A Directory Opus LHA Display Utility

by Merrill Callaway

Getting more from Directory Opus with an ARexx Utility

This month's program is a handy and ingenious ARexx macro for Directory Opus (DOpus) that allows you to view the contents of LHA or LZH compressed archives and select which files/directories you want to decompress, using only mouse clicks. I am not the author of the macro, but I did correct a few errors in the version given me by a friend who downloaded the original from a BBS. The listing shows where I modified the original. I also cleaned up the format and coding a little and put the ARexx instructions in caps for readability.

It would be instructive to look at this macro as a good example of a really clever use of ARexx in a DOpus macro and also for learning how to improve ARexx scripts you may find on a BBS. The file is named LhArc_ext.dopus ver 1.11 by KjPetlig. For proper documentation, I call my corrected version listed here as $VER: 1.12, since I modified it enough to warrant a different version number. Note the syntax of adding a version information to your ARexx programs. On a command line

```
versionREXX:LhArc_ext.dopus
```

will display ''1.12'' if the above syntax is anywhere in the file. The listing shows the history of the program, and even includes a Help file for appending to the Directory Opus Help file, DirectoryOpus.HLP in your S: directory.

The syntax for a DOpus Help file entry is to put an asterisk next to the button or gadget name on one line. Then put in as many lines of help as you need and finish off with a ''^'' character at the end. Help entries are all stored in an ordinary text file. I always write a help file for my ARexx DOpus scripts because I do not always remember what they do months later. I like the idea of including a ready made help file in the program itself. By the way, if you are writing ARexx programs that operate from the CLI or Shell, you may include help in the program file and test for an argument character of ''?''. If the ARexx program has the argument ''?'', then the program would display its own help file or template. This follows AmigaDOS format.

For instance, if you have a program called DOIT.rexx and from a command line you type:

```
rxdoit?
```

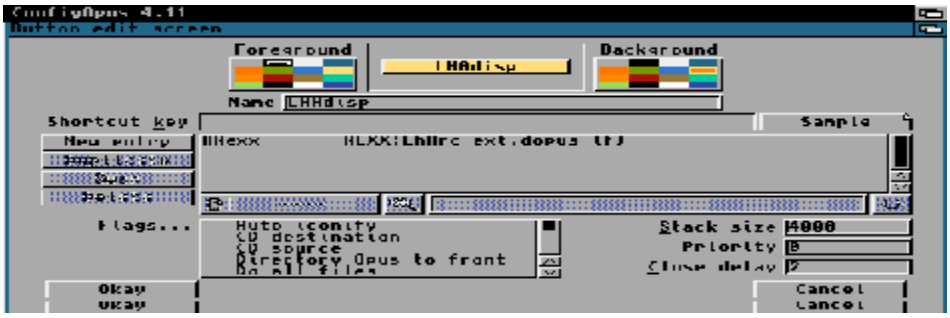
then if the first part of the program is coded,

```
/*DOIT.rexxwithhelp*/
ARGhelp

IFhelp='?'THENDO
/*displayhelplines...*/
SAY'helpline1...'
SAY'templatemaybe...'
SAY'etc...'
END
/*restofprogram*/
```

I changed the name of the DOpus button to LHAdisp instead of ''LhArc_ext'' because I thought that it was more mnemonic. Whatever you name your button, you want to set it up as an ARexx program in the DOpus configuration window. Use no flags and put in the DOpus line argument {f} after the program name to get the selected archive as an argument for the program. The operation is simple. Select an archive ending in ''.LHA'' or ''.LZH'' in either DOpus window. Click on the button named LHAdisp and a list of all files and directories in the archive will appear in the same window as the selected archive. You may penetrate subdirectories by selecting a directory name and clicking on the button again. The directory will be expanded. Note: you cannot travel back up the archive tree by pressing parent as you ordinarily would.

The program uses the same window that was there before and removes the names of its entries (but not the files they represent!), so if you click on parent, you get the parent of the directory whose entries were removed. What you see is the contents of a temporary file T:REXXtemp for holding the archive names. The LHA program itself has options for displaying the contents of an archive, and ARexx has the ability to execute AmigaDOS commands. The author took this one step further, displaying the results of the special LHA call in a DOpus window. There are of course further changes some of you may want to make. For instance, you may want to put the



ConfigOpus
A handy and ingenious ARexx macro for Directory Opus (DOpus) that allows you to view the contents of LHA or LZH compressed archives and select which files/directories you want to decompress, using only mouse clicks.

contents of the archive's file table into a separate buffer instead of removing the file entries from the current buffer. If you were viewing a lot of archives, this would let you keep each archive list in a separate buffer.

