specification is used to determine what files to
list. If no filespecs are given, all files will be listed.

  EXAMPLE

  1> lha -N v dl:c64new

  Listing of archive 'dl:c64new.lzh':
  Original Packed Ratio   Date   Time   Name
  -------- ------- ----- --------- -------- -------------
     36098   26979 25.2% 20-Oct-91 22:40:16 S/Stormlord
       482   293 39.2% 20-Oct-91 22:41:36 S/Stormlord.info
     23016   12100 47.4% 21-Oct-91 08:28:18 PlaySID
  : New version with 'equalizers'
  -------- ------- ----- --------- --------
     59596   39372 33.9% 25-Oct-91 21:22:48   3 files

  The '-N' suppresses the copyright notice.

1.49 `vq' List archive (verbose-quick)

    This command works just like the 'v' command, but the only
information listed is the filenames including path.

1.50 `vv' List archive (full)

    This command is just like the 'v' command, but displays all available
information in a slightly different format. The original and packed
size, last modification date and compression ratio is listed just as
with the 'v' command, plus file attributes ('Atts'), compression method,
file CRC and DOS ID for the OS the files were compressed on. If no DOS
ID is given in the archive (header level < 1), a question mark is
displayed. The most common DOS IDs are 'A', 'U' and 'M', where 'A' is
for AmigaDOS, 'U' is for **IX and 'M' is for MS-DOS. The filename

including path is displayed on a separate line. File notes are
displayed in the same way as the 'v' command does it, on a separate line
after the filename. The header level is also displayed, and if any
unhandled extended headers are found, an 'X' will be listed after the
DOS ID.

The action file specification is used to determine what files to
list. If no filespecs are given, all files will be listed.

 EXAMPLE

   1> lha -N vv dl:c64new

   Listing of archive 'dl:c64new.lzh':
   Original Packed Ratio   Date   Time   Atts   Method CRC L OS
   -------- ------- ----- --------- -------- -------- ------ ---- -----
   S/Stormlord
      36098   26979 25.2% 20-Oct-91 22:40:16 ----rwed -lh1- 2093 2 U X
   S/Stormlord.info
        482   293 39.2% 20-Oct-91 22:41:36 ----rwed -lh1- 710E 2 U X
   PlaySID
      23016   12100 47.4% 21-Oct-91 08:28:18 ----rwed -lh5- 89FF 0 ?
   : New version with 'equalizers'
   -------- ------- ----- --------- --------
      59596   39372 33.9% 25-Oct-91 21:22:48   3 files

   The '-N' option suppresses the copyright notice.

1.51 `x' Extract files with path

This command works exactly the same as the 'e' command, but it always
extracts files with paths (i.e. same as using the 'e' command with '-x'
option on), regardless of the state of the '-x' option.

1.52 `y' Copy archive with new options

This command takes an archive as input, and rewrites the selected (or
all, if none specified) files with the new options given on the command
line or in environment variables. This can often be useful. A couple
of examples will surely help to clarify;

 EXAMPLE

  'lha -H1 y dl:#?' will convert all archives in the 'dl:' directory to
  archives with level-1 headers.

  'lha -x0 y ram:files.lha *.c' will remove all paths from all files
  with names ending in '.c' in the archive 'ram:files.lha'.

                                   NOTE

```
     LhA  currently ignores the compression method setting,
     so  this  command  cannot  be  used  to re-archive old
     -lh1-  archives  to  new  -lh5- archives or vice versa.
     This will be possible in a future release.
```

1.53 Options

   This   section describes the various options that are available to you
when using LhA.   For a detailed explanation on how to enable/disable
specific   options   and where you can specify options, see section 2.1.1.
The letters in parantheses indicate what commands the options affect.

```
     Code   Commands
     -----  ------------
     (add) a,u,f
     (all) all commands
     (ext) e,x
     (upx) a,u,f,e,x
     (upd) a,u,f,d
```

1.54 `-a' (upx) Preserve file attributes

   This   option,   when   enabled,   will   make   LhA store and restore file
protection flags. The eight attributes are listed below:

   r: Read - This flag is set for files which are readable (a
       file is read-protected if the flag is unset).

   w: Write - This flag is set for files which are writeable
       (a file is write-protected if the flag is unset).

   e: Execute   - This flag is set for files which are
       executable (binary load files or shell scripts must have
       this bit set).

   d: Delete   -   This flag is set for files which are
       deleteable (a file is protected from deletion if this
       flag is unset).

   a: Archived   -   This flag is used by harddisk-backup
       programs (and optionally LhA) to indicate what files
       have been changed since the last backup. If this flag is
       set it indicates that the file is unchanged, and if it
       is unset the file has changed since the last backup.
       The bit is cleared whenever a write is made to the
       file.

