Iha

COLLABORATORS			
	TITLE : lha		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 8, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

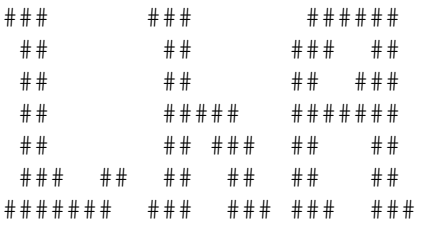
Chapter 1

lha

1.1 index

1.2 LhA Anwenderhandbuch

April 1999



Geschrieben von Stefan Boberg
Momentan verwaltet von Jim Cooper

Jim Cooper <jamie_c@bellsouth.net>
David Tritscher <petrol@rabble.uow.edu.au>

1.3 Einleitung

LhA ist ein mächtiger Archivierer für den Amiga. Er ist voll kompatibel mit LHA für MS-DOS-Systeme sowie LhArc für MS-DOS, Amiga und *NIX. Er ist ebenfalls kompatibel mit LhArcA und LZ für den Amiga. LhA unterstützt schnelle Komprimierung den Dekomprimierung und hat verschiedene Befehle und Optionen, die im keinen anderen momentan für den Amiga verfügbaren Archivierer zu finden sind.

1.4 Über das Handbuch

Das Handbuch ist in zwei große Sektionen unterteilt, die erste Sektion (diese hier) enthält verschiedene Informationen im Bezug auf das Programm. Die zweite Sektion ist ein Nachschlagewerk, in dem alle Kommandos und Optionen eingehend erklärt werden.

1.5 Systemanforderungen

LhA läuft auf jedem Amiga mit mindestens 512KB RAM und einem Diskettenlaufwerk, obwohl 1MB RAM und zwei Diskettenlaufwerke oder eine Festplatte empfohlen werden, um aus LhA das Beste zu machen. LhA läuft auf jedem Kickstart ab Version 2.0.

1.6 Terminologie

ARCHIV - ein Archiv ist eine Datei, die eine oder mehrere Dateien in komprimierten oder nicht komprimierten Zustand enthält, zusammen mit verwandten Informationen wie Dateinamen, Datum/Uhrzeit der letzten Modifizierung, Dateikommentaren usw.

KOMPRESSION - der Vorgang, in dem redundante Informationen in Daten umgewandelt werden, die weniger Speicherplatz erfordern. Es gibt eine Vielzahl von Möglichkeiten, wie dies geschehen kann. LhA benutzt eine modifizierte Form der Lempel-Ziv-Kompression, mit Block-adaptiver Huffman-Kodierung und einem Wörterbuch von bis zu 32768 Zeichen.

KOMPRESSIONSRATE - die Kompressionsrate, die von LhA angegeben wird, ist wie folgt berechnet: Rate = (1 - (Komprimierte Größe) / (Originalgröße)) * 100. D.h. wieviel Prozent der Datei GEWONNEN wurden. Andere Archivierer mögen andere Methoden verwenden. LHA und ARJ für MS-DOS berechnen die Rate z.B. wie folgt: Rate = (Komprimierte Größe) / (Originalgröße), also wie groß die komprimierte Datei im Vergleich zur Originaldatei ist. (MS-DOS-Rate = 1 - (ADOSRate/100)). Je höher die LhA-Kompressionsrate, je besser die Kompression. Die meisten Amiga-Archivierer benutzen dieselbe Berechnungsmethode für die Kompressionsrate.

CRC - CRC steht für 'Cyclic Redundancy Check'. Das ist eine relativ komplexe Methode, um die Integrität von Daten zu prüfen. Die Version, die in LhA verwendet wird, ist ein 16-Bit CRC.

EXTRAKTION oder DEKOMPRESSION - der Vorgang, bei dem exakt die Information wiederhergestellt wird, die zuvor komprimiert wurde (Dateiinhalte, Datum der letzten Modifikation, Dateikommentare, Schutzbits, Verzeichnisstruktur usw.)

SELBST-EXTRAHIERENDES MODUL (SFX-Modul) - dies ist ein Archiv, das eine ausführbare Datei darstellt, die in sich selbst enthaltene Dateien extrahieren kann.

1.7 LhA - was ist das?

Das Hauptziel von LhA ist es, die Amiga-Gemeinschaft mit einem schnellen, effizienten und zuverlässigen Archivierer auszustatten. LhA erstellt und bearbeitet Archiv-Dateien mit dem '.lha'-Suffix, und ist voll kompatibel sowohl mit MS-DOS LhArc und MS-DOS LhA, als auch mit dem *NIX LhA, und äteren LhA/LhArc Archivierern der meisten anderen Plattformen. Es kann sowohl mit der alten LhArc-Kompression (-lh1-, -lh0-), als auch mit der LhA-Kompression (-lh6-, -lh5-, -lh4-) umgehen.

1.8 Was ist überhaupt ein Archivierer?

Ein Archivierer, wie der Name schon sagt, archiviert (Dateien). Er sammelt die von Ihnen angegebenen Dateien und speichert sie alle in einer einzelnen Archivdatei. So gut wie alle Archivierer (inklusive LhA)

komprimieren die Dateien auch noch, bevor sie in die Archiv-Datei gespeichert werden, so daß sie weniger Speicherplatz verbrauchen. Wenn Sie einige Dateien wieder aus dem Archiv holen wollen, dekomprimiert der Archivierer die Datei und stellt ihre Attribute (Datum/Zeit der letzten Modifikation, Dateikommentare, Schutzbits usw.) wieder her. Ein Archivierer kann Archivdateien üblicherweise auch auf verschiedene Arten bearbeiten, z.B. Dateien löschen, aktualisieren, ausdrucken usw. Lesen Sie den Abschnitt 'Archivierer-Kommandos' dieses Handbuchs, um eine Erläuterung der verschiedenen Aktionen zu erhalten, die LhA ausführen kann.

Die häufigste Anwendung für einen Archivierer ist es, verschiedene zusammengehörige Dateien per Modem zu übertragen. Es wäre z.B. eine sehr langweilige und unhandliche Aufgabe, jede einzelne Quelldatei eines großen Projektes einzeln zu übertragen, also warum nicht alles in eine einzige Datei packen? Das ist der Moment, in dem der Archivierer ins Spiel kommt; wir müssen nur den Archivierer mit allen Dateien füttern, die wir übertragen wollen, und dann die einzelne Archivdatei übertragen, die der Archivierer erstellt! Nach der Übertragung muß der Empfänger nur den Archivierer benutzen, um alle Dateien aus der Archivdatei auf seine Festplatte (oder Diskette) zu extrahieren. Dazu kommt, das die Übertragung weniger lange dauert, wenn der Archivierer die Dateien komprimiert hat, womit uns der Telefonanruf weniger kostet. Es gibt natürlich auch andere Anwendungen für einen Archivierer, man kann ihn z.B. als Backup-Programm für Festplatten benutzen (wenn man eine andere Festplatten-Partition hat, auf der man die Archivdatei lagern kann...), und man kann ihn benutzen, um den Kram abzulegen, den man nicht oft benötigt, und wenn man ihn dann braucht, muß man einfach die Datei aus dem Archiv extrahieren, und nach Erledigung wieder löschen (und Speicherplatz sparen). Manche Leute benutzen LhA, um automatische Backups ihres Quellcodes für verschiedene Projekte anzufertigen.

Die Komprimierungsmethoden reichen von sehr einfach, wenig effektiv und schnell (z.B. Run-Length Encoding, RLE) bis hin zu komplex, effektiv und relativ zeitraubend (Lempel-Ziv-Huffman, LZHUF, wie in LhA angewandt). Die Methode, die in LZH-Archivierern verwendet wird (LZHUF), ist wahrscheinlich der beste Algorithmus, der heute in einem Archivierer verwendet wird, Es gibt andere, ähnliche Methoden, wie ZIP, aber diese sind nicht genauso gut. Obwohl die Datei kleiner wird, verliert man keinerlei Informationen durch die Komprimierung, die Information wird nur auf andere Weise gespeichert. Im Prinzip wird redundante (sich wiederholende) Information durch einen Zeiger auf einen anderen Teil der Datei ersetzt, in dem die Information abgelegt ist. Z.B. erscheint in diesem Text das Wort 'Archiv' an verschiedenen Stellen, das ist ein Beispiel für redundante Daten. Einfach ausgedrückt, müßte ein Komprimierer diese Datei komprimieren, würde er das erste Auftauchen von 'Archiv' unverändert lassen, und dann jedes weitere Erscheinen durch einen Zeiger auf das erste Vorkommen ersetzen. Beim Dekomprimieren der Datei benutzt der Archivierer diese Zeiger, um die Datei in ihren Ursprungszustand zurückzuversetzen.

Dateien, die bereits mit einem Verfahren komprimiert wurden, können ganz allgemein nicht weiter komprimiert werden, indem man sie nochmals dem gleichen Komprimierungsprogramm übergibt (wäre dies möglich, wären Modem-Transfers sehr viel billiger :), da die redundante Information bereits entfernt wurde. Es ist allerdings möglich, Dateien, die von bestimmten Komprimierern (z.B. RunLength-Encoder) ausgegeben wurden,

dadurch weiter zu komprimieren, daß man sie an Programme übergibt, das eine andere Methode verwendet (wie LZHUF), da sie andere Arten redundanter Information eliminieren.

1.9 Kompatibilität und Amiga-spezifische Funktionen

LhA zielt auf volle Kompatibilität mit LHA V2.55 für MS-DOS, was eine Verbesserung gegenüber dem originalen LhArc V1.13 darstellt. LhA ist ebenfalls kompatibel mit LhArc, LhArcA und LZ für den Amiga. Allerdings können LhArc und LhArcA nicht mit Archiven umgehen, die Köpfe der Stufe 1 oder 2 benutzen, oder mit den neuen LHA-Kompressionstypen (-lh5- oder -lh6-) komprimiert wurden. LZ 1.02 kann keine Archive mit Köpfen der Stufe 2 oder dem neuesten Kompressionstyp (-lh6-) bearbeiten. LHA V2.55 kann alle Archive bearbeiten, die mit LhA für den Amiga erstellt wurden.

1.10 Über den Autor, die Geschichte und die Zukunft des Programms

(Stefans Original)

Ich, Stefan Boberg - der Autor der Programme der LhA-Familie, bin 19 Jahre alt und studiere momentan 'angewandte Physik und Elektroingenieurswesen' im ersten Jahr am Linköping Institut der Technologie. Ich begann hauptsächlich deshalb an LhA zu arbeiten, weil ich dachte, es gäbe keine wirklich guten Archivierer für den Amiga; diejenigen, die existierten, als ich damals (Juni 1991) begann, waren entweder zu langsam, hatten schwache Kompressionsraten oder waren fehlerbehaftet bzw. nicht voll funktionsfähig, so daß sie nicht leisteten, was ich von einem Archivierer erwartete. Ich benutzte Archivierer hauptsächlich, um automatische Backups der Quellcodes für meine verschiedenen Programmierprojekte zu erstellen, und sehr of auch, um einfach Archive von Bulletin Boards und Computernetzen zu dekomprimieren. Ein weiterer Grund, es zu tun, war, etwas Geld zu verdienen, welches ich dringend brauche, denn ich bin ein armer Student mit _kleinem_ Taschengeld... :)

(Jim Coopers Geschwafel)

Ich, Jim Cooper, habe begonnen, Stefan nach dem LhA-Quellcode zu fragen, seit er vor langer Zeit aufhörte, daran zu arbeiten. Schließlich hat er ihn mir geschickt, wahrscheinlich nur, um mein Generve zu beenden! :-)

Auf jeden Fall habe ich entschieden, dies der Amiga-Gemeinschaft umsonst zur Verfügung zu stellen. Registrierungen sind nicht mehr notwendig. Dies ist die _vollständige_ Version von LhA, äquivalent zu der früheren "registrierten" Version. Vielleicht werde ich, zu einem späteren Zeitpunkt, auch den Quellcode zur Verfügung stellen, oder auch nicht. Noch nicht entschieden.

Nach v1.98 nötigte ich David Tritscher dazu, mir mit dem Code zu helfen, da er _sehr_ gut bei den Low-Level Kompressionsroutinen etc. ist.

Wenn Sie Fehler finden, lassen Sie es mich wissen, und ich werde

versuchen, sie zu beheben. Vielleicht nicht sofort, da ich einen "Tagesjob" zu erledigen habe, aber ich werde versuchen, es so gut wie möglich am Laufen zu halten.

Viel Spaß.

1.11 Nachschlagewerk

Dieser Abschnitt des Handbuchs soll hauptsächlich als Nachschlagewerk dienen, wenn Sie genau wissen wollen, wie eine bestimmte Option oder ein Kommando arbeitet. Wenn Sie LhA bisher noch nie benutzt haben (aber Erfahrungen mit anderen Archivierern haben), sollten Sie zumindest die Beschreibungen aller Kommandos und Optionen überfliegen, um eine Idee davon zu erhalten, was LhA leisten kann.

1.12 Kommandozeilen-Syntax

Die Kommandozeilen-Syntax lautet wie folgt:

LhA [-Optionen] <Kommando> <Archiv> [[AusgangsVerz] DateiDef] [@Datei] [ZielVerz]

Die Punkte in eckigen Klammern sind optional, die Punkte in spitzen Klammern vorgeschrieben. Lesen Sie die folgenden Abschnitte für exakte

Informationen zu den einzelnen Punkten.

1.13 Definition von Optionen

Anders als andere Archivierer erlaubt es LhA, Optionen überall in der Kommandozeile zu definieren. Die Einleitung einer Option ist '-' (Bindestrich), und alle Punkte der Kommandozeile, die mit diesem Zeichen beginnen, werden als Optionsschalter interpretiert. Wenn Sie einen Dateinamen oder etwas anderes definieren wollen, das mit einem '-'-Zeichen beginnt, schließen Sie den Namen in doppelte Anführungsstriche ein, oder benutzen Sie doppelte Bindestriche. Um z.B. einen Dateinamen '-minus' zu definieren, können Sie entweder '"-minus"' oder '--minus' angeben.

Wenn Sie '-o' schreiben, wird die Option 'o' unabhängig ihres ursprünglichen Zustandes aktiviert. Wenn Sie eine Option abschalten wollen, hängen Sie eine '0' (Null) an die Option an, wie in '-o0'. Wenn eine Option von einem anderen numerischen Zeichen als '0' gefolgt wird, wird die Option aktiviert.

Sie können mehrere Optionen definieren, ohne einen Bindestrich vor jeden Optionsbuchstaben setzen zu müssen. Ein Beispiel wäre '-ox0m', was die Option 'o' aktivieren, die Option 'x' deaktivieren, und die Option 'm' aktivieren würde. Die einzige Ausnahme sind Optionen, die numerische Argumente mit mehreren Stellen akzeptieren; diese müssen von einem Leerzeichen und einem weiteren Bindestrich gefolgt werden, wenn Sie mehr Optionen angeben möchten (wie in '-b32 -ox0m').

1.14 Definition von Kommandos

Das erste Argument der Kommandozeile, das keine Option darstellt, MUSS die Kommandodefinition sein. Kommandos sind nicht größensensitiv (‘l’ bedeutet das Gleiche wie ‘L’), und nur das erste Zeichen des Argumentes wird ausgewertet (mit Ausnahme des ‘vv’-Kommandos), so daß Sie ausgeschriebene Kommandos wie ‘list’ oder ‘add’ statt ‘l’ oder ‘a’ verwenden können.

1.15 Definition von Archiven

Die Definition des Archivs muß das zweite Argument sein, daß keine Option darstellt (das erste ist die Kommandodefinition). In den meisten Fällen können Sie hier ein Namensmuster angeben. Die Ausnahme ist das Kommando ‘m’ (Dateien in ein Archiv verschieben).

1.16 Dateidefinition

Dateidefinitionen werden nach der Archivdefinition angegeben. Die Dateidefinition darf Namensmuster enthalten. Beachten Sie, das Dateidefinitionen (wie überall in LhA) auch für Verzeichnisnamen Namensmuster enthalten dürfen - z.B. ist ‘hd:*/*/dir/*.h’ gültig.