The way the program is written now it displays the file/directory contents of one archive at a time in the active window. If you choose not to extract anything and rescan the source window, the file names in the directory that was there originally reappear. If you decide to extract one or more of the displayed files, select them and press the same button. The files will be extracted and put into the destination window (the non-active window), and the source window will be rescanned to show the original contents. Extracted files will really be in the destination directory, unlike the archive file names which are only displayed in the source window. Let's take a look at the listing.

Get the Information from DOpus

First, we tell the program where to find LHA, a freely distributable archive utility available from BBSs and on Fred Fish CD. I have my copy in C:, but if you keep LHA somewhere else, change the line to reflect your path. An OPTIONS RESULTS line informs ARexx that we want to have the special variable RESULT assigned a value after any function calls.

The address 'DOPUS.1' instruction is not necessary because the program runs from a DOpus window, and the address is implicit. I commented it out in case you have multiple copies of DOpus running, where the DOPUS.n address needs to be the open document address. The ID_Comment variable is assigned a value '**Extracted_from:' to be attached as a file comment later on. This comment string is how the program can tell whether you are viewing an archive's file names for the first time, or de-archiving selected files:

```
IF (the file spec does NOT contain this string) THEN (generate display of
names); ELSE (extract the selected files).
```

The test is performed by the ARexx function INDEX(). If a string is found inside another one, then INDEX() returns a non-zero number, the offset of the string; otherwise INDEX() returns zero.

DOpus Window Handling

First, however, the program needs to get the window and file information from DOpus. This part was where the author, KjPetlig made some mistakes. The DOpus command STATUS 3 returns the active window: 0 is the left window, 1 is the right window. We let the token, window, take this value (0 or 1). Therefore the destination window, destwindow is the absolute value of (window-1).

The ARexx function ABS() comes in handy here. ''STATUS 13 window'' sets the path to that of the active window, and path is assigned the RESULT. We use virtually the same code to set another path to that of the destination window and assign this result to destpath.

Next we need to get rid of a flaw in DOpus which chokes on the device name with spaces, ''Ram Disk:'' The program tests both path names and adjusts their names accordingly. The LEFT() function looks for a matching ''Ram Disk'' in the first 8 characters. Then a PARSE on a pattern '':'' removes the pattern and assigns the remainder of the path to drest. Then a simple concatenation reassembles our path as ''RAM:''||drest. This fix removed the problem of extracting to the Ram Disk device.

GETNEXTSELECTED is a DOpus command to return the next selected file name in the RESULT variable which gets assigned to ArcNm. Only the first file name is returned. This is always the case unless the entry is deselected by another command. Later in the program, the multiple selections are each deselected during a SELECTFILE command in a loop, so the GETNEXTSELECTED that occurs after it does get the next file in the list.

Now the program tests ArcNm. If it is 0 then there were no selected files, and the program exits after posting a message to this effect in the top bar of the DOpus window (TOPTEXT is the DOpus command to do it). If there is a file selected, another DOpus command, FILEINFO gets the file information from the selected file. The token, FInfo gets this result. Now the program tests FInfo to see if the string represented by ID_Comment is in it. If not, then ID_Index is zero and we are in ''display the archive files'' mode. If ID_Index is non zero then the program control skips to the ELSE DO clause near the bottom to generate a different LHA command string for AmigaDOS to process.

Assume it is the first time through. Then ID_Index is zero. I changed a token here for clarity. The author was using the same token from before, and although that is OK by ARexx, it is not nearly as readable. All the program is doing here is finding out if we have a ''.LHA'' or a ''.LZH'' file. The UPPER() function converts to upper case the RIGHT 4 characters of the file name, ArcNm, and tests them directly. If we don't have a compressed file, then the program again exits and posts a message.

A Quick Fix for a Glitch

If everything is OK so far, we are ready to list the files in the archive, so a TOPTEXT says ''Reading LhArc Archive...''. Next I had to fix what the author calls ''strange things happening''. He did not know how to set the windows in DOpus properly. I used code from another of my programs to perform the first part of the window operations in DOpus, so here I make a connection from my code, which uses the variable ''path'', to Mr. Petlig's which uses the variable ''ArcPath''. I simply assign ArcPath=path. This is easier than changing all his code and possibly making a typo.

I left the code less than perfect to show you how you can quickly fix a program that is not behaving correctly by patching in code you know is working properly. A ''STATUS 6 window'' command returns how many entries are in the active window. The RESULT is the number of entries. Since the very next instruction uses RESULT directly, we can ''Do RESULT'', but this isn't always a good habit! RESULT is reset after every command. You MUST use it immediately if you are to use it directly. So if there are 16 entries in the window, then the program will ''do 16'' iterations. First it uses GETENTRY 1 to get the first entry. Then it uses ''REMOVEFILE result'' to remove the file by name (name is in the result variable). Now we have a clean window to work with. The files are not actually removed, only the display of their names is removed.