   p: Pure - This flag is set for binary load files which are
       pure (i.e. multitasking reentrant), and can be made
       resident with the AmigaDOS 'resident' or equivalent
       command.
```

s:  Script - This flag is set for shell script files.

h:  Hidden  -  This  flag  is  set for files that should not
    show  up  on  directory  listings. It is not supported by
    the  current  release of the AmigaDOS shell/CLI commands,
    and should thus not be used.

Please  refer  to an AmigaDOS manual for more detailed explanation of
the various file protection flags.

If the option is disabled (by issuing '-a0' on the command line), the
protection  flags  are  set  to '----RWED' for all extracted and archived
files.  Important:  You MUST have this option enabled both when
archiving and extracting to preserve file attributes correctly.

<div align="center">NOTE</div>

Use  this option only if you know that the archive has
been  compressed or will be decompressed with an Amiga
archiver,  since  the  attribute  field  format  is
different  on different operating systems.  If you use
archive  headers  of  level  1  or higher you need not
care  about  this since the archiver then detects what
OS  the  archive  was  created  on  and  only uses the
protection  flags  if  it  is  the  native OS.  Always
leave  this  option enabled when using archive headers
of level 1 and higher!

This  option  is  enabled  by  default  when  archiving (a,f,u,m) and
disabled by default for all other commands.


## 1.55  `-A' (upd) Set archive attributes

When this option is active, LhA will set the file protection flags of
all archives it updates to '----RW-D'.

This option is OFF by default.


## 1.56  `-b' (all) Set I/O buffer size

This  option  will  set  the  size  of  the I/O buffers LhA uses when
reading  and  writing  to archive files.  You can set the buffer size to
anything  from  8KB to 64 KB.  Larger buffers normally makes LhA operate
slightly faster (depends on the nature of the archive and what files are
selected).

EXAMPLE
 'lha  -b64  a  archive.lzh  hubba' :  Will  add  file 'hubba'  to
 'archive.lzh' using an I/O buffer of 64K.

<div align="center">NOTE</div>

Running  LhA with a small I/O buffer on an accelerated
(68020  and  up)  Amiga  will  degrade  compression  /
decompression  performance significantly! The default
buffer  size  of  32KB  is  enough  in most cases, and
works  well  on  an unaccelerated Amiga as well.  Also
note  that when running LhA and doing all work on some
ram  disk,  the I/O buffer size is less important, and
it  is  unnecessary  to  run  with a large buffer. The
default  buffer  size  of 32K is a good choice in most
setups.

The default buffer size is 32K (32768 bytes)

## 1.57  `-B' (upd) Keep backup of archives

When  this  option  is enabled, LhA will always keep a backup copy of
the  archive  whenever  a file is removed from it by the delete, update,
freshen  or  replace  commands.   The  backup  archive  is  named
`<arcname>.bak' (note that the `.lzh' or `.lha' suffix is *NOT* replaced
by  the  `.bak'  suffix – rather, the `.bak' suffix is always appended at
the end of the filename).

This option is OFF by default.

## 1.58  `-c' (all) Confirm files

When  this  option is active LhA will ask you for confirmation on all
files and archives that are acted upon.

This option is OFF by default.

## 1.59  `-C' (ext) Clear arc-bit on extract

When this option is active LhA will mask the A–protection bit for all
files  it  extracts.  This is useful when extracting files from archives
to  a harddisk, since the extracted files would not be recognized as new
or changed files by the backup program if the A–bit was set.

This option is ON by default.

## 1.60  `-d' (upd) Archive date=newest file

When this option is active LhA will set the last modification date of
the  archive  to the same date as the last modified file in the archive.
This  more accurately reflects the real age of the archive contents than
the date of the last archive update.

This option is OFF by default.

## 1.61 `-D' (all) Alternate progress display

   This switch is used to change the look of the byte progress indicator
that  LhA displays when it is compressing or decompressing files.  There
are  several  different  types  of  progress indicators, you can specify
which one you want with a digit after the '-D' string.

0:    This  is the default progress indicator, it displays how many bytes
      of  the  file LhA has processed, and how many bytes there is in the
      file like this:

      (xxxxxx/yyyyyyy) where x = bytes processed, and y = total bytes in
      the file.

1:    This  progress  indicator  simply  shows  a 'rotating line' that is
      rotated  45  degrees  every  time the progress indicator display is
      updated).

2:    This  progress indicator shows a percentage of how much of the file
      LhA has processed.

3:    This  progress  indicator displays a growing bar that indicates how
      much of the file has been processed.

  EXAMPLE

   'lha  -D2  a src *.asm' will add files to the archive 'src.lha' with a
  percentage indicator (type 2).

                                NOTE

      When  used  with  the  'h'  command  this option has a
      slightly  different meaning. See the section about the
      'h' command for a detailed explanation.


   The default progress indication type is 0.


## 1.62 `-e' (add) Archive empty directories

   When  this  option  is  used together  with  the '-r' (collect files
recursively) option, LhA will archive all empty subdirectories.

   This   option  is  OFF  by  default  (empty  subdirectories  are  not
archived).


## 1.63 `-E' (ext) Touch extracted files

   When  this option is enabled, LhA will set the file modification date
of  all  extracted files to the current time.  This can be useful if you
do HD backups by date rather than by archive bit.

This   option   is   OFF by default (the original modification dates are
restored).


## 1.64   `-f' (all) Ignore filenotes


When  this  option  is  enabled,  LhA  will  not store or restore any
filenotes.   There  is  no real need to do this, since it does not cause
any  compatibility  problems  with  other systems because of the way the
filenotes  are  stored.   If  problems should arise anyway, try enabling
this  option  or  use  headers of level 1 or higher if the target system
supports it.

See the section about compatibility (1.7) for a discussion about this
and other compatibility issues.

This option is OFF by default (filenotes are stored and restored)


## 1.65   `-F' (all) Use fast progress display


In  this mode LhA uses a different method of display progress for the
extract  and  test  commands.  Normally, LhA emits a linefeed (LF) after
each  file  has been processed, thus advancing/scrolling the display one
line.   In  this mode LhA only emits a LF when an error occurs.  This is
useful if you are testing or extracting files with a lot of small files,
and the scrolling takes more time than the actual decompression!

                              NOTE

   If  you  use  the  default style progress display on a
   very  fast  Amiga  system  (68020+), beware  that the
   scrolling  of  the  screen may actually take more time
   than  the  actual  decompression!  This is especially
   true  for  archives  with  many small files.  So don't
   use  it  unless  you  really  _have_ to see what files
   have  been  processed.   LhA scrolls  the  display
   whenever  an  error occurs on a file, so you still can
   see  when  an  error  occurs  (better, even, since the
   only  filenames that remain on screen after the action
   is complete are those that failed!).

This option is OFF by default (use old style progress indication).


## 1.66   `-g' (add) Garble files with password


This option is not available in the current version.

## 1.67  `-G' (ext) Only extract newer files

When  this  option  is  used LhA will only extract files that already
exists and have a last modification date that is newer than the existing
files.