ANMERKUNG

Wenn Sie keinerlei Dateidefinition angeben, geht LhA davon aus, daß Sie mit allen Dateien in dem Archiv oder im Verzeichnis arbeiten wollen.

1.17 Ausgangsverzeichnisse

Ausgangsverzeichnisse sind ein neues Konzept, das mit LhA eingeführt wird; es bietet einen einfachen Weg, anzugeben, welcher Teil des Pfadnamens im Archiv bewahrt werden soll. Es kann auch dazu benutzt werden, die Definition verschiedener Dateien innerhalb des gleichen Verzeichnisses zu vereinfachen. Es wird vielleicht am Besten anhand einiger Beispiele erklärt:

BEISPIELE

Beispiel 1:

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

Dies würde die folgenden Dateien zu ‘newarc.lha’ hinzufügen:

hinzugefügte Datei(en)	Gespeichert als
dh0:files/file1	file1
dh0:files/dir1/file2	dir1/file2
dh0:files/dir2/file3	dir2/file3

```
   dh0:files2/*.c           *.c
```

Beispiel 2:

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

Dies würde alle '.c' und '.h' Dateien in 'hd:tmp' und dessen Unterverzeichnissen hinzufügen, wobei die Pfadnamen, aber ohne den 'hd:tmp'-Teil, mit gespeichert würden. Z.B. würde die Datei 'hd:tmp/src/foo/arargh.c' unter dem Namen 'src/foo/arargh.c' dem Archiv hinzugefügt. Zusätzlich würden alle '.s', '.snd' und '.iff'-Dateien in 'hd:px' und dessen Unterverzeichnissen hinzugefügt, wobei der 'hd:px/'-Teil des Pfades ausgelassen würde.

Ausgangsverzeichnis-Definitionen müssen mit einem '/' oder ':' enden, sonst werden sie nicht als solche erkannt.

Ausgangsverzeichnis-Definitionen dürfen Namensmuster enthalten.

ANMERKUNG

Sie sollten den Namen des Ausgangsverzeichnis nicht in der Dateidefinition hinter dem Ausgangsverzeichnis wiederholen. D.h. Sie sollten nicht 'devs:printers/ devs:printers/*HP*' eingeben; 'devs:printers/ *HP*' wäre korrekt.

Das Ausgangsverzeichnis bleibt für den Rest der Kommandozeile aktiv, oder bis zur nächsten Definition eines Ausgangsverzeichnisses. Wenn Sie das Ausgangsverzeichnis auf das aktuelle Verzeichnis setzen wollen (wie es zu Anfang war), benutzen Sie einen einzelnen Querstrich ('/') als Ausgangsverzeichnis-Definition. Das bedeutet, daß Sie nicht einen einfachen Querstrich verwenden können, um das Mutterverzeichnis zu definieren; um dies zu erreichen, müssen Sie einen zusätzlichen Querstrich angeben ('//' bedeutet Mutterverzeichnis, '///' das Verzeichnis zwei Ebenen höher und so weiter).

1.18 Rekursives Suchen

Wenn Sie Dateien rekursiv suchen (durch Benutzung der Option -r mit dem Kommando a oder u), werden Dateidefinitionen etwas anders behandelt. Ausgangsverzeichnisse funktionieren wie üblich. Im rekursiven Suchmodus wird der letzte Teil der Dateidefinition (also der Dateinamen-Teil) als Muster benutzt, das mit allen Dateien im definierten Verzeichnis und allen Unterverzeichnissen verglichen wird. Ein paar Beispiele werden diese etwas schwammige Beschreibung hoffentlich erläutern:

BEISPIELE

Beispiel 1:

```
lha -r a myarc *
```

wird alle Dateien im aktuellen Verzeichnis und dessen Unterverzeichnissen zu ‘myarc.lha’ hinzufügen.

Beispiel 2:

lha -r a myarc *.c *.cpp

wird alle ‘.c’ und ‘.cpp’-Dateien im aktuellen Verzeichnis und dessen Unterverzeichnissen zu ‘myarc.lha’ hinzufügen.

Beispiel 3:

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

wird alle Dateien in ‘ram:work’ und dessen Unterverzeichnissen, als auch alle ‘.c’-Dateien in ‘hd:tmp’ und dessen Unterverzeichnissen, zu ‘myarc.lha’ hinzufügen. Die vollen Pfadnamen werden erhalten (natürlich exklusive der Laufwerksspezifikation).

Beispiel 4:

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

wird exakt dasselbe tun wie Beispiel 3, aber LhA wird den ‘ram:work/’ und ‘hd:tmp/’-Teil der Dateinamen nicht im Archiv speichern (aufgrund der Definition des Ausgangsverzeichnisses).

Beispiel 5:

lha -r a myarc ram:dir1 ram:makefile

wird alle Dateien im Verzeichnis ‘dir1’ und dessen Unterverzeichnissen, als auch die Datei ‘ram:makefile’, archivieren.

Beispiel 6:

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

wird fast dasselbe tun wie Beispiel 5, aber wird ALLE ‘makefile’s in ram: und dessen Unterverzeichnissen archivieren (aufgrund der Klammern - siehe Anmerkung unten).

ANMERKUNG

Explizit angegebene Ausgangsverzeichnisse (explizit = ohne Namensmuster) werden behandelt wie ‘Verzeichnis/*’, d.h. alle Dateien in dem Verzeichnis und dessen Unterverzeichnissen werden archiviert. Explizit angegebene Dateien werden nur im aktuellen Ausgangsverzeichnis gesucht, solange der Dateiname nicht in Klammern eingeschlossen ist. In diesem Fall wird die Datei rekursiv gesucht. Ich habe mich entschlossen, es so zu implementieren, weil LhA dadurch besser zusammen mit Verzeichnistools wie Browser oder DirectoryOpus verwendet werden kann.

1.19 Definition des Zielverzeichnisses

Sie können ein (optionales) Zielverzeichnis für die Dateien angeben, die durch die Extraktionsbefehle geschrieben werden, indem Sie den gewünschten Verzeichnisnamen irgendwo in der Kommandozeile hinter dem Archivnamen angeben. Wenn kein Zielverzeichnis definiert wurde, wird LhA das aktuelle Verzeichnis als Ziel verwenden. Die Angabe des Zielverzeichnisses muß mit ':' oder '/' enden (genau wie Ausgangsverzeichnisse), oder LhA wird nicht in der Lage sein, Verzeichnisangaben von Dateinamen zu unterscheiden.

BEISPIEL

'lha x corpus ram:' würde den Inhalt von 'corpus.lzh' nach ram: entpacken.

'lha x project *.c dl:tmp/' würde den Inhalt von 'project.lzh' in das Verzeichnis 'dl:tmp' entpacken.

'lha x project dl:tmp/ *.c' würde das Gleiche tun.

ANMERKUNG

Sie können ein nicht existierendes Verzeichnis als Ziel angeben. LhA wird das Verzeichnis (ohne Sicherheitsabfrage) automatisch für Sie anlegen.

1.20 `\@'-Dateien

'@'-Dateien sind Dateien, die so behandelt werden, als ob ihr Inhalt in der Kommandozeile stehen würden. Sie können benutzt werden, um Dateien zu definieren, Optionen, Kommandos, und überhaupt alles, was in der Kommandozeile eingegeben werden kann. Ein Beispiel wäre das Kommando 'lha -r e arc.lzh *.[chas] @filelist ram:', das alle Dateien, die dem Muster '*.[chas]' entsprechen, und die Dateien, die in 'filelist' aufgezählt werden, nach ram: zu entpacken. Wagenrücklaufcode und Zeilenvorschub in '@'-Dateien werden wie Leerzeichen behandelt.

1.21 Grenzen von LhA

LhA wurde geschrieben, um so flexibel wie möglich zu sein, aber es gibt ein paar Einschränkungen, die Ihnen als User bewußt sein sollten.

- LhA Pfadnamen sind momentan auf 255 Zeichen beschränkt. Wenn Sie diese Grenze überschreiten, ist das Verhalten nicht definiert. Berichte von Usern deuten darauf hin, daß AmigaDOS Pfadnamen mit mehr als 180-190 Zeichen nicht korrekt handhabt.

- Wenn Archivköpfe der Stufe 0 verwendet werden, dürfen Dateikommentare nicht länger sein als 230-{Länge Dateiname (inklusive Pfad)} Zeichen. Mit Archivköpfen der Stufe 1 oder 2 dürfen Dateikommentare bis zu 255 Zeichen lang sein (AMigaDOS unterstützt momentan nur Dateikommentare

von maximal 80 Zeichen, so daß es keine Probleme geben sollte, außer mit außergewöhnlich langen Datei- und Pfadnamen).

- Die Anzahl von Dateien in einer Archivdatei ist nur durch den verfügbaren Plattenspeicher begrenzt. Die Größe eines Archivs darf 2.147.483.648 Bytes (2 Gigabytes) nicht überschreiten; LhA wird sonst SEHR verwirrt.

- Die Anzahl von Argumenten in der Kommandozeile ist nur durch das verfügbare RAM und die benutzte Shell beschränkt.

- Die erlaubte Anzahl von durch Namensmuster definierter Dateien ist nur durch das verfügbare RAM beschränkt. Jede Anzahl von Dateien kann in einem Durchgang aus einem Archiv extrahiert oder hinzugefügt werden.

- Archivköpfe der Stufe 2 dürfen nicht länger als 1024 Zeichen sein, oder LhA kann sie nicht verarbeiten.

- Im Moment verwaltet LhA Mehr-Medien-Archive bis zu einem Maximum von 100 Medien. Wenn Sie Archive mit mehr als dieser Anzahl von Medien erstellen, ist das Verhalten undefiniert.

1.22 Veraltete Optionen

Diese Optionen werden zur Kompatibilität mit älteren Skripts etc ↩
. von
LhA immer noch akzeptiert, aber ignoriert.

- ‘-1’ Benutze LhA-Kompression (-lh4-)
- ‘-g’ Verschlüssele mit Passwort
- ‘-U’ Setze Update-Intervall
- ‘-v’ Setze Kompressionsgeschwindigkeit

-lh4- Kompression wird nicht mehr länger unterstützt. -lh4- Extraktion ist, und wird wahrscheinlich für immer unterstützt, zur Kompatibilität mit älteren Archiven.

Wenn Sie ein Archiv zu einem System übertragen, das -lh5- oder -lh6- nicht unterstützt, benutzen Sie die
-0
Option.

Zu der Option ‘-g’... sie war niemals implementiert oder dokumentiert. Wenn Sie den Zugriff auf Ihr Archiv einschränken wollen, empfehlen wir PGP.

1.23 Umgebungsvariablen

LhA unterstützt sowohl lokale wie globale Umgebungsvariablen. Beim Start sucht LhA nach der Umgebungsvariable ‘LHAOPTS’, und fügt sie ein, als ob sie in der Kommandozeile direkt nach dem ‘LhA’-Kommando eingegeben

wurde. Wenn Sie die Einstellungen aus der Umgebungsvariable nicht verwenden wollen, benutzen Sie den ‘-I’-Schalter.

BEISPIEL

Wenn Sie LHAOPTS mit dem folgenden Kommando auf ‘-N -b64’ setzen:

```
1> setenv LHAOPTS -N -b64
```

wird LhA keine Datei-Fortschrittanzeige anzeigen und einen 64K I/O-Puffer für alle folgenden Aufrufe verwenden, bis ein Reset erfolgt oder LHAOPTS geändert wird. Wenn Sie einige Vorgabeeinstellungen setzen wollen, die einen Reset überleben, benutzen Sie den Namen ‘ENVARC:LHAOPTS’ für die Umgebungsvariable, z.B.:

```
1> setenv ENVARC:LHAOPTS -b64
```

Dies setzt die Umgebungsvariable LHAOPTS bei jedem Reboot auf ‘-b64’.

1.24 Namensmustererkennung

Dieser Abschnitt beschreibt, wie LhA Namensmustererkennung und das Sammeln von Dateien vornimmt. Bitte wenden Sie sich an Sektion 2.1 (Kommandozeilensyntax) für eine Beschreibung, welche Kommandos Namensmuster anerkennen.

Namensmuster sind in LhA immer unabhängig von Groß-/Kleinschreibung. (D.h. es macht keinen Unterschied, ob man Namen groß oder klein schreibt, ‘a’ entspricht sowohl ‘a’ als auch ‘A’.)

2.3.0 Was genau sind eigentlich Namensmuster?

2.3.1 Erkannte Musterzeichen

2.3.2 Nationale Zeichen

1.25 Was genau sind eigentlich Namensmuster?

Namensmuster sind ein Weg, verschiedene Dateien auf elegante und relativ einfache Weise anzugeben. Statt einfach alle Dateinamen nacheinander in der Kommandozeile anzugeben, mit denen man arbeiten möchte (was sehr aufwendig sein kann, wenn viele Dateien betroffen sind), können Sie eine Technik einsetzen, die ‘Namensmustererkennung’ genannt wird. Mit dieser Technik können Sie - wie der Name schon sagt - die Tatsache ausnutzen, daß die Namen der Dateien, mit denen Sie arbeiten wollen, oft bestimmte Details gemeinsam haben. Zum Beispiel enden Dateien, die C-Quellcode enthalten, fast immer auf ‘.c’. Wenn Sie nun alle C-Quellcode-Dateien im aktuellen Verzeichnis archivieren wollen, können Sie diese Eigenschaft nutzen, indem Sie ein Muster eingeben, das auf diese Dateien zutrifft

(in diesem Fall wäre das '*.c'). Wie genau diese Muster aufgebaut sind, wird ab Abschnitt 2.3.1 erklärt. Lesen Sie auch die Abschnitte über 'Dateidefinition' und 'Definition von Archiven'.

1.26 Erkannte Musterzeichen

LhA erkennt alle gültigen KickStart 2.x+ Musterzeichen.

In der folgenden Beschreibung meint das Wort 'Ausdruck' entweder ein einzelnes Zeichen oder Buchstaben (wie 'x' oder '?'), oder eine Auswahl (wie '(ab|cd|ef)'), oder eine Zeichenklasse (wie '[a-z,A-Z]').

2.3.1.1 Fragezeichen (?)

2.3.1.2 Stern (*)

2.3.1.3 Doppelkreuz (#)

2.3.1.4 Eckige Klammern ([])

2.3.1.5 Klammern und der senkrechte Strich

2.3.1.6 Tilde (~)

2.3.1.7 Prozentzeichen (%)

2.3.1.8 Apostroph (')

1.27 Fragezeichen (?)

Das Fragezeichen entspricht einem beliebigen _einzelnen_ Zeichen. Das Fragezeichen wird manchmal auch als "Platzhalter" bezeichnet.

BEISPIEL

'd?' : entspricht allen Namen mit zwei Buchstaben, die mit einem 'd' beginnen. Zum Beispiel 'dm' oder 'd8'.

'ab?d' : entspricht allen Namen mit vier Buchstaben, die mit 'ab' beginnen und mit 'd' enden. Zum Beispiel 'abcd', 'abad' und 'ab_d', aber nicht 'abd' oder 'acid'.

'f??' : entspricht allen Namen mit drei Buchstaben, die mit 'f' beginnen. Zum Beispiel 'foo', 'fel', 'fan' aber nicht 'ab', 'fuga' oder 'fini'.

1.28 Stern (*)

Der Stern entspricht jeder Zeichenfolge beliebiger Länge, inklusive der Länge Null (d.h. dem Leerstring). Das '*'-Zeichen wird oft 'Jokerzeichen' genannt.

BEISPIEL

'a*' : entspricht allen Namen, die mit einem 'a' beginnen. Zum Beispiel 'abba', 'anette'.

'a*z' : entspricht 'auugaz', 'awacz' und 'az' sowie allen Namen die mit 'a' beginnen und mit 'z' enden.

's*f*n' : entspricht 'stefan', 'staffan', 'steffen', 'sfn' oder jedem anderen Namen, der mit 's' beginnt, gefolgt von einer beliebigen Zahl (inklusive Null) von beliebigen Zeichen, gefolgt von 'f', und mit 'n' endet.

'*.lzh' : Enspricht allen mit '.lzh' endenden Namen.