LHA as an AmigaDOS Command

The line:

```
ADDRESS COMMAND LhAPath||'LhA>t:RexxTemp vv 'ArcPath||ArcNm
```

makes a table of the files inside the archive. The redirection ''>'' sends the output to a file T:RexxTemp instead of to the screen output, STDOUT. Here typical output:

```
LhA Evaluation V1.32 - Copyright (c) 1991,92 Stefan Boberg.
All rights reserved. Not for commercial use.

Listing of archive 'assignx12a.lzh':
Original Packed Ratio Date  Time  Atts Method CRC  Host OS Lv
-------- ------ ----- ----- ----- ------ ---- ------- -- --
AssignX.info
   4682   2056  56.8% 11-Mar-91 21:30:48 --p- -lh1- 8128 Unknown  0  0
file
  40417  7556  6.6% 06-Apr-91 19:40:56 ---- rwed -lh1- 03FB Unknown  0  0
c
```

```
1885766 59.3% 06-Apr-91 19:46:38 --p-rwed-lh1-3177 UnknownRexq.Docs
286 21325.5% 11-Mar-91 22:33:08 --p-rwed-lh1-1AC6 UnknownRexq.info
468 20256.8% 11-Mar-91 22:43:04 --p-rwed-lh1-8128 UnknownExit.a
457 22650.5% 06-Apr-91 19:52:56 --p-rwed-lh1-4BC9 UnknownRexq
3052 203033.4% 06-Apr-91 19:56:34 --p-rwed-lh1-FB73 UnknownAssignX.c
3148 129158.9% 22-Apr-91 13:37:28 ----rwed-lh1-8870 UnknownAssignX
4352 283634.8% 22-Apr-91 13:37:48 --p-rwed-lh1-E86D Unknown0
AssignX.Docs
3815 199447.7% 22-Apr-91 13:36:06 --p-rwed-lh1-9844 Unknown0
————————————————
18335 993545.8% 20-Aug-92 09:42:00 10 files
```

Operation successful.

The next function is ''CALL OPEN('LhaList','t:RexxTemp','R')'' to open the temporary file for reading. Mr. Petlig made some usage errors here, but two mistakes added up to a program that works! First, he did not quote the name LhaList, so ARexx was changing it to LHALIST. This is a logical name for a file, and it is CASE SENSITIVE! But since he did not quote it anywhere else, ARexx converted unquoted strings to upper case, and there wasn't a problem. However, this could have proved very confusing to beginners, particularly if they quoted 'LhaList'.

The best ARexx form is to 'quote' logical names to call attention to the fact that they are case sensitive even if they are all in caps. I corrected the listing to reflect good form. KjPetlig also failed to CALL the OPEN() function. Functions should always be CALLed, set equal to something, or used in an expression directly. Since DOpus doesn't provide a console unless you explicitly open one, not CALLing OPEN() would go unnoticed and would not cause the program to fail, but again, always use the correct form.

The program reads the open file until it finds 8 hyphens at the left, skipping the heading. Then it reads lines until it again encounters the hyphens to signify the end of the report. In between, it reads a line and then parses out the relevant information. I replaced the author's template targets he called ''garbage'' with periods (.) which are NOT assigned. He had used ''garbage'' for unwanted target tokens, but explicit tokens are still assigned a value, slowing down the program and using memory. We want to parse the file spec into: file size, date, time, and protection bits.

Date Conversions

The program needs to convert from something in the format

```
date=12-Nov-94
```

to the number of seconds since 01 Jan 1978, day zero on the Amiga, in order to write a file spec back into the DOpus window. Mr. Petlig makes clever use of the INDEX() function. He sets up a string that contains all months as abbreviated in the system. Then he found the position of the ''month'' substring, SUBSTR(date,4,3), in the long string. SUBSTR() is the substring of the token, date, starting at position 4 and going for 3 places. I like the parse instruction much better than using SUBSTR(), so I parsed the date string on the patterns, ''-'', assigning day month year to their equivalents from the string. Parsing on a pattern removes the pattern from the string.

At any rate, month = ''Nov'' and has index = 31. Add 2 and get 33. Now we divide by 3 (in the line below) and get Nov as month 11. Very clever! The line below also builds up Date as a concatenation of strings and functions in an expression that evaluates to 19941112 in our example. Another advantage of using parse instead of multiple substrings is that we get the year as well.

Mr. Petlig states in one of his notes that he had had trouble with dates in the next century. A simple test of year lets us set century=20 or century=19. I replaced his code here, too. In our example, date=19941112, the date in ''sort'' format. Seconds is calculated from the ARexx DATE(i,date,s) function. The option is i or days from day zero, date is the sorted form of the date above, reformatted from the temporary table, and s means that the format of the date argument is in sorted or numeric format. It's a bit confusing that in DATE(), i and s may be either options or formats (they are the only two formats, but not the only two options). The syntax of DATE() is DATE(option,date,format).