This option is OFF by default.

## 1.68  `-h' (add) Disable homedirectories

When  this option is enabled, the homedirectory specification feature
of LhA is disabled.

This option is OFF by default (homedirectories are recognized).

## 1.69  `-H' (add) Write header level

With this switch you can select which header types to use.  The valid
header  levels  are  currently  0,1  and 2.  Please refer to the section
about  header  levels  for a more detailed explanation about the various
header types.

The default header level is 0.

## 1.70  `-i' (all) Read filelist from file

With  this  option  you  can  include an action file list from a file
instead of specifying all action files on a command line.

 EXAMPLE

 If the file `ArcFList' contains the following lines:

 ---> Start of ArcFList data   (this line is NOT in the file)

 LhA.c ArcList.c FSys/*.(c|h|i|asm|prf|man|doc|txt)

 ---> End of ArcFList data     (this line is NOT in the file)

 The following command line:

 `lha -iArcFList u /aab/lha.lzh'

 Will do the same thing as this command:

 `lha u /aab/lha.lzh LhA.c ArcList.c FSys/*.(c|h|i|asm|prf|man|doc|txt)'

                         NOTE

This   command   works   almost   exactly like entering
the following command line:

LhA ? ???? @file

Thus  you  can include options in your -i file. The
only  difference is that the -i file cannot contain
a  destination  directory  specification  while you
can  do this with the @file method. The destination
directory  will  always  be  taken from the command
line when using the -i option.

See  the  section  about  '@'(include)-files  for an alternate way of
doing this.

## 1.71  `-I' (all) Ignore LHAOPTS variable

When  this option is specified, LhA will not try to read the defaults
from  the  LHAOPTS local or global environment variable.  Note that this
option  is  special because it has to be specified directly after a dash
('-') on the command line.

This option is OFF by default.

## 1.72  `-k' (all) Keep partial files

This  option will,  if it's  enabled,  prevent LhA  from  deleting
temporary  files  when  an  error  occurs.  Normally temporary files that
fail  the CRC check, cause I/O errors or are interrupted with CTRL-C are
deleted  before  exiting LhA with an error message, with this option you
can  force  LhA to keep those (often) partial files.  This can be useful
when  trying  to recover data from corrupted archives - LhA will attempt
to  extract  the  data from the erraneous archive file and put a special
filenote  on  the  file  to  indicate  that  it failed the CRC check and
probably is corrupted.

NOTE

Please   that  in  the  current  release,  for  certain
errors  all data that has been extracted may not be in
the  partially extracted file, because of internal I/O
buffering.  In  this  case,  set the I/O buffer to the
smallest  value  possible  (8KB) to recover as much as
possible.  Because  of  this,  small  files may not be
recovered  at  all.  This  only applies to LHA (-lh5-)
compression,  LhArc  compressed files will always have
all extracted data in the partially extracted file.

This option is OFF by default (partial files are deleted).

## 1.73 `-K' (move) Kill empty directories

When this option is used together with the move ('m') command LhA will delete all directories that are empty after moving all files to the archive. Useful for moving an entire subdirectory tree with the '-r' (collect files recursively) option.

This option is OFF by default (empty directories are not deleted).

## 1.74 `-l' (ALL) Make filenames lowercase

This option, when active, will cause LhA to convert all filenames to lowercase. This is useful when extracting files from archives created on MSDOS systems, whose filenames are all uppercase, which look completely braindead (IMHO). Use this option to make them look nicer!

 EXAMPLE

  'LhA -l x myarc' will extract all files from 'myarc.(lzh|lha)', making
  all filenames lowercase.

  This option if OFF by default

## 1.75 `-L' (ALL) Create filelist

When this option is enabled, it will cause LhA to create a list of the files it has acted upon (i.e. what files in the last operation that matched the action file specification you gave on the command line). The name of the list file must follow immediately after the '-L' string. If you need spaces in the filename, enclose the name in double quotes.

 EXAMPLE

  'lha -Lram:ListFile d src.lzh *.asm' will delete all files in
  'src.lzh' with names ending in '.asm' and create a list of the deleted
  files in the file 'ram:ListFile'.

  'lha -L"ram:List File" u src.lzh *.asm' will update 'src.lzh', and
  create a list of the files that were added/replaced in the file
  'ram:List File'.

                    NOTE

    The file that this option creates is a plain ASCII
    file with every name on a separate line. The files
    created by this option are suitable for use as action
    or exclude lists for LhA using the '@' or '-i'
    options.

  This option is OFF by default (no filelist created).

## 1.76  `-m' (ALL) No messages for query

When this option is active LhA will suppress all queries that
normally are issued before overwriting existing files for example.
Enabling this option will also cause LhA to ignore TelOps (autoshow
files).  When this option is on you LhA will behave like you choose the
default action in response to all the queries (yes).  This option is
automatically enabled if the standard input is not interactive (if run
in the background for example).

This option if OFF by default.

## 1.77  `-M' (ext) No autoshow files

When this option is enabled, LhA will suppress the display of
autoshow files (files with names ending in `.displayme').

NOTE

Autoshow files are also suppressed if one or more of
the `-N', `-q' or `-m' options are enabled.

This option is OFF by default (autoshow files are displayed).

## 1.78  `-n' (upx) No byte progress indicator

When this option is enabled, the byte progress indicator is disabled.
LhA will still display what file it is working on however, use `-N' to
disable all progress indication.

This option is OFF by default.

## 1.79  `-N' (all) No progress indicator

This option is similar to the `-n' option, but supresses higher-level
progress indication (i.e.  the display of what file LhA is bashing).  It
also disables the short copyright banner that is printed at each
invokation otherwise.

This option is off by default (file progress indication ON).

## 1.80  `-p' (ALL) Pause after loading

When selected, this option will cause LhA to pause and wait for the
user to press any key before executing a command.  This is useful for
users with floppies, who can then swap disks after LhA has been loaded
and is waiting for a keypress.

This option is OFF by default.


## 1.81 `-P' (ALL) Set task priority

This option is used to set the LhA process priority. The priority may be set to any value in the range -5 to +5, including 0. The higher priority you give LhA, the more CPU time it will grab (processes with lower priority will almost never get the chance to run since LhA is very processor-intensive). Setting it to a low value (like -5) will make LhA only use the processor time that nobody else wants (nice when running LhA as a background task while running a comm program).

The priority must be specified with a single (optionally prefixed with a minus sign for negative priority) digit immediately after the P as in:

 EXAMPLE

  'lha -P-1 a nonsense.lzh bogus.txt' will make LhA add the file
  'bogus.txt' to the archive 'nonsense.lzh', running at priority -1.

The default priority is inherited from the calling process (i.e. the CLI or program that called Execute()/RunCommand() ). This is usually zero (0).


## 1.82 `-q' (ALL) Be quiet

This option will supress ALL messages from LhA.

This option is OFF by default


## 1.83 `-Q' (ALL) Alternate option set

This option character ('Q') will cause all following option characters until next space character to be interpreted as extended options. These are documented at the end of this section.