1.29 Doppelkreuz (#)

Das Doppelkreuz entspricht einem folgenden Ausdruck (Muster) 0 oder mehrmals. Das einfachste Beispiel hierfür ist '#?', was jedem beliebigen Namen entspricht (äquivalent zum '*'-Zeichen).

BEISPIEL

'#a' : entspricht jedem Namen, der nur aus dem Buchstaben 'a' besteht. Zum Beispiel 'aaaa' und 'a'.

'b#ad' : entspricht jedem Namen, der mit 'b' beginnt, gefolgt von einer beliebigen Zahl von 'a's (inklusive Null), und mit 'd' endet. Zum Beispiel 'bad', 'bd' und 'baaaad'.

'#(ha)#(hi)urgh' : entspricht einer beliebigen Anzahl von 'ha's gefolgt von einer Anzahl von 'hi's gefolgt von 'urgh'. Zum Beispiel 'hahahahahihiurgh', 'haurgh' und 'hahiurgh'.

1.30 Eckige Klammern ([])

Die eckigen Klammern umschließen eine Auswahl zutreffender Zeichen. Sie sind ein wenig wie Klammern, aber entsprechen nur einzelnen Zeichen. Sie können entweder nur die Zeichen eingeben, die dem Ausdruck entsprechen sollen, wie in '[abcx]' (dies würde 'a', 'b', 'c' und 'x' entsprechen), oder Sie können Bereiche angeben, wie in '[a-c,x-z]' (dies würde 'a', 'b', 'c' sowie 'x', 'y' und 'z' entsprechen).

BEISPIEL

'prg.[1-9]' : entspricht jedem Namen mit fünf Buchstaben, der

mit ‘prg.’ beginnt, gefolgt von einer Ziffer außer ‘0’. Zum Beispiel ‘prg.1’, ‘Prg.8’.

‘Ver_[1-2].[0-9].[a-z]’ : entspricht jedem Namen mit neun Buchstaben beginnend mit ‘ver_’, gefolgt von entweder einer ‘1’ oder einer ‘2’, gefolgt von einem Punkt (‘.’), einer Ziffer und schließlich einem Zeichen zwischen ‘a’ und ‘z’ (d.h. allen Zeichen im Alphabet). Zum Beispiel ‘Ver_1.2.a’, ‘Ver_2.9.d’.

‘#[a-z,0-9]’ : entspricht jedem Namen, der eine beliebige Zahl von alphanumerischen Zeichen enthält (d.h. entweder aus dem Alphabet, oder Ziffern). Zum Beispiel ‘ados’ oder ‘PDP11’. Es entspricht jedoch nicht ‘AXE.dat’, da dies einen ‘.’ enthält, der nicht im Zeichenbereich angegeben ist.

‘*.[chas]’ : entspricht jedem Namen, der mit ‘.c’, ‘.h’, ‘.a’ oder ‘.s’ endet.

1.31 Klammern und der senkrechte Strich

Klammern können zu verschiedenen Zwecken eingesetzt werden. Der erste Verwendungszweck ist der Gleiche wie in der Mathematik - um verschiedene einzelne Ausdrücke zu einem einzigen Ausdruck zu gruppieren. Der andere Zweck ist, eine Liste zu akzeptierender Ausdrücke, getrennt durch ‘|’, zu erstellen. Der gesammte geklammerte Ausdruck wird von anderen Musterzeichen (wie ‘#’ und ‘~’) als ein Ausdruck gewertet. Diese beiden Zwecke sind eigentlich gleich, da der erste Fall nur ein Sonderfall des zweiten ist. Dies ist mit ein paar Beispielen einfacher erklärt:

BEISPIEL

‘(abc|def|xyz)’ : entspricht den Namen ‘abc’, ‘def’ und ‘xyz’, und keinem anderen.

‘*.(doc|prf|man)’ : entspricht allen Namen, die mit ‘.doc’, ‘.prf’ oder ‘.man’ enden.

‘~(pfile)’ : entspricht allen Namen außer ‘pfile’. (NB: dieser Ausdruck ist NICHT das Gleiche wie ‘~pfile’, siehe Abschnitt 2.3.1.6 für Details.)

‘(*.c|*.h|*.doc|ab*)’ : entspricht allen Dateien, die mit ‘.c’, ‘.h’ oder ‘.doc’ enden, sowie allen, die mit ‘ab’ beginnen.

Eine Dateidefinition ‘(xxx|yyy|zzz)’ zu schreiben hat die gleiche Funktion wie ‘xxx yyy zzz’ (xxx, yyy, zzz können beliebige gültige Ausdrücke sein, inklusive Namensmuster mit Klammern).

Klammern können geschachtelt werden.

1.32 Tilde (~)

Die Tilde negiert den unmittelbar folgenden Ausdruck. Sie negiert NUR das unmittelbar folgende Zeichen oder geklammerten Ausdruck, nicht die gesamte folgende Zeichenkette, wie manche Leute annehmen.

BEISPIEL

'~x?' : entspricht jedem Namen mit zwei Buchstaben, außer den mit 'x' beginnenden. Zum Beispiel 'ah', 'ko' oder 'ba', aber nicht 'x0' oder 'xi'.

'~(x?)' : entspricht allem außer Namen mit zwei Buchstaben, die mit 'x' beginnen. Zum Beispiel 'xaa' oder 'ab', aber nicht 'xa' oder 'x9'.

'~(#?)' : entspricht gar nichts. (Die Tilde negiert das '#?', welches allen Namen entspricht.)

'~lha' : entspricht allen Namen, die nicht mit 'l' beginnen und mit 'ha' enden. Zum Beispiel 'uha', 'why_lha' aber nicht 'lumbha' oder 'lha'.

1.33 Prozentzeichen (%)

Das Prozentzeichen entspricht dem Leerstring. D.h. es entspricht immer 0 Zeichen. Es ist nur ein geklammerten Ausdrücken sinnvoll, und darf nicht dem '#'-Zeichen folgen ('#%' wäre ziemlich sinnlos, da % immer 0 Zeichen entspricht).

BEISPIEL

'lha(.doc|.man|%)' : entspricht 'lha.doc', 'lha.man' und 'lha'.

'l%u%a' : entspricht nur 'lua'; die Prozentzeichen sind hier völlig bedeutungslos und könnten genausogut weggelassen werden.

1.34 Apostroph (')

Das Apostroph hebt den Effekt des folgenden Musterzeichens auf. Das ist nützlich, wenn Ihr Dateiname Zeichen wie ? usw. enthält. Es wirkt bei den folgenden Zeichen: ?*#[]()~%'

BEISPIEL

'hello'?' : enspricht nur dem Dateinamen 'hello?'.

'lha'**' : entspricht jedem mit 'lha*' beginnendem Dateinamen.

'a''b''c' : entspricht dem Dateinamen 'a'b'c'. Zwei Apostrophs zusammen werden wie ein einzelnes behandelt.

1.35 Nationale Zeichen

LhA wandelt nationale Zeichen korrekt entsprechend der aktuellen 'Locale'-Einstellung in Klein- oder Großschrift um, indem es Funktionen der locale.library aufruft.

Siehe auch den Abschnitt 2.5.56 "'-Qn' Nationalen Zeichenmodus aktivieren".

1.36 Kommandos

Dieser Abschnitt beschreibt die Kommandos, die LhA zur ↩
Manipulation und
Pflege von Archiven anbietet. Lesen Sie Abschnitt 2.1.2 (Definition von Kommandos) für Details darüber, wie man Kommandos in der Kommandozeile angibt.

2.4.1 'a' Dateien zum Archiv hinzufügen

2.4.2 'c' Archive zusammenfügen/anhängen

2.4.3 'd' Dateien aus dem Archiv löschen

2.4.4 'e' Dateien aus dem Archiv extrahieren

2.4.5 'f' Dateien im Archiv auffrischen

2.4.6 'h' Unterschiede zwischen Archiv <-> Filesystem finden

2.4.7 'l' Archivinhalt (knapp) auflisten

2.4.8 'm' Dateien ins Archiv verschieben

2.4.9 'p' Dateien nach stdout ausgeben

2.4.10 'r' Dateien ersetzen

2.4.11 't' Archivintegrität testen

2.4.12 'u' Archiv aktualisieren

2.4.13 'v' Archivinhalt (ausführlich) auflisten

2.4.14 'vv' Archivinhalt (vollständig) auflisten

2.4.15 'x' Dateien inklusive Pfad extrahieren

2.4.16 'y' Archiv mit neuen Optionen kopieren

1.37 `a' Dateien zum Archiv hinzufügen

Dieses Kommando fügt eine Anzahl von Dateien einer Anzahl von Archiven hinzu. Wenn das angegebene Archiv nicht bereits existiert, wird es angelegt. Sie können Dateien einem Archiv nicht hinzufügen, wenn sie im Archiv bereits existieren. Wenn Sie es trotzdem versuchen, wird eine Warnung ausgegeben, aber LhA wird fortfahren, die übrigen angegebenen Dateien hinzuzufügen.

Nach Vorgabe wird nur der Dateiname abgespeichert. Wenn Sie Verzeichnisnamen und -struktur bewahren wollen, müssen Sie die -x Option angeben, um die Pfaderhaltung einzuschalten. Wenn Sie vollständige Unterverzeichnisse rekursiv archivieren wollen, können Sie die -r Option verwenden, welche die -x Option automatisch aktiviert. Diese Optionen werden in Sektion 2.5 erklärt.

BEISPIEL

'LhA a myarchive dict.txt' wird die Datei 'dict.txt' dem Archiv 'myarchive.lha' hinzufügen.

'LhA a arc.lzh *.c *.h' wird alle Dateien des aktuellen Verzeichnisses, die mit '.c' oder '.h' enden, dem Archiv 'arc.lzh' hinzufügen.

'LhA -r -0 arch *.c' wird alle '.c'-Dateien im aktuellen Verzeichnis und dessen Unterverzeichnissen dem Archiv 'arch.lzh' hinzufügen, wobei die -lh1- (LhArc 1.x) Kompressionsmethode verwendet wird.

'LhA -r archive src:(lharca|lha)/*.[cha] asrc:*.asm' wird alle '.c', '.h' und '.a'-Dateien in den Verzeichnissen 'src:lharca', 'src:lha' und deren Unterverzeichnissen, sowie alle '.asm'-Dateien im Verzeichnis 'asrc:' dem Archiv 'archive.lha' hinzufügen.

1.38 `c' Archive zusammenfügen/anhängen

Mit diesem Kommando ist es möglich, verschiedene Archive zu einem zusammenzufügen, oder verschiedene Archive aneinander anzuhängen. Momentan prüft LhA nicht auf doppelt vorkommende Dateien. Wenn zwei Archive eine Datei mit dem gleichen Namen enthalten, werden somit zwei Einträge mit dem gleichen Namen im entstehenden Archiv enthalten sein.

Zusammenfügen und Anhängen funktioniert genauso, als ob man alle Dateien aus den Archiven entpacken und sie dann ins Zielarchiv verschieben würde -außer, das dabei keine Dekompression/Kompression vorgenommen wird.

Um verschiedene Archive zu kombinieren (zusammenzufügen), definieren Sie ein nicht existierendes oder leeres Archiv als das Zielarchiv - diese Datei wird dann das entstehende Archiv enthalten.

Um Archive an ein anderes anzuhängen, geben Sie das Archiv, an das angehängt werden soll, als Zielarchiv an - die übrigen Archive werden dann an dieses Archiv angehängt.

BEISPIEL

'LhA c ram:new arc:csrc arc:csrc2' wird die zwei Archive 'arc:csrc.lha' und 'arc:csrc2.lzh' in ein Archiv mit dem Namen 'ram:new.lha' zusammenfügen.

'LhA c arc:csrc arc:csrc2' hätte das Gleiche Ergebnis wie das obige Kommando, doch das resultierende Archiv ist stattdessen 'arc:csrc.lha' ('csrc2.lzh' ist an das Ende von 'arc:csrc.lha' angefügt worden).

Beachten Sie, daß Sie Namensmuster angeben können, um die Dateien zu spezifizieren, die zusammengefügt / angehängt werden sollen.

1.39 `d' Dateien aus dem Archiv löschen

Dieses Kommando entfernt eine oder mehrere Dateien aus einem Archiv. Bitte beachten Sie, daß die Nachricht 'packing' nicht bedeutet, daß LhA die Dateien nochmals komprimiert, sondern daß es die nicht benutzen Dateien aus dem Archiv entfernt (und die verbliebenen Dateien enger packt).

ANMERKUNG

Dateien, die mit dem 'd'-Kommando gelöscht wurden, können nicht wiederhergestellt werden. Wenn eine Datei aus der Archivdatei gelöscht wurde, ist sie für immer verloren.

1.40 `e' Dateien aus dem Archiv extrahieren

Dieses Kommando wird benutzt, um Dateien aus einem Archiv zu extrahieren. Es funktioniert genau wie das 'x' Kommando, nur daß es die '-x' Option beachtet (das 'x' Kommando betrachtet sie als gesetzt). Wenn die '-x' Option abgeschaltet ist, werden Dateien ohne ihre Pfadangaben entpackt, und wenn es eingeschaltet ist, wird LhA alle Dateien mit ihrem Pfad entpacken und die benötigten Verzeichnisse erstellen, wenn sie noch nicht existieren.

BEISPIEL

'lha -x0 e foo.lzh ram:' wird alle Dateien aus dem Archiv 'foo.lzh' nach ram: entpacken, ohne Pfadangaben (alle Dateien werden ins Wurzelverzeichnis entpackt).

'lha x foo.lzh *.c ram:' wird alle auf '.c' endenden Dateien nach ram: entpacken, inklusive Pfad - d.h. die originale Verzeichnisstruktur wird wieder hergestellt.

Weitere Beispiele finden sich im Tutorial.

1.41 `f' Dateien im Archiv auffrischen

Dieses Kommando wird benutzt, um Dateien in einem Archiv aufzufrischen; d.h. ältere Dateien im Archiv durch neue Dateien im aktuellen Verzeichnis zu ersetzen. Pfadnamen werden berücksichtigt, solange die '-x' Option explizit abgeschaltet wurde (mit '-x0'). Dieses Kommando fügt niemals Dateien zu einem Archiv hinzu, es ersetzt nur die Dateien, die ein älteres Modifikationsdatum haben als die entsprechenden Dateien im aktuellen Verzeichnis.

BEISPIEL

'lha f /aab/lha' frischt alle Dateien im Archiv '/aab/lha.lha' auf.

'lha f /aab/fsys *.[ch]' frischt alle '.c' und '.h'-Dateien im Archiv '/aab/fsys.lha' auf.

Dieses Kommando schaltet automatisch die '-x' Option ein, wenn dieses nicht explizit durch ein '-x0' auf der Kommandozeile ausgeschaltet wurde.

1.42 `h' Unterschiede zwischen Archiv <-> Filesystem finden

Dieses Kommando wird benutzt, um zu sehen, welche Dateien in einem Archiv seit der Archivierung verändert wurden. Die '-D' Option (Anzeigentyp) hat bei diesem Kommando eine spezielle Bedeutung; das Listing-Format ist wie folgt:

'-D0' (Vorgabe) : Jede abweichende Datei wird mit dem Namen zur Rechten und einer 'Checkliste' zur Linken aufgeführt, mit Xen an den entsprechenden Stellen, die angeben, welche Unterschiede zwischen Archiv und Filesystem bestehen. 'Tm' bedeutet das Dateidatum unterscheidet sich, 'Sz' bedeutet die Größe ist unterschiedlich, 'Pr' bedeutet die Schutzbits wurden verändert, 'Fn' bedeutet der Dateikommentar wurde verändert, und 'Del' bedeutet, die Datei exisiert nicht mehr.

'-D1' : Jede abweichende Datei wird mit dem Namen zur Linken aufgeführt, gefolgt von einer kurzen Aufzählung der Unterschiede. Wenn mehr als ein Detail sich unterscheidet, wird für jeden Punkt eine neue Zeile ausgegeben.

'-D2' : Wie '-D1', aber alle unterschiedlichen Punkte werden in derselben Zeile ausgegeben.

'-D3' : Nur die Namen der abweichenden Dateien werden ausgegeben, einer pro Zeile.

Wenn in der Kommandozeile kein Verzeichnis angegeben wird, nimmt LhA an, Sie wollen das Archiv mit dem aktuellen Verzeichnis vergleichen. Das zu vergleichende Verzeichnis wird genauso angegeben wie das Zielverzeichnis der Kommandos 'e' und 'x'.

BEISPIEL

'lha h arc:utils.lha sys:utilities/' wird alle Dateien im Archiv 'arc:utils.lha' mit den entsprechenden Dateien im Verzeichnis 'sys:utilities' vergleichen und alle Unterschiede aufzählen.

'lha -x0 h src:misc #?.c misc:' wird alle Dateien im Archiv 'src:misc.lha', deren Namen auf '.c' enden, mit den entsprechenden Dateien im Verzeichnis 'misc:' vergleichen.

'lha h dl:backup' wird die Dateien im Archiv 'dl:backup.lha' mit den Dateien und Verzeichnissen im aktuellen Verzeichnis vergleichen.

1.43 `l' Archivinhalt (knapp) auflisten

Dieses Kommando gibt eine knappe Liste der Inhalte einer Archivdatei, ↩
inklusive Dateinamen (inklusive Pfad), originale und komprimierte Größe, Datum der letzten Modifikation und Kompressionsrate.

Dateien mit Pfadnamen werden durch ein '+' vor dem Namen gekennzeichnet. Siehe untenstehendes Beispiel.

Dateikommentare werden bei Verwendung dieses Kommandos NICHT angezeigt, benutzen Sie hierfür das Kommando 'v' oder 'vv'.

Die Dateidefinition wird verwendet, um die anzuzeigenden Dateien zu bestimmen. Wenn keine Dateidefinition angegeben wird, werden alle Dateien aufgelistet.

BEISPIEL

```
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
```

Das '+' (Plus) Zeichen vor den ersten beiden Namen zeigt an, daß die Datei einen Pfad besitzt, der vom 'l' Kommando nicht angegeben wird (benutzen Sie das Kommando 'v' oder 'vv', um auch Pfadnamen anzuzeigen). Das '-N' unterdrückt den Copyright-Vermerk.

2.4.7.1 'lq' Archivinhalt (knapp-schnell) auflisten

1.44 `lq' Archivinhalt (knapp-schnell) auflisten

Dieses Kommando funktioniert genau wie das Kommando ‘l’, aber als einzige Information werden Dateinamen ohne Pfade aufgelistet. Leere Verzeichnisse werden als Leerzeile angezeigt.

1.45 `m’ Dateien ins Archiv verschieben

Dieses Kommando funktioniert genau wie das Kommando ‘a’, aber die Quelldateien werden nach dem erfolgreichen Archivieren gelöscht.

BEISPIEL

‘lha m includes.lzh src:*.[hi]’ wird alle Dateien des Verzeichnisses ‘src:’, die auf ‘.h’ oder ‘.i’ enden, in das Archiv ‘includes.lzh’ verschieben.

‘lha m myarc.lzh lhb_log.911012 lhb_idx.911012’ wird die beiden angegebenen Dateien (‘lhb_log.911012’ und ‘lhb_idx.911012’) in das Archiv ‘myarc.lzh’ verschieben.

1.46 `p’ Dateien nach stdout ausgeben

Dieses Kommando funktioniert genau wie die Extraktionskommandos (‘e’, ‘x’), aber sendet die extrahierten Daten an stdout (normalerweise die Konsole oder eine Datei, in die die Ausgabe umgelenkt wurde).

1.47 `r’ Dateien ersetzen

Dieses Kommando funktioniert genau wie die Kommandos zum aktualisieren / hinzufügen, aber ersetzt bereits im Archiv existierenden Dateien ohne Rücksicht auf das Dateidatum. (Das Kommando ‘u’ zusammen mit der ‘-Qr’ Option ist äquivalent zu diesem Kommando.)

1.48 `t’ Archivintegrität testen

Dieses Kommando überprüft die Integrität des angegebenen Archivs, indem die enthaltenen Dateien ins Nirgendwo entpackt werden, d.h. die Daten werden nur dekomprimiert, nicht in eine Datei geschrieben. Dieses Kommando funktioniert nur mit ganzen Archiven, d.h. Sie können nicht nur eine Datei im Archiv testen. Wenn dieses Kommando fehlschlägt, ist das Archiv korrupt, und ein warnender Rückgabecode wird ausgegeben.

BEISPIEL

‘lha t work:arcs/*’ wird die Integrität aller Archive im Verzeichnis ‘work:arcs’ testen.

‘lha t s:envarc.lzh’ überprüft die Integrität von ‘s:envarc.lzh’

'lha -R t dh0:*' wird die Integrität aller Archive auf dem Laufwerk 'dh0:' überprüfen ('-R' = Archive rekursiv suchen).

1.49 `u' Archiv aktualisieren

Wie der Kommandoname schon sagt, aktualisiert dieses Kommando Archive. Es fügt Dateien hinzu, die sich noch nicht im Archiv befinden, und ersetzt existierende aber ältere Dateien. Um festzustellen, welche Datei die neuere ist, wird das Dateidatum benutzt.

BEISPIEL

'lha u /aab/lha.lzh *.c' wird das Archiv '/aab/lha.lzh' mit allen '.c'-Dateien im aktuellen Verzeichnis aktualisiert.

1.50 `v' Archivinhalt (ausführlich) auflisten

Dieses Kommando funktioniert genau wie das 'l' Kommando, aber ↩ gibt den vollen Pfadnamen der Datei aus, während 'l' nur den Namen ohne Pfad ausgibt. Ein weiterer Unterschied zwischen 'l' und den 'v'/'vv' Kommandos ist, daß das 'l' Kommando keine Dateikommentare ausgibt. Dateikommentare werden in einer seperaten Zeile mit einem einleitenden Doppelpunkt (':') ausgegeben, genau wie durch das 'list' Kommando des AmigaDOS.

Die Dateidefinition wird verwendet, um die anzuzeigenden Dateien zu bestimmen. Wenn keine Dateidefinition angegeben wird, werden alle Dateien aufgelistet.

BEISPIEL

```
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
```

Das '-N' unterdrückt den Copyright-Vermerk.

2.4.13.1 'vq' Archivinhalt (ausführlich-schnell) auflisten

1.51 `vq' Archivinhalt (ausführlich-schnell) auflisten

Dieses Kommando funktioniert genau wie das 'l' Kommando, aber als einzige Information werden Dateinamen ohne Pfade aufgelistet.

1.52 `vv' Archivinhalt (vollständig) auflisten

Dieses Kommando ist genau wie das Kommando 'v', zeigt aber alle verfügbaren Informationen in einem etwas unterschiedlichen Format an. Die orginale und gepackte Größe, das Dateidatum und die Kompressionsrate wird genauso wie beim 'v' Kommando ausgegeben, plus die Dateiattribute ('Atts'), Kompressionsmethode, Datei-CRC und die DOS ID des OS, unter dem die Dateien komprimiert wurden. Wenn keine DOS ID für das Archiv angegeben ist (Archivkopfstufe < 1), wird ein Fragezeichen ausgegeben. Die häufigsten DOS IDs sind 'A', 'U' und 'M', wobei 'A' für AmigaDOS, 'U' für **IX und 'M' für MS-DOS steht. Der Dateiname inklusive Pfad wird in einer eigenen Zeile angezeigt. Dateikommentare werden auf die gleiche Weise wie beim 'v' Kommando ausgegeben, in einer eigenen Zeile nach dem Dateinamen. Die Archivkopfstufe wird ebenfalls angezeigt, und wenn irgendwelche nicht behandelten Erweiterungsfelder gefunden werden, wird nach der DOS ID ein 'X' ausgegeben.

Die Dateidefinition wird verwendet, um die anzuzeigenden Dateien zu bestimmen. Wenn keine Dateidefinition angegeben wird, werden alle Dateien aufgelistet.

BEISPIEL

```
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

Das '-N' unterdrückt den Copyright-Vermerk.
```

1.53 `x' Dateien inklusive Pfad extrahieren

Dieses Kommando funktioniert genau wie das 'e' Kommando, extrahiert Dateien aber immer inklusive Pfad (d.h. genau wie das 'e' Kommando mit aktivierter '-x' Option), unabhängig von der '-x' Option.

1.54 `y' Archiv mit neuen Optionen kopieren

Dieses Kommando nimmt ein Archiv als Input, und schreibt die ausgewählten (oder alle, wenn nicht anders angegeben) Dateien mit den neuen, in der Kommandozeile oder Umgebungsvariable angegebenen Optionen. Dies kann oft nützlich sein. Ein paar Beispiele werden sicher bei der Klärung helfen:

BEISPIEL

'lha -H1 y dl:#?' konvertiert alle Archive im 'dl:' Verzeichnis in Archive mit Stufe 1 Archivköpfen.

'lha -x0 y ram:files.lha *.c' wird im Archiv 'ram:files.lha' alle Pfade von allen Dateien entfernen, die mit '.c' enden.

ANMERKUNG

Momentan ignoriert LhA die Angabe der Kompressionsmethode, so das dieses Kommando nicht benutzt werden kann, um alte -lh1Archive in neue -lh5- Archive oder andersherum zu konvertieren. Dies wird in einer kommenden Version möglich sein.

1.55 Optionen

Dieser Abschnitt beschreibt die verschiedenen Optionen, die ↩
Ihnen bei der Benutzung von LhA zur Verfügung stehen. Lesen Sie Abschnit 2.1.1 für eine genaue Beschreibung, wie und wo Optionen (de-)aktiviert werden können. Die Buchstaben in Klammern geben an, welche Kommandos durch die Option beeinflußt werden.

Code	Kommandos
(add)	a,u,f
(all)	alle Kommandos
(ext)	e,x
(upx)	a,u,f,e,x
(upd)	a,u,f,d

2.5.1 '-a' (upx) Dateiattribute bewahren

2.5.2 '-A' (upd) Archivattribute setzen

2.5.3 '-b' (all) I/O Puffergröße setzen

2.5.4 '-B' (upd) Archiv-Backups erstellen

2.5.5 '-c' (all) Dateien bestätigen

2.5.6 '-C' (ext) Archivbit beim Extrahieren löschen

2.5.7 ‘-d’ (upd) Archivdatum=Neuste Datei

2.5.8 ‘-D’ (all) Alternative Fortschrittanzeige

2.5.9 ‘-e’ (add) Leere Verzeichnisse archivieren

2.5.10 ‘-E’ (ext) Extrahierte Dateien berühren

2.5.11 ‘-f’ (all) Dateikommentare ignorieren

2.5.12 ‘-F’ (all) Schnelle Fortschrittanzeige

2.5.13 ‘-G’ (ext) Nur neuere Dateien extrahieren

2.5.14 ‘-h’ (add) Keine Ausgangsverzeichnisse

2.5.15 ‘-H’ (add) Archivkopfstufe

2.5.16 ‘-i’ (all) Dateiliste aus Datei lesen

2.5.17 ‘-I’ (all) Ignoriere LHAOPTS-Variable

2.5.18 ‘-k’ (all) Bewahre Teildateien

2.5.19 ‘-K’ (move) Leere Verzeichnisse löschen

2.5.20 ‘-l’ (ALL) Dateinamen in Kleinschrift wandeln

2.5.21 ‘-L’ (ALL) Dateiliste anlegen

2.5.22 ‘-m’ (ALL) Keine Rückfragen

2.5.23 ‘-M’ (ext) Keine Autoshow-Dateien

2.5.24 ‘-n’ (upx) Keine Byte-Fortschrittanzeige

2.5.25 ‘-N’ (all) Keine Fortschrittanzeige

2.5.26 ‘-o’ (add) Nur Dateien mit demselben oder neuerem Datum

2.5.27 ‘-O’ (add) Nur Dateien mit demselben oder älterem Datum

2.5.28 ‘-p’ (ALL) Pause nach dem Laden

2.5.29 ‘-P’ (ALL) Taskpriorität setzen

2.5.30 ‘-q’ (ALL) Quiet (Stumm)

2.5.31 ‘-Q’ (ALL) Alternativer Optionssatz

2.5.32 ‘-r’ (add) Dateien rekursiv suchen

2.5.33 ‘-R’ (ALL) Archive rekursiv suchen

2.5.34 ‘-s’ (add) Nur Dateien mit gelöschten A-Bit hinzufügen

2.5.35 ‘-S’ (add) A-Bit bei archivierten Dateien setzen

2.5.36 ‘-t’ (ext) Nur neue Dateien

2.5.37 ‘-T’ (upx) Neue und neuere Dateien

2.5.38 ‘-u’ (ALL) Dateinamen in Großschrift wandeln

2.5.39 ‘-V’ (all) Mehr-Medien-Archive aktivieren

2.5.40 ‘-w’ (upd) Setze Arbeitsverzeichnis

2.5.41 ‘-W’ (add) Dateinamen ausschließen

2.5.42 ‘-x’ (all) Pfadnamen beibehalten

2.5.43 ‘-X’ (ALL) Keine Namenserweiterung

2.5.44 ‘-y’ (all) Namenserweiterung immer anhängen

2.5.45 ‘-Y’ (add) Speichere große Dateien nach Rate

2.5.46 ‘-z’ (add) Dateien nicht komprimieren

2.5.47 ‘-Z’ (add) Archive komprimieren

2.5.48 ‘-0’ (add) Benutze LhArc 1.x-Kompression

2.5.49 ‘-2’ (add) Benutze LhA Kompression (-lh5-)

2.5.50 ‘-3’ (add) Benutze LhA Kompression (-lh6-)

2.5.51 ‘-Qa’ (all) Benute einfache Konsolen-I/O

2.5.52 ‘-Qb’ (ext) Archiv vor dem Extrahieren testen

2.5.53 ‘-Qd’ (ext) Autoshow-Dateien löschen

2.5.54 ‘-Qh’ (add) Größe des Huffman-Puffers setzen

2.5.55 ‘-Qm’ (all) Dateien in Fortschrittanzeige kappen

2.5.56 ‘-Qn’ (all) Nationalen Zeichenmodus aktivieren

2.5.57 ‘-Qo’ (all) Optionen nach Kommando ignorieren

2.5.58 ‘-Qp’ (move) Ignoriere Löschen-Schutzbit

2.5.59 ‘-Qq’ (add) Schnelles Hinzufügen

2.5.60 ‘-Qr’ (add) Dateidatum ignorieren

2.5.61 ‘-Qv’ (all) Setze Laufwerke für Mehr-Medien-Archive

2.5.62 ‘-Qw’ (all) Keine Namensmuster

1.56 `-a' (upx) Dateiattribute bewahren

Diese Option, wenn aktiviert, wird die Datei-Schutzbits bewahren. Die acht Attribute sind hier aufgelistet:

r: Lesen ('r'ead) - dieses Bit wird für Dateien gesetzt, die lesbar sind (eine Datei ist Lesegeschützt, wenn das Bit gelöscht ist).

w: Schreiben ('w'rite) - dieses Bit wird für Dateien gesetzt, die beschreibbar sind (eine Datei ist Schreibgeschützt, wenn dieses Bit gelöscht ist).

e: Ausführen ('e'xecute) - Dieses Bit wird für Dateien gesetzt, die ausführbar sind (Programme und Shell-Skripte müssen dieses Bit gesetzt haben).

d: Löschen ('d'elete) - Dieses Bit wird für Dateien gesetzt, die löschbar sind (eine Datei ist Löschgeschützt, wenn dieses Bit gelöscht ist).

a: Archiviert - Dieses Bit wird von Festplatten-Backup-Programmen (und optional LhA) benutzt, um anzuzeigen, welche Dateien seit dem letzten Backup verändert wurden. Wenn dieses Bit gesetzt ist, zeigt dies, daß die Datei nicht verändert wurde; und wenn es gelöscht ist, wurde die Datei seit dem letzten Backup verändert. Das Bit wird gelöscht, wann immer schreibend auf die Datei zugegriffen wird.

p: Pure - Dieses Bit wird für Programmdateien gesetzt, die "pure" sind (d.h. Multitasking-Reentrant), und die mit dem AmigaDOS 'resident' oder einem äquivalenten Kommando resident gemacht werden können.

s: Skript - Dieses Flag wird für Shell-Skriptdateien gesetzt.

h: Halten - Dieses Bit sagt der Shell, das Programm beim Start automatisch resident zu machen. Um dies ausführen zu können, müssen die Bits p (Pure) und e (Ausführbar) gesetzt sein. Dieses Bit wird nur von der 2.04 - 3.1 Shell erkannt.