## 1.84 `-r' (add) Collect action files recursively

When this option is used, LhA will recursively collect files from subdirectories.

 EXAMPLE

  'lha -r a ram:disk1 df0:' will archive all files on the disk in drive
  0 to 'ram:disk1.lha'.

'lha  -r  a  ram:disk2src df0:*.c' will archive all '.c' files on df0:
to 'ram:disk2src.lha'.

'lha -r a ram:exthup hd:prg/src/ lha/*.[chasi] lhi/*.[chasi]' will add
all  '.c',  '.h',  '.a',  '.s',  '.i'  files  in  'hd:prg/src/lha'  and
'hd:prg/src/lhi'  and their subdirectories.  The 'hd:prg/src/' part of
the   names   will   not   be   stored   in   the   archive (home directory
'hd:prg/src/' was specified).

                                NOTE

    Files  that are specified explicitly (i.e. without any
    pattern  matching)  are looked for only in the current
    (home)   directory,   while   patterns   are   used   for
    matching  in  all  subdirectories.  If a directory is
    specified   explicitly   without   any   following   file
    pattern (like  in  'lha -r a ram:test sys:l') it will
    be  treated as if a '/*' was appended to the directory
    name  -  i.e.  all  files  in  the  directory and it's
    subdirectories will be archived.

    This  option  is  OFF  by  default.  Note that the  '-x' option is
automatically enabled when the -r option is used.  If you do not want to
store pathnames simply specify '-x0' on the command line.


## 1.85  `-R' (ALL) Collect archive files recursively

    When  this  option  is  enabled  LhA  will  search  for archive files
recursively  using  the  archive file specification given at the command
line.  This works like the '-r' option but for archive files.

 EXAMPLE

  'lha  -R  l  dh0:files/a*' will list the contents of all archive files
  whose   names   begin   in   'a'   in  directory  'dh0:files'  and  its
  subdirectories.

  'lha  -R  l  *'  will  list  the  contents of all archive files in the
  current directory and its subdirectories.

  'lha  -R  l  myarc'  will  list  the  contents  of all archives called
  'myarc.lzh'  or  'myarc.lha'  in  the  current  directory  and  its
  subdirectories.

   This option is OFF by default.


## 1.86  `-s' (add) Add files with a-flag unset

    When  this option is active, LhA will only add files which have the A
(for  Archived)  file  protection  flag  unset.  This is useful for doing
incremental backups together with the '-S' option.

This option is OFF by default (add files regardless of file
protection flags).

## 1.87 `-S' (add) Set A-flag on archived files

When this option is on, LhA will set the A (for Archived) file
protection flag on all files that are added to an archive. This can be
used to simplify automatic backups when used together with the -s (Add
files without A-flag only). See previous section for more details.

This option is OFF by default.

## 1.88 `-t' (ext) Only new files

When this option is active, LhA will not overwrite or replace any
files.

                              NOTE

This option overrides the '-T' option.

This option is OFF by default.

## 1.89 `-T' (upx) New and newer files

When this option is active, LhA will overwrite or replace files that
already exists and are older than the current file, and create files
that does not already exist.

                              NOTE

This option overrides the '-t' option.

This option is OFF by default.

## 1.90 `-u' (ALL) Make filenames uppercase

This option, when active, will force LhA to convert all filenames to
uppercase. This can be useful when making archives that are supposed to
be used on MSDOS-Systems running LhArc/LHA. While these have no
problems with extracting files with mixed-case filenames, the pattern
matching routines will not work correctly.

This option is OFF by default.

## 1.91  `-U' (upx) Set update interval

This  option  is  used  to  set  the interval (in bytes) at which LhA
updates  the  byte  progress  indicator.  The  desired interval must be
expressed in kilobytes, and must immediately follow the 'U' character.

 EXAMPLE

  'LhA  -U4096 a bar.lzh *.c' will add all c-source files in the current
  directory to 'bar.lzh' with a progress indicator interval of 4096 (4K)
  bytes.

  'LhA -U32768 a bar.lzh *.c' will do the same as the example above, but
  with a update interval of 32768 bytes (32K).

                          NOTE

     This  option does currently not affect the update rate
     of  the  LHA decompression ('-lh5-' compression mode).
     When  LhA  decompresses  files  with  this compression
     mode,  the  update  rate  will  be whatever I/O buffer
     size  is  used  (set with the '-b' option). The reason
     of  this  behaviour  is  that  the  normal  progress
     indication would slow down decompression.

The  default  update  interval is 8192 (8K) bytes for -lh1- and -lh5-
compression  and  4096  (4K)  bytes for -lh1- decompression. The update
rate  for  -lh5-  decompression  is  determined  by  the I/O buffer size
setting (see note above).

## 1.92  `-v' (add) Set compression speed

This  option  can  be  used  to  increase or decrease the compression
speed.  -v0 is the slowest, and -v9 is the fastest. As usual you can't
get  anything  for  free,  so compression performance is slightly looser
with  -v9  than  with -v0 but the difference in speed can be significant
(especially  with  some binary graphics data). Higher compression speed
is attained by using less statistics in the compression phase.

The default compression speed is 5 - best in 99% of all cases.

## 1.93  `-V' (all) Enable multi-volume archives.

This  option enables the multi-volume feature of LhA. Please consult
the section about multi-volume archives for more information. Also read
the  section  about the '-Qv' option. Further options must be separated
from  the  'V' by at least one whitespace character. The desired volume
size in KB should be specified after the 'V' character. If you want LhA
to  automatically  detect what volume size it should use, use '-Va' (for
'use all available space').

 EXAMPLE

'LhA  -Va  a  df0:MyArc *.c'  would  archive all files in the current
directory  with  names  ending  in '.c' to DF0:  if the disk should get
full  before  the archive is finished LhA will prompt for a new disk to
be inserted.

This option is OFF by default.

## 1.94   `-w' (upd) Set work directory

This  option  is  used to specify what directory LhA should use to store
temporary  files.  Temporary  files  are  created  when adding files to
archives,  or  when  updating  an  archive in some way (like deleting or
freshening  files).  The  work  directory  name  must  be  specified
immediately after the '-w' string.

 EXAMPLE

 'LhA  -wrad:tmp  a  MyArc.lzh  *'  will use the directory 'rad:tmp' as
 temporary  storage  location  when  adding  all  files  in the current
 directory to the archive 'MyArc.lzh'.

By  default  LhA uses the 'T:' directory for temporary files, if this
assign or device does not exist, LhA will use the current directory.

## 1.95   `-W' (add) Exclude filenames

This option is not available in the current version.

## 1.96   `-x' (all) Preserve and use pathnames

As  of  LhA V1.30, this option comes in three flavors, which mode LhA
will depend on the digit (if any) that follows the 'x'.

'-x1'  or  '-x':  When  this  option  is  enabled, LhA will use and
preserve  pathnames  when  extracting  and  archiving  files.  When
extracting, LhA will create the directories that does not already exist.
Use  this  option  when  you  want to preserve some directory structure.
This option is automatically enabled when the '-r' option is used.

'-x2':  In this mode, which is only useful with the extract commands,
LhA  will  use the full paths of the files in the archive when selecting
files  to  extract,  but  disregard  them  when extracting. Useful when
several files with the same filename (but different paths) exists in the
archive.

'-x3':   This  mode  is  the opposite of '-x2'. LhA disregards paths
when  selecting files to extract, but uses them when extracting. Useful
when you're too lazy to remember the exact name including path.