Bitte wenden Sie sich an das AmigaDOS-Handbuch für detailiertere Erklärungen der verschiedenen Datei-Schutzbits.

Wenn die Option deaktiviert ist (durch ein '-a0' auf der Kommandozeile), werden die Datei-Schutzbits aller extrahierten und archivierten Dateien auf '----RWED' gesetzt. Wichtig: Sie MÜSSEN diese Option sowohl beim Archivieren als auch beim Dearchivieren einschalten, um die Dateiattribute korrekt zu bewahren.

ANMERKUNG

Benutzen Sie diese Option nur, wenn Sie wissen, daß

das Archiv auf einem Amiga komprimiert wurde oder dekomprimiert werden wird. Das Format der Attributsfelder ist für verschiedene Betriebssysteme unterschiedlich. Wenn Sie Archivköpfe der Stufe 1 oder höher verwenden, brauchen Sie darüber nicht nachzudenken, da der Archivierer dann erkennt, unter welchem OS das Archiv erstellt wurde, und die Archivbits nur unter dem gleichen OS benutzt. Lassen Sie diese Option immer eingeschaltet, wenn Sie Archivköpfe der Stufe 1 oder höher verwenden!

Diese Option wird beim Archivieren (a,f,u,m) per Default aktiviert, und für alle anderen Kommandos per Default deaktiviert.

1.57 `-A' (upd) Archivattribute setzen

Wenn diese Option aktiv ist, setzt LhA die Datei-Schutzbits aller aktualisierten Archive auf '----RW-D'.

Diese Aktion ist per Default AKTIVIERT.

1.58 `-b' (all) I/O Puffergröße setzen

('b'uffer size)
Diese Option setzt die Größe der I/O-Puffer, die LhA beim Lesen und Schreiben von Archivdateien benutzt. Sie können die Puffergröße auf jeden Wert zwischen 8 und 64 KB setzen. Größere Puffer machen LhA normalerweise etwas schneller (abhängig von der Art des Archivs, und welche Dateien ausgewählt wurden).

BEISPIEL

'lha -b64 a archive.lzh hubba' : fügt die Datei 'hubba' 'archive.lzh' hinzu, wobei ein I/O-Puffer von 64K verwendet wird.

ANMERKUNG

LhA auf einem beschleunigten (68020+) Amiga mit kleinem I/O-Puffer zu betreiben, wird die Kompressions- / Dekompressionsrate signifikant verschlechtern! Die vorgegebene Puffergröße von 32KB ist in den meisten Fällen ausreichend, und funktioniert auch auf nicht beschleunigten Amigas gut. Beachten Sie zusätzlich, daß beim Betrieb von LhA auf einem Medium wie der Ram Disk die I/O-Puffergröße weniger wichtig ist, und der Einsatz eines großen Puffers unnötig ist. Die Vorgabegröße von 32K ist für die meisten Setups eine gute Wahl.

Die Vorgabegröße ist 32K (32768 Bytes).

1.59 `-B' (upd) Archiv-Backups erstellen

Wenn diese Option eingeschaltet ist, wird LhA immer ein Backup des Archivs behalten, wann immer eine Datei durch die Kommandos Löschen, Aktualisieren, Auffrischen oder Ersetzen entfernt wird. Das Archiv-Backup erhält den Namen '<arcname>.bak' (beachten Sie, daß das Suffix '.lzh' oder '.lha' *NICHT* durch das Suffix '.bak' ersetzt wird -vielmehr wird das Suffix '.bak' immer an das Ende des Dateinamens angehängt).

Diese Option ist per Default DEAKTIVIERT.

1.60 `-c' (all) Dateien bestätigen

('c'onfirm files)
Wenn diese Option aktiviert ist, wird LhA für alle bearbeiteten Dateien und Archive um eine Bestätigung bitten.

Diese Option ist per Default DEAKTIVIERT.

1.61 `-C' (ext) Archivbit beim Extrahieren löschen

('C'lear arc-bit on extract)
Wenn diese Option aktiv ist, wird LhA das A-Schutzbit für alle Dateien maskieren, die extrahiert werden. Das ist nützlich, wenn man Dateien von Archiven auf Festplatte extrahiert, da die extrahierten Dateien vom BackupProgramm sonst nicht als neu oder verändert erkannt würden, wenn das A-Bit gesetzt wäre.

Diese Option ist per Default AKTIVIERT.

1.62 `-d' (upd) Archivdatum=Neueste Datei

(archive 'd'ate = newest file)
Wenn diese Option aktiv ist, wird LhA das Datum der letzten Modifikation des Archivs auf das gleiche Datum wie das der zuletzt modifizierten Datei in dem Archiv setzen. Dies gibt das wahre Alter des Archivs besser wieder als das Datum des letzten Archivupdates.

Diese Option ist per Default DEAKTIVIERT.

1.63 `-D' (all) Alternative Fortschrittanzeige

(alternative 'D'isplay)
Dieser Schalter wird benutzt, um das Aussehen der Byte-Fortschrittanzeige zu verändern, die von LhA beim Komprimieren und Dekomprimieren von Dateien angezeigt wird. Es gibt verschiedene Typen

von Fortschrittanzeigern, so daß Sie durch die Angabe einer Ziffer nach dem ‘-D’-String die gewünschte Variante spezifizieren können.

0: Dies ist die Standardanzeige, die die Anzahl der bearbeiteten Bytes sowie die Gesamtzahl der Bytes in der Datei wie folgt anzeigt:

(xxxxxxx/yyyyyyy) wobei x = verarbeitete Bytes, und y = Gesamte Bytes in der Datei.

1: Diese Anzeige zeigt einfach eine ‘rotierende Linie’, die jeweils um 45 Grad gedreht wird, wenn die Anzeige aktualisiert wird.

2: Diese Anzeige zeigt, wieviel Prozent der Datei von LhA schon verarbeitet wurden.

3: Diese Anzeige zeigt einen wachsenden Balken, der anzeigt, welcher Teil der Datei bereits bearbeitet wurde.

4: Diese Anzeige zeigt einen grafischen Füllbalken in einem Intuition-Fenster. Perfekt für die Verwendung in einem Directory-Utility (wie z.B. Directory Opus), wo man keine Textausgabe braucht, aber immer noch wissen will, daß LhA tatsächlich arbeitet.

BEISPIEL

‘lha -D2 a src *.asm’ wird dem Archiv ‘src.lha’ hinzufügen, wobei eine Prozentanzeige verwendet wird (Typ 2).

‘lha -D4 x foo’ wird Dateien extrahieren und dabei einen grafischen Füllbalken anzeigen.

‘lha -q -D4 x foo’ wird Dateien aus dem Archiv ‘foo.lha’ extrahieren, einen grafischen Füllbalken und KEINERLEI Konsolenausgabe zeigen.

ANMERKUNG

In Verbindung mit dem Kommando ‘h’ hat diese Option eine etwas andere Bedeutung. Lesen Sie den Abschnitt über das Kommando ‘h’ für eine detaillierte Erklärung.

Die Vorgabe für die Fortschrittanzeige ist Typ 0.

1.64 `-e’ (add) Leere Verzeichnisse archivieren

(archive ‘e’mpty directory)
Wenn diese Option in Verbindung mit ‘-r’ verwendet wird (Dateien rekursiv suchen), wird LhA alle leeren Unterverzeichnisse mit archivieren.

Diese Option ist per Default DEAKTIVIERT (leere Unterverzeichnisse werden nicht archiviert).

1.65 `-E' (ext) Extrahierte Dateien berühren

Wenn diese Option aktiviert ist, wird LhA das Dateidatum aller extrahierten Dateien auf das aktuelle Datum stellen. Das kann nützlich sein, wenn Sie Festplattenbackups nach Datum statt nach Archivbit erstellen.

Diese Option ist per Default DEAKTIVIERT (das originale Dateidatum wird wiederhergestellt).

1.66 `-f' (all) Dateikommentare ignorieren

(ignore 'f'ilenotes)
Wenn diese Option aktiviert ist, wird LhA keine Dateikommentare speichern oder wiederherstellen. Es ist eigentlich nicht wirklich nötig, da es aufgrund der Art und Weise, wie die Dateikommentare gespeichert werden, keine Kompatibilitätsprobleme mit anderen Systemen gibt. Wenn es trotz allem Probleme geben sollte, aktivieren Sie versuchsweise diese Option oder benutzen Sie Archivköpfe der Stufe 1 oder höher, sofern das Zielsystem diese unterstützt.

Lesen Sie den Abschnitt über Kompatibilität (1.7) für eine Diskussion über dieses und andere Themen der Kompatibiltät.

Diese Option ist per Default DEAKTIVIERT (Dateikommentare werden gespeichert und wiederhergestellt).

1.67 `-F' (all) Schnelle Fortschrittanzeige

In diesem Modus benutzt LhA eine unterschiedliche Methode für die Fortschrittanzeige. Normalerweise gibt LhA einen Zeilenvorschub (LF) nach jeder bearbeiteten Datei aus, so daß die Anzeige um eine Zeile vorgeschoben / gescrollt wird. In diesem Modus gibt LhA nur dann ein LF aus, wenn ein Fehler auftritt. Das ist nützlich, wenn Sie Archive mit einer Vielzahl von kleinen Dateien testen oder extrahieren, und das Scrolling länger dauert als die eigentliche Dekompression!

ANMERKUNG

> Wenn Sie die Default-Fortschrittanzeige auf einem sehr schnellen Amiga-System (68020+) benutzen, beachten Sie, daß das Scrollen des Bildschirms tatsächlich mehr Zeit brauchen kann als die eigentliche Dekompression! Das trifft vor allem auf Archive mit vielen kleinen Dateien zu. Also benutzen Sie sie nicht, solange Sie nicht unbedingt sehen _müssen_, welche Dateien bearbeitet wurden. LhA scrollt die Anzeige, sobald ein Fehler bei einer Datei auftritt, so daß Sie Fehlermeldungen immer noch sehen (sogar noch besser, da die einzigen Dateinamen, die nach der Aktion auf dem Bildschirm verbleiben, die der fehlgeschlagenen Dateien sind!).

Wenn Sie die Anzeigeoption ‘-D4’ benutzen, hat diese Option keinen zusätzlichen Effekt, da die Fortschrittinformation in einem seperaten Fenster angezeigt wird, so daß es keine Extra-LFs gibt, die überhaupt ein Scrolling verursachen würden.

Diese Option ist per Default DEAKTIVIERT (es wird die alte Fortschrittanzeige benutzt).

1.68 `-G’ (ext) Nur neuere Dateien extrahieren

Wenn diese Option aktiviert ist, wird LhA nur Dateien extrahieren, die bereits existieren, und die ein neueres Dateidatum haben als die bereits existierenden Dateien.

Diese Option ist per Default DEAKTIVIERT.

1.69 `-h’ (add) Keine Ausgangsverzeichnisse

(disable ‘h’omedirectories)
Wenn diese Option angegeben wird, werden die Ausgangsverzeichnis-Funktionen von LhA deaktiviert.

Diese Option ist per Default DEAKTIVIERT (Ausgangsverzeichnisse werden erkannt).

1.70 `-H’ (add) Archivkopfstufe

(archive ‘H’eader level)
Mit diesem Schalter können Sie entscheiden, welchen Archivkopf-Typ Sie benutzen wollen. Die momentan gültigen Archivkopfstufen sind 0, 1 und 2. Bitte wenden Sie sich an den Abschnitt über Archivköpfe für eine detailiertere Exrklärung über die verschiedenen Archivkopfstufen.

Die Vorgabestufe für Archivköpfe ist 0.

1.71 `-i’ (all) Dateiliste aus Datei lesen

(‘i’nsert filelist file)
Mit dieser Option können Sie eine Liste von Dateien aus einer Datei einlesen, statt alle zu behandelnden Dateien in der Kommandozeile zu definieren.

BEISPIEL

Wenn die Datei ‘ArcFList’ die folgenden Einträge enthält:

```
  ---> Start der ArcFList-Einträge  (diese Zeile ist NICHT in der Datei)

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

  ---> Ende der ArcFList-Einträge   (diese Zeile ist NICHT in der Datei)

  Die folgende Kommandozeile:

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

  Tut das Gleiche wie dieses Kommando:

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

                           ANMERKUNG

       Dieses Kommando funktioniert fast genauso  wie  die
       folgende Kommandozeile:

       LhA ? ???? @file

       Also  können  Sie  Optionen  in  Ihrer  '-i'-Datei
       einfügen.  Der  einzige  Unterschied  ist,  daß die
       '-i'-Datei     keine     Zielverzeichnis-Definition
       enthalten   kann,   während   Sie   dies   mit  der
       @file-Methode      erreichen      können.      Das
       Zielverzeichnis  wird  bei Verwendung der Option -i
       immer der Kommandozeile entnommen.
```

Lesen Sie den Abschnitt über '@'(include)-Dateien für eine Alternative, dasselbe zu erreichen.

1.72 `-l' (all) Ignoriere LHAOPTS-Variable

Wenn diese Option angegeben wird, wird LhA nicht versuchen, die Defaults aus der lokalen oder globalen Umgebungsvariable LHAOPTS auszulesen. Beachten Sie das diese Option ein Spezialfall ist, weil Sie direkt nach einem Strich ('-') auf der Kommandozeile eingegeben werden muß.

Diese Option ist per Default DEAKTIVIERT.

1.73 `-k' (all) Bewahre Teildateien

('k'eep partial files)
Wenn aktiviert, wird diese Option verhindern, daß LhA Teildateien löscht sobald ein Fehler auftritt. Normalerweise werden Teildateien, deren CRC-Check fehlschlägt, die I/O-Fehler verursachen oder die mit CTRL-C abgebrochen werden, gelöscht, bevor LhA mit einer Fehlermeldung abbricht. Mit dieser Option können Sie LhA zwingen, diese (oft) nur teilweise behandelten Dateien zu bewahren. Das kann nützlich sein, wenn Sie versuchen, Daten aus einem beschädigten Archiv zu extrahieren - LhA

wird versuchen, die Daten aus dem fehlerhaften Archiv zu extrahieren und wird die Datei mit einem speziellen Dateikommentar versehen, um anzuzeigen, daß der CRC-Check fehlgeschlagen ist, und die Datei wahrscheinlich beschädigt ist.

ANMERKUNG

> Bitte beachten Sie daß, in der aktuellen Version und aufgrund der internen I/O-Pufferung, bei bestimmten Fehlern nicht alle extrahierten Daten in der teilweise extrahierten Datei enthalten sind. Setzen Sie in diesem Fall den I/O-Puffer auf den kleinstmöglichen Wert (8KB), um so viel wie möglich zu retten. Aus diesem Grund kann es sein, daß kleine Dateien überhaupt nicht restauriert werden können. Dies trifft nur auf LHA (-lh5-) Kompression zu, LhArc-komprimierte Dateien enthalten immer alle extrahierten Daten in der teilweise extrahierten Datei.

Diese Option ist per Default DEAKTIVIERT (Teildateien werden gelöscht).

1.74 `-K' (move) Leere Verzeichnisse löschen

('K'ill empty directories)
Wenn diese Option zusammen mit dem Kommando Verschieben ('m') benutzt wird, wird LhA nach dem Verschieben aller Dateien in das Archiv alle leeren Verzeichnisse löschen. Nützlich, um einen ganzen Baum von Unterverzeichnissen mit der Option '-r' (Dateien rekursiv suchen) zu verschieben.

Diese Option ist per Default DEAKTIVIERT (leere Verzeichnisse werden nicht gelöscht).

1.75 `-l' (ALL) Dateinamen in Kleinschrift wandeln

(make filenames 'l'owercase)
Wenn aktiviert, wird diese Option LhA veranlassen, alle Dateinamen in Kleinschrift umzuwandeln. Das ist nützlich, wenn Dateien aus einem Archiv extrahiert werden, das auf einem MSDOS-System erstellt wurde, dessen Dateinamen alle in Großschrift sind, was vollkommen schwachsinnig aussieht (IMHO). Benutzen Sie diese Option, um sie besser aussehen zu lassen!