```
EXAMPLE
```

'LhA -x2 e dl:rexx.lzh examples/Main.c ram:' would extract the file
'examples/Main.c' from the archive to 'ram:Main.c'.

'LhA -x3 e dl:src.lzh #?Main#? ram:' would extract all files with
names containing 'Main'.   Notice that this is not equivalent to the
'LhA x dl:src.lzh #?Main#? ram:' since the latter would extract files
like 'dir/Maindir/file1.h' as well.

This option is disabled ('-x0') by default for update operations and
enabled ('-x1') by default for extract operations.

1.97   `-X' (ALL) Do not append suffix

When this option is enabled, LhA will not append an '.lzh' or '.lha'
suffix to the given archive name. The default behaviour is to append a
suffix of '.lha' or '.lzh' (suffix is chosen depending on compression
mode) if the name does not already have an extension.

This option is OFF by default (suffixes are appended).

1.98   `-y' (all) Always append suffix

When this option is enabled, LhA will always append a '.lzh' or
'.lha' suffix to the archive name, even when the archive name already
contains a suffix.

This option is OFF by default (a suffix is appended only if there is
no suffix in the archive name already).

1.99   `-Y' (add) Store big files with ratio

When this option is enabled, LhA will store big files (>32KB) without
compression if compression ratio is lower than 3%. This is because
extraction times of these files are long on slower machines.

This option is OFF by default (all files are compressed).

1.100   `-z' (add) Do not compress files

This option, when active, will force LhA to store all updated or
added files in the archive without attempting to compress them. Useful
for making fast backups where archive size is of no importance. It is
not advisable to use this option when making archives for distribution
via modem or networks since the archive will end up much larger than if
it was compressed.