BEISPIEL

> 'LhA -l x myarc' wird alle Dateien aus 'myarc.(lzh|lha)' extrahieren, und alle Dateinamen in Kleinschrift umwandeln.

Diese Option ist per Default DEAKTIVIERT.

1.76 `-L' (ALL) Dateiliste anlegen

(make file 'L'ist)
Wenn diese Option aktiviert ist, wird sie LhA veranlassen, eine Liste aller Dateien anzulegen, die bearbeitet wurden (d.h., welche Dateien bei der letzten Operation den Dateidefinitionen entsprachen, die Sie in der Kommandozeile angegeben haben). Der Name der Listendatei muß unmittelbar auf den '-L'-String folgen. Wenn Sie Leerzeichen im Dateinamen benötigen, müssen Sie den Namen in Anführungsstriche setzen.

BEISPIEL

'lha -Lram:ListFile d src.lzh *.asm' wird alle Dateien in 'src.lzh' löschen, die auf '.asm' enden, und eine Liste der gelöschten Dateien in 'ram:ListFile' anlegen.

'lha -L"ram:List File" u src.lzh *.asm' wird 'src.lzh' aktualisieren, und eine Liste der hinzugefügten/ersetzten Dateien in 'ram:List File' anlegen.

ANMERKUNG

Die von dieser Option angelegte Datei ist eine einfache ASCII-Datei mit jedem Namen in einer einzelnen Zeile. Die Dateien, die von dieser Option erzeugt werden, können als Aktions- oder Ausschlußlisten für die Optionen '@' oder '-i' von LhA verwendet werden.

Diese Option ist per Default DEAKTIVIERT (es wird keine Dateiliste erzeugt).

1.77 `-m' (ALL) Keine Rückfragen

(no 'm'essages)
Wenn diese Option aktiviert ist, wird LhA alle Rückfragen unterdrücken, die normalerweise gestellt werden z.B. beim Überschreiben existierender Dateien. Mit dem Einschalten dieser Option wird LhA auch TelOps (Autoshow-Dateien) ignorieren. Wenn diese Option aktiviert ist, wird LhA sich so verhalten, als ob Sie die Default-Aktion als Antwort auf alle Rückfragen gegeben hätten (ja). Diese Option wird automatisch aktiviert, wenn der Standard-Input nicht Interaktiv ist (z.B. beim Start im Hintergrund).

Diese Option ist per Default DEAKTIVIERT.

1.78 `-M' (ext) Keine Autoshow-Dateien

Wenn diese Option aktiviert ist, wird LhA die Anzeige von Autoshow-Dateien unterdrücken (Dateien, die auf '.displayme' enden).

ANMERKUNG

Autoshow-Dateien werden auch dann unterdrückt, wenn eine oder mehrere der Optionen '-N', '-q' oder '-m' aktiviert werden.

Diese Option ist per Default DEAKTIVIERT (Autoshow-Dateien werden angezeigt).

1.79 `-n' (upx) Keine Byte-Fortschrittanzeige

('n'o byte progress indicator)
Wenn diese Option aktiviert ist, wird die Fortschrittanzeige ausgeschaltet. LhA wird allerdings immer noch anzeigen, welche Dateien gerade bearbeitet werden; benutzen Sie '-N', um jede Fortschrittanzeige auszuschalten.

Diese Option ist per Default DEAKTIVIERT.

1.80 `-N' (all) Keine Fortschrittanzeige

('N'o progress indicator)
Diese Option gleicht der Option '-n', unterdrückt aber die Fortschrittanzeige auf einer höheren Ebene (d.h. die Anzeige, welche Datei LhA gerade bearbeitet. Sie deaktiviert auch die kurze Copyright-Anzeige, die ansonsten bei jedem Aufruf angezeigt wird.

Diese Option ist per Default deaktiviert (Fortschrittanzeige AN).

1.81 `-o' (add) Nur Dateien mit demselben oder neuerem Datum

Momentan nicht implementiert (war auch niemals).

1.82 `-O' (add) Nur Dateien mit demselben oder älterem Datum

Momentan nicht implementiert (war auch niemals).

1.83 `-p' (ALL) Pause nach dem Laden

Wenn aktiviert, veranlaßt diese Option LhA, zu pausieren und auf einen Tastendruck des Anwenders zu warten, bevor ein Kommando ausgeführt wird. Das ist nützlich für Anwender mit Floppies, die dann nach dem Laden von LhA Disketten wechseln können, während LhA auf einen Tastendruck wartet.

Diese Option ist per Default DEAKTIVIERT.

1.84 `-P' (ALL) Taskpriorität setzen

Diese Option wird verwendet, um die Priorität des LhA-Prozesses zu setzen. Die Priorität kann auf jeden Wert im Bereich -5 bis +5 gesetzt werden, inklusive 0. Je höher die Priorität ist, die Sie LhA zuteilen, je mehr CPU-Zeit wird es an sich nehmen (Prozesse mit niedrigerer Priorität werden fast niemals eine Chance bekommen, zu laufen, da LhA sehr Prozessorintensiv ist). Eine niedrige Einstellung (wie -5) wird LhA dazu bringen, nur die Prozessorzeit zu nutzen, die niemand sonst will (nett, wenn man LhA als Hintergrundtask laufen läßt, während ein comm-Programm läuft).

Die Priorität muß durch eine einzelne Ziffer direkt nach dem P angegeben werden (optional mit einem Minuszeichen für negative Priorität), wie in:

BEISPIEL

'lha -P-1 a nonsense.lzh bogus.txt' wird LhA die Datei 'bogus.txt' dem Archiv 'nonsense.lzh' hinzufügen, bei einer Priorität von -1.

Die Default-Priorität wird vom aufrufenden Prozess geerbt (d.h. von der CLI oder dem Programm, das Execute()/RunCommand() aufgerufen hat). Das ist normalerweise Null (0).

1.85 `-q' (ALL) Quiet (Stumm)

Diese Option wird ALLE Konsolenausgaben von LhA unterdrücken.

ANMERKUNG

Diese Option hat keinen Effekt auf das Fortschrittfenster, das Sie mit der Option '-D4' sehen werden. Das geschah mit Absicht, zur Verwendung mit Directory-Utilities, Installer, oder etwas anderem, das LhA im Hintergrund starten, aber trotzdem den Fortschritt von LhA's Operation mit einem Archiv anzeigen will.

Diese Option ist per Default DEAKTIVIERT.

1.86 `-Q' (ALL) Alternativer Optionssatz

Dieser Optionscharakter ('Q') veranlaßt, das alle folgenden Optionscharaktere bis zum nächsten Leerzeichen als erweiterte Optionen interpretiert werden. Diese sind am Ende dieses Abschnitts dokumentiert.

1.87 `-r' (add) Dateien rekursiv suchen

Wenn diese Option verwendet wird, sucht LhA Dateien rekursiv in Unterverzeichnissen.

BEISPIEL

'lha -r a ram:disk1 df0:' wird alle Dateien auf der Diskette im Laufwerk 0 in 'ram:disk1.lha' archivieren.

'lha -r a ram:disk2src df0:*.c' wird alle auf '.c' endenden Dateien auf df0: in 'ram:disk2src.lha' archivieren.

'lha -r a ram:exthup hd:prg/src/ lha/*.[chasi] lhi/*.[chasi]' wird alle Dateien, die auf '.c', '.h', '.a', '.s', oder '.i' enden, in 'hd:prg/src/lha' und 'hd:prg/src/lhi' und deren Unterverzeichnissen archivieren. Der 'hd:prg/src/'-Teil der Namen wird nicht im Archiv gespeichert werden ('hd:prg/src/' wurde als Ausgangsverzeichnis angegeben).

ANMERKUNG

Dateien, die explizit angegeben werden (d.h. ohne die Verwendung von Namensmustern), werden nur im aktuellen (Ausgangs-) Verzeichnis gesucht, während Namensmuster zur Suche in allen Unterverzeichnissen verwendet werden. Wenn ein Verzeichnis explizit ohne folgendes Namensmuster angegeben wird (wie in 'lha -r a ram:test sys:l'), wird es behandelt, als ob ein '/*' an den Verzeichnisnamen angehängt worden wäre -d.h. alle Dateien in dem Verzeichnis und seinen Unterverzeichnissen wird archiviert.

Diese Option ist per Default DEAKTIVIERT. Beachten Sie, daß die Option '-x' automatisch zusammen mit der Option '-r' aktiviert wird. Wenn Sie nicht wollen, daß Pfadnamen gespeichert werden, geben Sie einfach '-x0' auf der Kommandozeile an.

1.88 `-R' (ALL) Archive rekursiv suchen

Wenn diese Option aktiviert ist, wird LhA Archivdateien rekursiv anhand der Archivdefinition der Kommandozeile suchen. Das funktioniert wie die Option '-r', nur für Archivdateien.

BEISPIELE

'lha -R l dh0:files/a*' wird den Inhalt aller Archivdateien in 'dh0:files' und dessen Unterverzeichnissen auflisten, die mit 'a' beginnen.

'lha -R l *' wird den Inhalt aller Archive im aktuellen Verzeichnis und dessen Unterverzeichnissen auflisten.

'lha -R l myarc' wird den Inhalt aller Archive mit Namen 'myarc.lzh' oder 'myarc.lha' im aktuellen Verzeichnis und dessen Unterverzeichnissen auflisten.

Diese Option ist per Default DEAKTIVIERT.

1.89 `-s' (add) Nur Dateien mit gelöschten A-Bit hinzufügen

(only files with arc-bit un's'et)
Wenn diese Option aktiviert ist, wird LhA nur Dateien hinzufügen, bei denen das Dateischutzbit 'A' (für Archiviert) nicht gesetzt ist. Das ist nützlich für inkrementierende Backups in Verbindung mit der Option '-S'.

Diese Option ist per Default DEAKTIVIERT (Dateien ohne Rücksicht auf Schutzbits hinzufügen).

1.90 `-S' (add) A-Bit bei archivierten Dateien setzen

('S'et arc-bit)
Wenn diese Option aktiviert ist, wird LhA das Schutzbit 'A' (für 'A'rchiviert) bei allen Dateien setzen, die zu einem Archiv hinzugefügt werden. Dies kann verwendet werden, um in Verbindung mit der Option '-s' (Nur Dateien mit gelöschten A-Bit hinzuzufügen) automatische Backups zu vereinfachen. Lesen Sie den vorigen Abschnitt für weitere Details.

Diese Option ist per Default DEAKTIVIERT.

1.91 `-t' (ext) Nur neue Dateien

Wenn diese Option aktiviert ist, wird LhA keine Dateien überschreiben oder ersetzen.

ANMERKUNG

Diese Option überschreibt die Option '-T'.

Diese Option ist per Default DEAKTIVIERT.

1.92 `-T' (upx) Neue und neuere Dateien

Wenn diese Option aktiviert ist, wird LhA Dateien überschreiben oder ersetzen, die bereits existieren und älter als die aktuelle Datei sind, und nicht existierende Dateien erstellen.

ANMERKUNG

Diese Option überschreibt die Option '-T'.

Diese Option ist per Default DEAKTIVIERT.

1.93 `-u' (ALL) Dateinamen in Großschrift wandeln

(make filenames ‘u’ppercase)
Wenn diese Option aktiviert ist, wird LhA alle Dateinamen in Großschrift wandeln. Das kann nützlich sein, wenn man Archive erstellt, die auf einem MSDOS-System mit LhArc/LHA verwendet werden sollen. Auch wenn diese keine Schwierigkeiten mit dem Extrahieren von Dateien gemischter Schreibweise haben, funktioniert die Erkennung von Namensmustern in diesem Fall nicht korrekt.

Diese Option ist per Default DEAKTIVIERT.

1.94 `-V’ (all) Mehr-Medien-Archive aktivieren

(set multi-‘V’olume-archives)
Diese Option aktiviert die Mehr-Medien-Funktion von LhA. Bitte lesen Sie den Abschnitt über Mehr-Medien-Archive für weitere Informationen. Lesen Sie bitte auch den Abschnitt über die Option ‘-Qv’. Weitere Optionen müssen von dem ‘V’ durch mindestens ein Leerzeichen getrennt sein. Die gewünschte Mediengröße in KB sollte nach dem Buchstaben ‘V’ angegeben werden. Wenn Sie wollen, daß LhA die verfügbare Mediengröße automatisch erkennt, benutzen Sie ‘-Va’ (für ‘allen verfügbaren Platz verwenden’).

BEISPIEL

‘LhA -Va a df0:MyArc *.c’ wird alle Dateien im aktuellen Verzeichnis, die auf ‘.c’ enden, auf DF0: archivieren. Wenn die Diskette voll ist, bevor das Archiv abgeschlossen ist, wird LhA dazu auffordern, eine neue Diskette einzulegen.

Diese Option ist per Default DEAKTIVIERT.

1.95 `-w’ (upd) Setze Arbeitsverzeichnis

(set ‘w’ork archives)
Diese Option wird benutzt, um anzugeben, welches Verzeichnis LhA zur Speicherung temporärer Dateien verwenden soll. Temporäre Dateien werden angelegt, wenn Dateien zu Archiven hinzugefügt werden, oder ein Archiv irgendwie aktualisiert wird (z.B. beim Löschen oder Aktualisieren von Dateien). Der Name des Arbeitsverzeichnisses muß unmittelbar nach dem ‘-w’-String angegeben werden.

BEISPIEL

‘LhA -wrad:tmp a MyArc.lzh *’ wird beim Hinzufügen aller Dateien im aktuellen Verzeichnis zum Archiv ‘MyArc.lzh’ das Verzeichnis ‘rad:tmp’ als temporären Speicher benutzen.

Als Default benutzt LhA das Verzeichnis ‘T:’ für temporäre Dateien. Wenn diese Zuweisung oder dieses Laufwerk nicht existiert, wird LhA das aktuelle Verzeichnis benutzen.

1.96 `-W' (add) Dateinamen ausschließen

Diese Option ist in der aktuellen Version nicht verfügbar.

1.97 `-x' (all) Pfadnamen beibehalten

Seit LhA V1.30 gibt es diese Option in drei Variationen; welchen Modus LhA verwendet, ist abhängig von der Ziffer, die dem 'x' folgt.

'-x1' oder '-x': Wenn diese Option aktiviert ist, wird LhA beim Extrahieren und Archivieren Pfadnamen berücksichtigen und beibehalten. Beim Extrahieren wird LhA die noch nicht existierenden Verzeichnisse anlegen. Benutzen Sie diese Option, wenn Sie eine Verzeichnisstruktur beibehalten wollen. Diese Option wird automatisch aktiviert, wenn die Option '-r' benutzt wird.

'-x2': Dieser Modus ist nur beim Extrahieren nützlich. Bei der Bestimmung der zu extrahierenden Dateien wird LhA den vollen Pfad berücksichtigen, ihn beim Extrahieren selbst aber verwerfen. Nützlich, wenn verschiedene Dateien mit demselben Namen (aber unterschiedlichen Pfaden) im Archiv existieren.

'-x3': Dieser Modus ist das Gegenteil von '-x2'. LhA ignoriert Pfade bei der Auswahl der zu extrahierenden Dateien, aber berücksichtigt sie beim Extrahieren. Nützlich, wenn Sie zu faul sind, sich an den genauen Namen inklusive Pfad zu erinnern.

BEISPIEL

'LhA -x2 e dl:rexx.lzh examples/Main.c ram:' wird die Datei 'examples/Main.c' aus dem Archiv nach 'ram:Main.c' entpacken.

'LhA -x3 e dl:src.lzh #?Main#? ram:' wird alle Dateien, die 'Main' im Namen enthalten. Beachten Sie, daß dies nicht gleichbedeutend mit 'LhA x dl:src.lzh #?Main#? ram:' ist, da letzteres auch Dateien wie 'dir/Maindir/file1.h' entpacken würde.

Diese Option ist per Default auf '-x1' gesetzt.

1.98 `-X' (ALL) Keine Namenserweiterung

Wenn diese Option aktiviert ist, wird LhA keine '.lzh'- oder '.lha'-Erweiterung an den angegebenen Archivnamen anhängen. Das Default-Verhalten ist, eine '.lha'- oder '.lzh'-Erweiterung (je nach gewählter Kompressionsmethode) anzuhängen, wenn der Name nicht bereits eine Erweiterung besitzt.

Diese Option ist per Default DEAKTIVIERT (Erweiterungen werden angehängt).

1.99 `-y' (all) Namenserweiterung immer anhängen

Wenn diese Option aktiviert ist, wird LhA immer ein `.lzh' oder `.lha' an den Archivnamen anhängen, selbst wenn der Archivname bereits eine Erweiterung besitzt.

Diese Option ist per Default DEAKTIVIERT (eine Erweiterung wird nur angehängt, wenn nicht bereits eine im Archivnamen vorhanden ist).

1.100 `-Y' (add) Speichere große Dateien nach Rate

Wenn diese Option aktiviert ist, wird LhA große Dateien (>32KB) ohne Kompression abspeichern, wenn die Kompressionsrate kleiner als 3% ist. Dies geschieht, weil die Dekompression solcher Dateien auf langsamen Maschinen lange dauern kann.

Diese Option ist per Default DEAKTIVIERT (alle Dateien werden komprimiert).

1.101 `-z' (add) Dateien nicht komprimieren

Wenn diese Option aktiviert ist, wird LhA alle aktualisierten oder hinzugefügten Dateien im Archiv abspeichern, ohne zu versuchen, sie zu komprimieren. Nützlich, um schnelle Backups zu machen, bei denen die Archivgröße nicht wichtig ist. Es wird nicht empfohlen, diese Option zu benutzen, wenn Archive für die Distribution per Modem oder Netzwerk vorbereitet werden, da die Archive sehr viel größer sein werden als mit Kompression.

BEISPIEL
`lha -z a foo.lha *.bmp' wird alle Dateien im aktuellen Verzeichnis mit der Namenserweiterung `.bmp' im Archiv `foo.lha' archivieren, ohne sie zu komprimieren.

Diese Option ist per Default DEAKTIVIERT.

1.102 `-Z' (add) Archive komprimieren

Ist diese Option aktiviert, versucht LhA, auch bereits komprimierte Dateien zu komprimieren.

Per Default wird LhA nicht versuchen, Dateien zu komprimieren, die bereits komprimiert sind (typischerweise Archiv- oder Bilddateien im GIF oder JPEG Format). Der Dateityp wird anhand der Namenserweiterung bestimmt, und Dateien, deren Namen auf `.lzh', `.lha', `.zoo', `.zip', `arj', `.arc', `.dms', `.wrp', `.lhw', `.zap', `.pak', `.pp', `.gif', oder `.jpg' werden unkomprimiert gespeichert.

Der Grund, aus dem bereits komprimierte Dateien nicht mehr komprimiert

werden sollten, liegt darin, daß die Anzahl der so gewonnenen Bytes so klein ist, daß sich die zum Komprimieren/Dekomprimieren aufgewendete Zeit nicht lohnt.

Diese Option ist per Default DEAKTIVIERT.

1.103 `-0' (add) Benutze LhArc 1.x-Kompression

Diese Option veranlaßt LhA, die alte -lh1- Kompressionsmethode beim aktualisieren von Archiven zu verwenden. Diese Kompressionsmethode ist etwas schneller als die normale -lh5- Kompression, hat aber eine schlechtere Kompressionsrate und ist beim Dekomprimieren sehr viel langsamer.

Wenn diese Kompressionsmethode verwendet wird, wird LhA beim Erstellen von Archiven per Default die Namenserweiterung '.lzh' benutzen.

Wenn diese Option angegeben wird, wird Option '-2' automatisch deaktiviert.

Per Default wird die -lh5- Kompression verwendet.

1.104 `-2' (add) Benutze LhA Kompression (-lh5-)

Diese Option veranlaßt LhA, die -lh5- Kompressionsmethode beim Aktualisieren von Archiven zu verwenden. Diese Kompressionsmethode ist etwas langsamer als die alte -lh1- Kompression, hat aber eine bessere Kompressionsrate und ist sehr viel schneller zu dekomprimieren.

Wenn diese Kompressionsmethode verwendet wird, wird LhA beim Erstellen von Archiven per Default die Namenserweiterung '.lha' benutzen.

Wenn diese Option angegeben wird, wird Option '-0' automatisch deaktiviert.

Dies ist die per Default verwendete Kompressionsmethode.

1.105 `-3' (add) Benutze LhA Kompression (-lh6-)

Diese Option veranlaßt LhA, die neue -lh6- Kompressionsmethode beim Aktualisieren von Archiven zu verwenden. Diese Kompressionsmethode ist etwas langsamer als die alte -lh5- Kompression, hat aber eine sehr viel bessere Kompressionsrate und ist schnell zu dekomprimieren.

Wenn diese Kompressionsmethode verwendet wird, wird LhA beim Erstellen von Archiven per Default die Namenserweiterung '.lha' benutzen.

Wenn diese Option angegeben wird, wird Option '-0' automatisch deaktiviert.

1.106 `-Qa' (all) Benutze einfache Konsolen-I/O

Wenn diese Option aktiviert ist, wird LhA nicht versuchen, irgendwelche Tricks durchzuführen, wie die Größe des Konsolenfensters zu bestimmen, oder den Cursor abzuschalten oder neu zu positionieren. Die Aktivierung dieser Option schaltet auch die Byte-Fortschrittsanzeige ab (wie mit `-n'), da dies Neupositionierung des Cursors erfordert.

Diese Option ist per Default DEAKTIVIERT.

1.107 `-Qb' (ext) Archiv vor dem Extrahieren testen

Wenn diese Option aktiviert ist, wird LhA die Integrität eines Archivs vor dem Extrahieren testen. Wenn das Archiv den Integritätstest nicht besteht, wird aus dem Archiv überhaupt nicht extrahiert. Nützlich in einigen FIDO BBS Setups.

Diese Option ist per Default DEAKTIVIERT.

1.108 `-Qd' (ext) Autoshow-Dateien löschen

(`d'elete autoshow files)
Wenn diese Option aktiviert ist, wird LhA Autoshow-Dateien nach der Anzeige automatisch löschen.

Diese Option ist per Default DEAKTIVIERT.

1.109 `-Qh' (add) Größe des Huffman-Puffers setzen

Diese Option kann benutzt werden, um die Größe des Puffers zu bestimmen, der bei der LHA-Kompression (Default, oder mit den Optionen `-2' oder `-1' ausgewählt) zum Sammeln von Statistiken benutzt wird. Die Größe dieses Puffers beeinflußt die Kompressionsrate auf unvorhersehbare Weise (man kann nicht mit Sicherheit sagen, ob ein großer Puffer besser oder schlechter sein wird). Lassen Sie diesen Wert im Allgemeinen auf dem Default, aber wenn Sie homogene Daten mit einer relativ gleichen Verteilung von Symbolen (wie Textdateien), wird ein großer Wert die Kompression verbessern. Binärdateien werden am Besten mit den Defaultwerten komprimiert.

Der Huffman-Puffer kann jede Größe zwischen 4K und 64K sein, und muß direkt nach dem `-Qh' in Kilobytes angegeben werden.

BEISPIEL

`LhA -Qh32 -2 a foo.lha *' wird alle Dateien im aktuellen Verzeichnis mit einem Huffman-Puffer von 32768 (32K) Bytes archivieren.

`LhA -Qh4 -2 a foo.lha *' wird alle Dateien im aktuellen Verzeichnis

mit einem Huffman-Puffer von 4096 (4K) Bytes archivieren.

Die Default-Größe für den Huffman-Puffer ist 16K.

1.110 `-Qm' (all) Dateinamen in Fortschrittanzeige kappen

(filename 'm'unching)
Diese Option veranlaßt, das alle Ausgaben von Pfaden/Dateinamen so gekappt werden, daß sie in die Breite des Konsolenfensters passen; so werden die Zeilen der Fortschrittanzeige nicht umgebrochen. Pfadteile werden je nach Bedarf gekappt, um die Zahl der auf der Konsole ausgegebenen Zeichen zu reduzieren, wobei die entfernten Teile mit einer Elipse ('...') ersetzt werden, um anzuzeigen, das hier gekappt wurde.

Nur vollständige Pfadteile werden gekappt, keine Bruchstücke ('path/longnode/file' wird nicht in 'path/long.../file' umgewandelt, sondern in 'path/.../file'), und es wird nur eine Elipse ('...') angezeigt, unabhängig von der Zahl der Pfadteile, die gekappt wurden.

Wenn der vollständige Pfad bis auf den Dateinamen gekappt wurde, und es _immer_ noch nicht paßt, wird der Dateiname ausgedruckt, und ein Umbruch erlaubt.

BEISPIEL

Wenn ein Archiv folgenden Pfad enthält:

```
long/path/with/many/levels/of/subdirectories/and/a/file
```

und Sie nur 40 Zeichen zur Verfügung haben, um den Pfad anzuzeigen, wird er angezeigt als:

```
long/path/with/many/levels/of/.../file
```

NOTE

Diese Option wird für die '-D4' Fortschrittanzeige IMMER aktiviert, damit Pfadnamen in das Fortschrittfenster passen.

Diese Option ist per Default DEAKTIVIERT.

1.111 `-Qn' (all) Nationalen Zeichenmodus aktivieren

Wenn diese Option aktiviert ist, wird LhA nationale Zeichen korrekt in Groß-/Kleinschrift wandeln. Per Default wird LhA keins der oberen 127 ASCII-Zeichen umwandeln, da ältere (<2.1) Filesysteme nationale Zeichen beim Berechnen der Hash-Werte nicht korrekt behandeln. Diese Option sollte benutzt werden, wenn nationale Filesysteme benutzt werden (NOFS/NFFS).

Diese Option ist per Default DEAKTIVIERT.

1.112 `-Qo' (all) Optionen nach Kommando ignorieren

Wenn diese Option aktiviert ist, wird LhA die Kommandozeile nach dem Archivnamen nicht nach Optionen durchsuchen. Diese Option ist nützlich, wenn Sie Dateien definieren wollen, deren Namen mit '-' beginnt.

Diese Option ist per Default DEAKTIVIERT.

1.113 `-Qp' (move) Ignoriere Löschen-Schutzbit

Wenn Sie diese Option aktivieren, wird LhA beim Verschieben-Kommando ('m') auch Dateien mit nicht gesetztem Löschen-Schutzbit löschen.

Diese Option ist per Default DEAKTIVIERT (löschgeschützte Dateien werden nicht gelöscht).

1.114 `-Qq' (add) Schnelles Hinzufügen

('q'uick add)
Wenn diese Option aktiviert ist, wird LhA das Archiv nicht nach doppelten Dateien durchsuchen, bevor Dateien hinzugefügt werden. Das kann nützlich sein, wenn eine einzelne Datei zu einem großen Archiv hinzugefügt wird, und man weiß, daß das Archiv keine Datei desselben Namens enthält (wie es bei einigen FIDO BBS Setups der Fall ist).

Diese Option ist per Default DEAKTIVIERT.

1.115 `-Qr' (add) Dateidatum ignorieren

Wenn aktiviert, deaktiviert diese Option die Überprüfung des Dateidatums bei den Kommandos Aktualisieren ('u') und Auffrischen ('f'), so daß die Dateien, die im Archiv bereits existieren, ohne Rücksicht auf das Dateidatum ersetzt werden.

Diese Option ist per Default DEAKTIVIERT für alle Kommandos außer 'r'.

1.116 `-Qv' (all) Setze Laufwerke für Mehr-Medien-Archive

(set 'v'olumes)
Mit dieser Option können Sie LhA veranlassen, verschiedene Laufwerke bei der Erstellung von Mehr-Medien-Archiven zu verwenden. LhA wird die Laufwerke in der angegebenen Reihenfolge benutzen, und nach dem letzten Laufwerk wieder mit dem ersten beginnen. Die Laufwerke sollten OHNE Doppelpunkt direkt nach dem '-Qv' angegeben werden, getrennt durch Kommas (','). Wenn Sie diese Option benutzen, müssen Sie immer noch das erste zu benutzende Laufwerk im Archivnamen angeben.

BEISPIEL

'LhA -Va -Qvdf0,df2,df3 -r a df0:Bak hd:#?' wird ein Mehr-Medien-Archiv erstellen, beginnend mit df0:, und danach df2:, df3:, df0:, df2: und so fort benutzend. Beachten Sie, daß Sie immer noch 'df0:' in der Definition des Archivnamens angeben müssen.

1.117 `-Qw' (all) Keine Namensmuster

(no 'w'ildcards)
Wenn Sie diese Option angeben, wird LhA keine Namensmustererkennung durchführen. Das ist nützlich, wenn Sie (illegale) Dateinamen mit Musterzeichen ('()#?~%|*') angeben wollen.

Diese Option ist per Default DEAKTIVIERT.

1.118 Autoshow-Dateien

Autoshow-Dateien sind Dateien, die dem User beim Entpacken der Datei aus einem Archiv automatisch angezeigt werden. LhA entscheidet, ob eine Datei angezeigt werden soll, anhand des Dateinamens; wenn der Dateiname mit '.displayme' endet, wird die Datei angezeigt, sofern Autoshow-Dateien nicht deaktiviert wurden (durch die '-M' Option). Abgesehen davon, daß sie auf dem Bildschirm angezeigt werden, werden Autoshow-Dateien genau wie normale Dateien entpackt, ohne den '.displayme'-Teil zu entfernen.

1.119 Residentfähigkeit

LhA ist Multitasking-reentrant und pure, und kann durch die üblichen Shell Resident-Kommandos resident gemacht werden - 'resident' unter AmigaOS Shell, und 'resi' unter WShell. Wenn Sie eine andere Shell benutzen, suchen Sie im Handbuch Ihrer Shell nach Informationen, wie man Programme resident macht.

1.120 Mehr-Medien-Archive

Mehr-Medien-Archive werden einfach dadurch erzeugt, daß größere Archive
in kleinere Dateien aufgeteilt werden.

2.8.1 Mehr-Medien-Dateinamen

2.8.2 Backups mit Mehr-Medien-Archiven

2.8.3 Extrahieren von Mehr-Medien-Archiven

1.121 Mehr-Medien-Dateinamen

Die erste Datei eines Mehr-Medien-Archivs heißt 'name.lha' oder 'name.lzh'. Die folgenden Medien heißen 'name.l01', 'name.l02' usw.. Diese Namenskonvention wurde gewählt, weil bestimmte schwachsinnige Filesysteme keine längeren Dateinamen erlauben (MSDOS). Mehr-Medien-Archive mit mehr als 100 Medien werden im Moment nicht unterstützt.

1.122 Backups mit Mehr-Medien-Archiven

Die Mehr-Medien-Fähigkeiten von LhA können dazu eingesetzt werden, effiziente Festplatten-Backups zu erstellen. Dazu benötigen Sie einige formatierte Disketten (oder ähnliches) - LhA unterstützt momentan noch kein Formatieren beim Beschreiben der Medien. Ein Besipiel für ein Backup-Kommando wäre:

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

Dies würde alle Dateien archivieren, die sich im Verzeichnis / Laufwerk 'LhA:' befinden, und auf Diskette speichern, beginnend mit Laufwerk DF0:, dann DF2: und schließlich DF3:. LhA würde diesen Zyklus wiederholen, bis das Backup abgeschlossen ist.

LhA ist etwas langsamer als ein spezialisiertes Backup-Programm, das es über das Filesystem gehen muß, statt direkt auf die Diskette zu schreiben. Allerdings bietet LhA eine bessere Kompression als jedes exitsierende Backup-Programm.

1.123 Inkrementierende Backups

Inkrementierende Backups sind Backups, bei denen nur die Dateien gesichert werden, die sich seit dem letzten Backup verändert haben. In LhA kann dies wie folgt erreicht werden:

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

Dies würde alle Dateien in 'Work:', deren 'a'-Schutzbit ('A'rchiviert) nicht gesetzt ist, archivieren (durch die -s Option). Nach dem

Hinzufügen einer Datei setzt LhA das 'a'-Bit der Datei (die -S Option). Wann immer auf diese Datei schreibend zugegriffen wird, löscht AmigaDOS automatisch dieses Bit, so daß die Datei beim nächsten inkrementierenden Backup wieder enthalten wäre.