```
EXAMPLE
  'lha -z a foo.lha *.bmp' Will store all files in the current directory
  with  a  suffix  of  '.bmp'  in  the  archive  file  'foo.lha'  without
  compressing them.
```

This option is OFF by default.

1.101 `-Z' (add) Compress archives

This option will cause LhA to attempt compressing already compressed
files.

By default, LhA will not attempt to compress files which are already
compressed (typically archive files or picture files in GIF or JPEG
format).   The file type is determined from the suffix, and files with
names ending in '.lzh', '.lha', '.zoo', '.zip', 'arj', '.arc', '.dms',
'.wrp', '.lhw', '.zap', '.pak', '.pp', '.gif', or '.jpg'.

The reason why already compressed files should not be compressed is
that the number of bytes gained by this is so small that it is not worth
the time spent compressing/decompressing it.

This option is OFF by default.

1.102 `-0' (add) Use LhArc 1.x compression

This option causes LhA to use the old -lh1- compression method when
updating archives. This compression method is slightly faster than the
normal -lh5- compression but has looser compression and is much slower
to decompress.

When this compression mode is used, LhA defaults to appending a
suffix of '.lzh' when creating archives.

When this option is specified, option '-2' is automatically
deactivated.

By default the -lh5- compression is used.

1.103 `-1' (add) Use LhA compression (-lh4-)

This option causes LhA to use the new -lh4- compression method when
updating archives. This compression method is slightly faster than the
-lh5- compression but has looser compression and is generally slightly
slower to decompress.

When this compression mode is used, LhA defaults to appending a
suffix of '.lha' when creating archives.

1.104   `-2' (add) Use LhA compression (-lh5-)

This option causes LhA to use the new –lh5– compression method when updating archives. This compression method is slightly slower than the old –lh1– compression but has tighter compression and is much faster to decompress.

When this compression mode is used, LhA defaults to appending a suffix of '.lha' when creating archives.

When this option is specified, option '-0' is automatically deactivated.

This is the default compression mode.

1.105   `-Qa' (all) Use simple console I/O

When this option is enabled LhA will not try to do any fancy stuff like examining the size of the console window, or turning off or repositioning the cursor. Enabling this option also disables the byte progress indicator (like with '-n'), since this requires cursor repositioning.

This option is OFF by default.

1.106   `-Qb' (ext) Test archive before extract

When this switch is enabled LhA will test an archive's integrity before extracting. If the archive fails the integrity check, the archive is not extracted from at all. Useful in certain FIDO BBS setups.

This option is OFF by default.

1.107   `-Qd' (ext) Delete autoshow files

When this option is enabled LhA will delete autoshow files after displaying them.

This option is OFF by default.

1.108   `-Qh' (add) Set Huffman buffer size

This option can be used to set the size of the buffer used in LHA compression (default or selected with the '-2' or '-1' options) for collecting statistics. The size of this buffer affects the compression ratio in unpredictable ways (you cannot tell with certainty whether a

large buffer will be better or worse). As a general rule, keep this at the default, but if you are compressing homogenous data with a relatively fixed relative frequency of symbols (like text files), setting this to a large value will improve compression. Binaries generally compress best with the default setting.

The Huffman buffer may be of any size between 4K and 64K and must be specified immediately following the '-Qh' string, in kilobytes.

EXAMPLE

'LhA -Qh32 -2 a foo.lha *' will compress all files in the current directory using a Huffman buffer size of 32768 (32K) bytes.

'LhA -Qh4 -2 a foo.lha *' will compress all files in the current directory using a Huffman buffer size of 4096 (4K) bytes.

The default Huffman buffer size is 16K.

1.109 `-Qn' (all) Set national character mode

When this option is enabled, LhA will correctly convert national characters to upper/lowercase. By default LhA does not convert any characters with the MSB set due to the fact that older (pre-2.1) filesystems does not correctly handle national characters when computing hash values. This switch should be used when national filesystems are used (NOFS/NFFS).

This option is OFF by default.

1.110 `-Qo' (all) Ignore options after command

When this option is enabled LhA will not search the command line for options beyond the archive name. This option is useful if you need to specify files with names beginning in '-'.

This option is OFF by default.

1.111 `-Qp' (move) Ignore delete protection flag

When you enable this option LhA will delete files with the delete protection flag set when using the 'm' (move) command.

This option is OFF by default (delete protected files are not deleted).

1.112 `-Qq' (add) Quick add

When this option is enabled, LhA will not scan through the archive
looking for duplicate files before adding to the archive. This can be
useful when adding one file at a time to a large archive, knowing the
archive does not contain a file by the same name (as is the case in some
FIDO BBS setups).

This option is OFF by default.

1.113 `-Qr' (add) Skip datestamp check

This option, when on, disables the datestamp comparison for the
update (`u') and freshen (`f') commands, so that the files that already
exist in the archive will be replaced regardless of file modification
dates.

This option is OFF by default for all commands but `r'.

1.114 `-Qv' (all) Set multivolume arc devices

With this option you can make LhA use several devices when creating
multivolume archives. LhA will use the devices you specify in sequence
and wrap around to the beginning when the last device has been used.
The devices should be specified WITHOUT a colon directly after the `-Qv'
string, separated by commas (`,'). When using this option you still
have to specify the first device to use in the archive name.

 EXAMPLE

 `LhA -Va -Qvdf0,df2,df3 -r a df0:Bak hd:#?' would create a multivolume
 archive starting on df0: and then use df2:, df3:, df0:, df2: and so
 on. Notice that we still have to specify `df0:' in the archive name
 specification.

1.115 `-Qw' (all) Disable wildcards

When you specify this option LhA will not do any wildcard matching.
This is useful for adding files with (illegal) names containing wildcard
characters (`()#?~%|*').

This option is OFF by default.

1.116 Autoshow files

Autoshow files are files that are displayed automatically to the user when extracting the file from an archive. LhA determines if a file should be displayed by looking at the filename; if the filename ends in '.displayme' then the file is displayed unless autoshow files have been disabled (with the '-M' option). Apart from being displayed on-screen, autoshow files are extracted just like normal files, without stripping off the '.displayme' part.

1.117 Residentability

LhA is multitasking reentrant and pure, and it can be made resident with the standard shell resident commands – 'resident' under AmigaOS Shell, and 'resi' under WShell. If you use another shell, please refer to the shell's user manual for information about how to make programs resident.

1.118 Multi-volume archives

Multi-volume archives are created simply by splitting a larger archive into smaller files. The evaluation version does not do this automatically on the fly (but the registered version does), so you will need to do it manually unless you register.

The source for two utilities used to create multivolume archives are included in the distribution archive. If you have access to a C compiler you can compile the (hopefully) portable 'splitlzh' and 'joinlzh' programs simply by entering 'make' in the 'unixutil' directory. These two programs are useful when transferring big archives to and from **IX systems.

1.119 Multivolume file names

The first file of a multivolume archive is named 'name.lha' or 'name.lzh'. The following volumes are named 'name.l01', 'name.l02' and so on. This naming convention has been chosen because certain brain-damaged filesystems don't allow long filenames (MSDOS). Multivolume archives spanning more than 100 volumes are not currently supported.

1.120 Making backups with multivolume archives

The multivolume capability of LhA can be used to make efficient harddisk backups. In order to do this you will need some formatted floppy disks (or equivalent) – LhA does not currently format disks while writing. An example backup command would be:

```
LhA -r -v9 -Qh64 -Va -Qvdf0,df2,df3 a df0:Backup920712 lha:#?
```

This would archive all files in the 'LhA:' directory/device to disks starting with drive DF0:, then DF2: and then after using DF3: LhA would repeat the cycle until the backup is finished.

LhA is somewhat slower than using a dedicated backup program since it has to go through the filing system instead of writing directly to the disks. However, LhA offers greater compression than any existing backup program.

1.121   Incremental backups

Incremental backups are backups where you only backup files that has been changed since the last backup. In LhA this can be accomplished like this:

  LhA -V -s -S -Qvdf0,df1 -r a df0:Backup920912 Work:#?

This would archive all files on 'Work:' that does not have its 'a' (archived) bit set (the -s option). After adding a file LhA will set that file's 'a' bit (the -S option). Whenever a file is written to, AmigaDOS automatically clears this bit so it will be included in the next incremental backup.

1.122   Extracting from multivolume archives

When extracting files from multivolume archives, LhA must scan the entire archive from first to last volume. An example command would be:

  LhA -V -Qvdf0,df1 x df0:MltArc #?.c

This would extract all '.c' files (#?.c) from the multivolume archive (-V) 'MltArc.lha', alternating between drive df0: and df1: (-Qvdf0,df1).

1.123   Restoring incremental backups

Incremental backups should be restored starting with the latest backup (i.e. the newest archives should be restored first). An example would be:

 LhA -V -T -Qvdf0,df1 x df0:Backup920909 work:
 LhA -V -T -Qvdf0,df1 x df0:Backup920902 work:
 LhA -V -T -Qvdf0,df1 x df0:Backup920821 work:

The '-T' option must be specified so LhA will not try to overwrite any file that is newer than the one present in the archive (that has already been extracted from a newer archive).

1.124   Listing multivolume archive contents

Multivolume archives are listed like this:

    LhA -V v df0:MyArc

This will list all files in the multivolume archive starting with
file 'MyArc.lha'. At the end of every volume LhA will ask for a new
volume until the end of the archive is reached. Listing of individual
volumes is not supported in the current implementation.

1.125   Updating multivolume archives

In the current version it is not possible to delete, freshen or
update files in a multivolume archive.

1.126   Interrupting multivolume archiving

Don't interrupt multivolume archiving.

Currently, interrupting an archiving operation will cause the archive
to become slightly messed up. All data will be OK but you will not be
able to add any files to the archive since LhA will prompt you for a
non-existent volume at the end of the archive. This is inavoidable with
the current implementation.

1.127   A bit about headers

A 'header' has to be written to the archive for every file in order
for the archiver to know what the files are called, how they were
compressed etc.   The original LhArc had a very primitive header layout
and had no good way of storing any machine-specific info like filenotes
(I created a workaround in LhArcA 0.99, by putting the filenote in the
filename field - LhArc and LZ later adopted this method). In **IX LhArc
V1.02 the authors introduced a new type of header (level 1 header) that
allowed slightly more info to be stored, but the header length was still
limited to 255 bytes.   In LHA 2.13 for MS-DOS a new header type was
introduced - level 2 headers. With this latest header type an arbitrary
amount of information can be stored. LhA can both read and write all
these header types.   To select what type of headers to write, use the
'-H' option. LHA for MSDOS and LHa for **IX creates level-1 headers by
default.   LhA uses level 0 headers by default for compatibility reasons
(LZ and LhArc does not handle level 1 and level 2 headers correctly).
If you want to know what header levels an archive contains, use the 'vv'
command.

1.128 Some tips for archiving efficiently

If you are going to archive a big bunch of similar or small files –
text files for example – you can improve compression performance greatly
by first creating an archive WITHOUT compression (using the '-z'
option), and then   add this file to archive (with compression). As an
example I added a big directory with various sources and some binaries
(total 2480 files, 5102117 bytes). this way with:

LhA -z -r a hd:test msrc:

and then compressed it with

LhA -Z -Qh64 a hd:msrc hd:test.lha

The final   'hd:msrc.lha'   archive ended up being 1545076 bytes. When
compressed the normal way ('LhA -r -Qh64 a hd:msrc msrc:'), the archive
was 2114777 bytes long. Quite a difference..

1.129 Using as little memory as possible

When using the default settings, LhA requires about 300KB to archive,
and 180KB to extract files. To reduce this to a minimum you can reduce
the I/O buffer size to 8K. This will save you about 48K when archiving
and at least 24K when extracting. You can reduce the archiving memory
usage even more by reducing the Huffman buffer size to 4K, but it is not
recommended since compression performance will drop significantly.
Please note that the above figures for memory usage includes stack and
program code.

1.130 Creating fully MS-DOS compatible archives

In order to satisfy MSDOS archivers, you may have to disable a few
Amiga-specific features. Filenotes are not supported under MSDOS and
thus the filenote archiving should be disabled with the '-f' option.
Furthermore you should disable file attribute preservation with the '-a'
option. Autoshow files are not supported by MSDOS LHA V2.13. If you
use header level 1 or 2 you don't have to worry about disabling the file
attribute preservation. LHA V2.13 for MSDOS and LHa 0.04 for **IX
creates level 1 headers by default.

To summarize, use the following options to create archives for use
with MSDOS LHA:

'-UH0a0f'

In order to create archives that are extractible with LhArc the
following options should be used when creating archives:

'-H0 -0'

and for MS-DOS LHarc:

```
'-UH0a0f -0'
```

1.131 Recovering data from corrupt archives

It is never possible to recover all lost data from a corrupt archive, but you can retrieve as much data as possible by using the '-k' option and a small I/O buffer (8K). An example would be:

```
LhA -k -b8 x dl:Corrupt ram:
```

This would extract as much as possible from the corrupt archive to 'ram:'.

1.132 Acknowledgements

(Stefan's original acknowledgements)

Haruyasu Yoshizaki   For releasing the source of the original LHA for MSDOS. The source was used as a reference when writing this program. No actual code was copied from this source, rather LhA was written from scratch for the Amiga.

Haruhiko Okumura   For devising the -lh5- and -lh4- compression algorithms, and for releasing the C source for these to the public domain. These sources were used as a reference when writing the 680x0 assembler versions of the compression code. Some algorithms were replaced with my own faster ones, but the ideas are the same.

Robert K.Jung   For making the feature-packed ARJ for MSDOS, from which several ideas for commands and features for LhA were taken.

Paolo Zibetti   For making the first LhArc-style archiver for the Amiga, which made me interested in file archivers and more advanced data compression techniques.

Roger Nordin   Beta tester extraordinaire

Ron Birk   For digging out the source codes I needed before I gained access to InterNet myself - Thanks!

Martin Olsson   For supplying me with the source for LhA V2.11, which was used as a reference. (I wrote the -lh5- decompression with only the 80x86 source available.. hard work!)

LhArcA users   Big thanks to all of you who registered for LhArcA and LhA even before the programs were finished

                        (LhArcA never was, but those who registered will
                        receive LhI/LhA when it's finished).

LhA users   Big thanks to all who registered so far, and even
                        bigger thanks to those who reported bugs and
                        problems with the previous releases – without you
                        this program would never be what it is now.

    The program was developed using the Lattice C Compiler and Assembler
on a 25MHz Amiga 3000. Great compiler, great computer! Furthermore RCS
and MKID were used to simplify the maintenance and development process
greatly.

            "Infinities of dreams imploding into one ..."

                (Jim Cooper's acknowledgements)

Stefan Boberg   Without whom, I would never have had this
                        program to play with.

    These docs were originally formatted with a version of 'proff' (by
Stefan), but were converted to a .guide file "by hand" by myself.