1.124 Extrahieren von Mehr-Medien-Archiven

Beim Extrahieren von Dateien aus Mehr-Medien-Archiven muß LhA das gesamte Archiv vom ersten bis zum letzten Medium scannen. Ein Beispielkommando wäre:

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

Dies würde alle '.c'-Dateien (#?.c) aus dem Mehr-Medien-Archiv (-V) 'MltArc.lha' extrahieren, wobei abwechselnd auf Laufwerk DF0: und DF1: zugegriffen wird (-Qvdf0,df1).

2.8.3.1 Inkrementierende Backups wiederherstellen

1.125 Inkrementierende Backups wiederherstellen

Inkrementierende Backups sollten beginnend mit dem letzten Backup wiederhergestellt werden (d.h. das neueste Archiv sollte zuerst entpackt werden). Ein Beispiel wäre:

```
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:
```

Die '-T' Option muß angegeben werden, damit LhA nicht versucht, Dateien zu überschreiben, die neuer als jene innerhalb des Archives sind (da diese schon aus neueren Archiven extrahiert wurden).

1.126 Inhalt von Mehr-Medien-Archiven auflisten

Mehr-Medien-Archive werden wie folgt aufgelistet:

```
LhA -V v df0:MyArc
```

Dies wird alle Dateien in dem Mehr-Medien-Archiv auflisten, beginnend mit der Datei 'MyArc.lha'. Am Ende jedes Mediums wird LhA nach einem neuen Medium fragen, bis das Ende des Archives erreicht ist. Listings von individuellen Medien werden in der jetzigen Implementierung nicht unterstützt.

1.127 Mehr-Medien-Archive aktualisieren

In der aktuellen Version ist es nicht möglich, Dateien in einem Mehr-Medien-Archiv zu löschen, aufzufrischen oder zu aktualisieren.

1.128 Mehr-Medien-Archivierung unterbrechen

Unterbrechen Sie Mehr-Medien-Archivierung nicht.

Momentan bedeutet die Unterbrechung einer Archivoperation, daß das Archiv leicht durcheinander sein wird. Alle Daten werden in Ordnung sein, aber Sie werden keine Dateien mehr hinzufügen können, weil LhA Sie um ein nicht existierendes Medium bitten wird, das das Ende des Archivs beinhalten soll. Dies läßt sich in der jetzigen Implementierung nicht verhindern.

1.129 Ein Wort zu Archivköpfen

Ein 'Archivkopf' muß für jede Datei in das Archiv geschrieben werden, damit der Archivierer weiß, wie die Dateien heißen, wie sie komprimiert wurden usw.. Das originale LhArc hatte ein sehr primitives Layout für Archivköpfe, und bot keinen guten Weg, um plattformspezifische Info wie Dateikommentare zu speichern (ich habe ein Workaround in LhArcA 0.99 erarbeitet, indem ich den Dateikommentar im Dateinamensfeld abgelegt habe -LhArc und LZ übernahmen später diese Methode). Im **IX LhArc V1.02 führten die Autoren einen neuen Archivkopf ein (Archivkopf Stufe 1), der etwas mehr Info zu speichern erlaubte, aber die Kopflänge war immer noch auf 255 Bytes beschränkt. In LHA 2.13 für MS-DOS wurde ein neuer Archivkopf eingeführt - Stufe 2. Mit diesem neuen Archivkopf-Typ konnte eine beliebige Menge an Informationen gespeichert werden. LhA kann all diese Archivkopftypen sowohl lesen als auch schreiben. Um auszuwählen, welcher Archivkopftyp geschrieben werden soll, benutzen Sie die '-H' Option. LHA für MSDOS und LHa für **IX schreiben per Vorgabe Archivköpfe der Stufe 1. LhA benutzt aus Kompatibilitätsgründen Stufe-0-Köpfe (LZ und LhArc können nicht korrekt mit den Stufen 1 und 2 umgehen). Wenn Sie wissen wollen, welchen Archivkopftyp ein Archiv enthält, benutzen Sie das 'vv' Kommando.

1.130 Einige Tips zum effektiven Archivieren

Wenn Sie vorhaben, eine große Menge von ähnlichen oder kleinen Dateien zu archivieren - z.B. Textdateien - können Sie die Kompressionsrate dadurch erheblich erhöhen, indem Sie zunächst ein Archiv OHNE Kompression erzeugen (durch Benutzung der '-z' Option), und anschließend diese Datei MIT Kompression in ein Archiv packen. Zum Beispiel habe ich ein großes Verzeichnis mit verschiedenen Quellcodes und einigen Binaries (insgesamt 2480 Dateien, 5102117 Bytes) auf diese Weise archiviert, mit:

```
LhA -z -r a hd:test msrc:
```

und dann komprimiert mit

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

Das endgültige Archiv 'hd:msrc.lha' war 1545076 Bytes groß. Wenn auf dem normalen Weg komprimiert ('LhA -r -Qh64 a hd:msrc msrc:'), war das Archiv 2114777 Bytes lang. Ein ziemlicher Unterschied...

1.131 So wenig Speicher wie möglich verbrauchen

Wenn Sie die Voreinstellungen verwenden, benötigt LhA etwa 300KB zum Archivieren, und 180KB zum extrahieren von Dateien. Um dies auf ein Minimum zu senken, können Sie die Größe des I/O-Puffers auf 8K reduzieren. Dies wird etwa 48K beim archivieren und mindestens 24K beim extrahieren einsparen. Sie können den Gebrauch von Speicher beim Archivieren noch weiter reduzieren, indem Sie die Größe des Huffman-Puffers auf 4K senken, aber das ist nicht ratsam, da die Kompressionsrate erheblich sinken wird. Bitte nehmen Sie zur Kenntnis daß die obigen Angaben den Speicherverbrauch inklusive Stack und Programmcode angeben.

1.132 Voll MS-DOS-kompatible Archive herstellen

Um MSDOS Archivierer zufriedenzustellen, werden Sie einige Amiga-spezifische Features abschalten müssen. Dateikommentare werden unter MSDOS nicht unterstützt, und deshalb sollte deren Archivierung mit der '-f' Option abgeschaltet werden. Zusätzlich sollten Sie die Speicherung von Dateiattributen mit der '-a' Option ausschalten. Autoshow-Dateien werden von MSDOS LHA 2.13 nicht unterstützt. Wenn Sie Archivköpfe der Stufen 1 und 2 benutzen, brauchen Sie sich keine Sorgen über die Abschaltung der Dateiattribute machen. LHA 2.13 für MSDOS und LHa 0.04 für **IX erstellen per Vorgabe Stufe 1 Archivköpfe.

Zur Zusammenfassung, benutzen Sie die folgenden Optionen, um Archive zu erstellen, die mit MSDOS LHA bearbeitet werden sollen:

```
'-UH0a0f'
```

Um Archive zu erstellen, die mit LhArc dearchiviert werden können, benutzen Sie die folgenden Optionen:

```
'-H0 -0'
```

und für MS-DOS LHarc:

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

1.133 Dateien aus beschädigten Archiven retten

Es ist niemals möglich, alle verlorenen Daten aus einem korrupten Archiv zu retten, aber Sie können so viel Daten wie möglich zurückerhalten, wenn Sie die '-k' Option und einen kleinen I/O-Puffer (8K) verwenden. Ein Beispiel wäre:

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

Dies würde so viel wie möglich aus dem korrupten Archiv nach 'ram:' extrahieren.

1.134 Danksagungen

(Stefans originale Danksagungen)

Haruyasu Yoshizaki	Für das Veröffentlichen der Quellcodes des originalen LHA für MSDOS. Der Quellcode wurde beim Schreiben dieses Programms als Referenz benutzt. Es wurde kein Code aus diesem Quellcode kopiert, vielmehr wurde LhA von Grund auf neu für den Amiga geschrieben.
Haruhiko Okumura	Für die Entwicklung der -lh5- und -lh4- Kompressionsalgorithmen, und für das Veröffentlichen der C-Quellcodes derselben als Public Domain. Diese Quellen wurden als Referenz beim Schreiben der 680x0 Assembler-Versionen des Kompressionscodes verwendet. Manche Algorithmen wurden durch meine eigenen, schnelleren ersetzt, aber die Ideen sind die Gleichen.
Robert K.Jung	Für die Entwicklung des mit Funktionen überladenen ARJ für MSDOS, dem verschiedene Ideen für Kommandos und Funktionen von LhA entnommen wurden.
Paolo Zibetti	Für die Entwicklung des ersten Archivierer im LhArc-Stil für den Amiga, der mein Interesse an Dateiarchivierern und fortgeschrittenen Techniken zur Datenkomprimierung weckte.
Roger Nordin	Beta-Tester extraordinaire
Ron Birk	Dafür, daß er die von mir benötigten Quellcodes ausgrub, bevor ich selbst Zugriff auf das InterNet erhielt - Danke!
Martin Olsson	Dafür, daß er mir den Quellcode für LhA V2.11 zur Verfügung stellte, den ich als Referenz verwendete. (Ich habe die -lh5- Dekompression ausschließlich anhand des verfügbaren 80x86 Quellcode geschrieben... harte Arbeit!)
LhArcA User	Vielen Dank an Euch alle, die sich für LhArcA und LhA registriert haben, bevor die Programme überhaupt fertiggestellt waren (LhArcA wurde

niemals fertig, aber die, die sich registriert haben, werden LhI/LhA erhalten, sobalt es fertiggestellt ist).

LhA User — Vielen Dank an alle, die sich bis jetzt registriert haben, und noch größeren Dank an alle, die Fehler und Probleme mit vergangenen Versionen berichtet haben - ohne Euch wäre dieses Programm niemals, was es jetzt ist.

Das Programm wurde mit Hilfe des Lattice C Compilers und Assembler auf einem 25MHz Amiga 3000 entwickelt. Großartiger Compiler, großartiger Computer! Zusätzlich wurden RCS und MKID benutzt, um die Pflege und den Entwicklungsprozeß erheblich zu vereinfachen.

"Infinities of dreams imploding into one ..."

(Jim Coopers Danksagungen)

Stefan Boberg — Ohne den ich niemals dieses Programm zum herumspielen hätte.

David Tritscher — Der seine Magie auf den Kompressionscode wirkte, und der mir allgemein mit anderen Dingen aushalf, die benötigt wurden.

Martin Baute — Für das Korrekturlesen und die Übersetzung des .guide ins Deutsche.

Diese Dokumentation war ursprünglich von Stefan mit einer Version von 'proff' formatiert, wurde aber von mir "von Hand" in eine .guide-Datei konvertiert.

Diese Version von LhA wurde auf einem A3500 (einer Vorläuferversion des A3000T) mit installiertem PowerUp-Board (060-50MHz/604e-200MHz) (Jim) und einem A1200 (David) entwickelt, mit SAS/C 6.5x zum compilieren und CPR (Jim) und MonAm (David) zum Debuggen.

Die Übersetzung erfolgte auf einem A1200 mit PowerUp-Board (060-50MHz/603e-240MHz) und der unschätzbaren Hilfe von GoldED Studio v5.1.5. (Martin)

1.135 Geschichte

2.1 Jim Cooper & David Tritscher

Probleme mit -Qm & zu schmalen Fenstern behoben. (Jim)

Abschließende Säuberungen, usw. vor neuem Aminet-Release. (Jim)

Website eingerichtet - lha.warped.com - simpel, aber funktioniert. (Jim)

AMINET RELEASE

2.0 Jim Cooper & David Tritscher

Dateinamen-'munging' funktioniert nun für alle Fortschritt-anzeigen. (Jim)

Benutzt private Amiga Escape-Sequenzen (aWSR, aWBR) um die Konsolengröße zu erhalten, statt ACTION_DISK_INFO Paket. Blockiert keine /AUTO Fenster mehr. (Jim)

Dokumentation bereinigt, neue Funktionen dokumentiert, etc. (Jim)

NEUES RELEASE

1.110 Jim Cooper & David Tritscher

Ein weiterer kleiner Gewinn bei der Kompression. (David)

'-D4' Fortschrittanzeige hinzugefügt. (Jim)

'-1' (-lh4-) für Kompression deaktiviert, da das Format alt, langsam und ziemlich wertlos ist. -lh4- bei Dekompression weiter unterstützt, zur Kompatibilität mit älteren Archiven. (Jim & David)

Vorgabestatus für einige Optionen geändert, um die Funktion der anderer LhA-Versionen (UNIX & PC) anzugleichen. (Jim)

Die Art geändert, in der 'inverse video' abgeschaltet wird - statt die Konsole einfach nach 'normal' zu resetten, wird der spezielle 'inverse off' Code verwendet. Sollte denen helfen, die ihre Konsole normalerweise in nicht-Vorgabe-Modi verwenden (z.B. andere Farben etc.). (Jim)

BETA RELEASE

1.100 Jim Cooper & David Tritscher

Kompressionsalgorithmus geändert - komprimiert jetzt besser als das alte LhA, und schneller! (David)

Kompressionsformat '-lh6-' hinzugefügt, um mit LhA für UNIX gleich-zuziehen. (David)

Kommandozeilenunterstützung für '-lh6-'-Verwendung hinzugefügt. (Jim & David)

Uralten Fehler mit korrupten Teilarchiven behoben (ließ LhA 1.3x, 1.5x usw. abstürzen). (Jim)

Verschiedene Löcher in altem Code gestopft, die zu Abstürzen hätten führen können. (Jim & David)

Dokumentation, Verwendung etc. aktualisiert. (Jim)

BETA RELEASE

1.99 Jim Cooper & David Tritscher

```
    Verbliebener ASM-Code nach C konvertiert. Beginn der Arbeit an
    Veränderungen, um den Code schneller zu machen, alte (& neue :-)
    Fehler behoben, und neue Funktionen hinzugefügt.

    NUR INTERN

 1.98   Jim Cooper

    Den "funktioniert nicht mit Leerzeichen in Namen" Fehler erschlagen,
    den einige Leute bemerkt haben.

    Direkt an einige Leute geschickt, die Fehler berichtet haben, die
    ich in früheren Einträgen als "behoben" bezeichnet habe, da ich
    diese Version nicht ins Aminet hochladen konnte.

 1.97   Jim Cooper

    INTERN

 1.96   Jim Cooper
    <seufz> Zu früh, nicht genug Tests, etc... Der letzte "Fix" hat die
    ^C-Behandlung außer Gefecht gesetzt. Behoben.

 1.95   Jim Cooper
    Uh-Ups. Ausgabepuffer für "Overwrite..." Nachricht nicht geleert.

 1.94   Jim Cooper
    Erste Aminet-Release nach der Übernahme von Stefan.

 <1.94  Stefan Boberg
    Mit vielem Dank von der Amiga-Gemeinschaft.
```

1.136 TODO

```
(In keiner besonderen Reihenfolge.)

- Fortschrittanzeige für 'stored' Dateien. Stefans Code tat es niemals.

- vom momentan gepufferten I/O auf asynchrones I/O umstellen.
- "-Q" Option hinzufügen, um asynchrones I/O auszuschalten, aus welchem
  Grund auch immer. :-)

- Unterstützung für Wechsel des Kompressionsformats mit dem
  'y'-Kommando.

- Unterstützung für Hard-/Softlinks.

- Multi-Medien-Archive besser dokumentieren.
- Änderung des Medienwechsel-Prompts.

- -e sollte '-lhd-'-Einträge für ALLE Verzeichnisse erzeugen, um
  Erstellungsdatum und Attribute zu speichern.

- Kompression weiter verbessern.
```

- Ports zu verschiedenen anderen 'Plattformen', inklusive Windoze, PowerUp, WarpOS,...

- Lokalisierung der LhA-Ausgaben. Viele Leute sprechen entweder kein Englisch, oder würden sich mit Ausgaben in ihrer eigenen Sprache einfach wohler fühlen.

- Weitere übersetzte Versionen der Dokumentation.

- Abbruch-Button für Grafik-Fortschritt... vielleicht. ^C funktioniert immer noch gut für den Hauptprozeß, aber das ist für GUI-Anwender nicht offensichtlich.

- Geschwindigkeit, Tricks, Rest des Codes bereinigen, usw. usw. usw.