Note that format must match the format of date: If date is in form YYYYMMDD, then format must be 'S'; if date is in days from day zero format, DDDD, then format must be 'I'. Option determines the format of the returned value, in this case days since day zero. Finally the result in number of days since day zero is multiplied by the number of seconds in a day, 86400 and the expression evaluates to the file date expressed as the number of seconds since day zero.

Finally, the ADDFILE DOpus command adds the file back into the window. The expression evaluates into the filename, size, type, seconds, comment, protection, reserved, and show (see your DOpus Command docs for more on this command). The program continues to read lines from the table, performing a test for a colon (:) in position 1. Apparently some archives have a throw away line with this syntax. Finally the loop ends.

Now the table file is closed. I corrected Mr. Petligs code. He did not ''call close()'', he merely put in ''close()'' which is bad form. In a command line interface this will cause an error. Deleting the temporary file and a display of what to do next in the top bar complete the archive display part of the program.

Expanding Archive Files

If the control enters the ELSE DO segment, then we need to expand selected files. FInfo is the rest of the line after the ''**Extracted from:'' (17 characters long) part of the file display in the window. It is the archive file name plus the protection bits. FileNm is the complete archive file spec. The protection bits are 10 characters long, so they are trimmed off the file spec, while the path is added to the left of the file name.

Now in a loop, one long command string containing all the selected files is built up from its components, and tested each time a filename to be extracted is concatenated to it, to see if the command will be longer than 254 characters. If it would be longer than 254, the new extract file is not concatenated and the command is executed in two or more parts, rebuilding a new command string after each execution. The string is built up and tested for length in the form:

```
LhAxarchivefileextractfile1[extractfile2...]destinationpath
```

This command executes in AmigaDOS when preceded by the ADDRESS COMMAND ARexx instruction. As long as ArcNm does not equal zero, (there are selected files left), a loop continues to concatenate the selected files from the archive file to the long command string. A long string is used to speed up the extraction, as an individual extraction for each file would be very slow. After the loop, a RESCAN of both windows completes the program. This is a clever program both in concept and in its uses of DOpus commands and the ARexx function, INDEX().

Please Write to:
Merrill Callaway
c/o Amazing Computing
P.O. Box 2140
Fall River, MA 02722-2140

Listing

```
/*
** LhArc_ext.dopus
**
** $VER: 1.12 by KjPetlig modified by MerrillCallaway
**
** Directory Opus v4.11 script to show files in a LhArc file and
** extract on request. Make a button called LHAdisp and have
** it call LhArc_ext.dopus and make sure the button that normally says
** executable is changed to ARexx.
** You must have LhA v.1.32 (or compatible versions).
**
*/

LhAPath='C:'

/*
Here is an addition for your help file:

*LHAdisp
This custom function will display all of the compressed files inside of a
.LHA or .LZH and allow you to extract individual ones. Select the .LHA or
.LZH file you wish to view and then press this gadget. The files inside
the
archive will appear in the window. You can then select multiple files and
press this gadget to extract the files you want, they will be extracted to
the directory in the other window.^

   History List - can be deleted for speed!

  ver 1.12 Modified/fixed by MerrillCallaway,
        author of THE ARexx Cookbook:
       Fixed problem with left/right windows, and problem if
      ''RAM DISK'' or other directories with a space are selected.
       Also named ''button'' LHAdisp instead of LHA ext
       (in Toptext), CALLed OPEN(), CLOSE() etc. properly.
       Corrected next century date conversion.
       Improved readability of code and string handling.
  ver 1.11 The -f switch of LhA doesn't work all the time, so I
       fixed it so this program doesn't get confused.
      Found problem in DirOpus when passing a single quote in
       a filename, don't know how to get around it!
      Added more plain instructions about LhAPath, set it
       to default to C: instead of my C:Utility/
      Re-Arranged order of header comment
  ver 1.10 Now the LhA command for each and every file extracted
       (I'm sure that was slow for floppy users!), it queues
       up the command to a max of 254 long before execution.
      Checked for no files selected, reports errors
        as ERROR: -=< what it is >=-
  ver 1.02 Added if-then for files dated after year 2000
       (Somebody had done some strange things with
       their clock! I had a file dated 30-Dec-05 !)
  ver 1.01 Had to suppress filenotes listing of LhA
      Changed LhArc's to LhA, (even renamed my
       LhArc correctly in my C: directory :-)
       Put a place in for the correct LhA path to
        kill an intermittent bug.
      Saw KAT's changes and incorporated one (rescan source!)
      Fixed for different versions of LhA, discards correct
         number of header lines.


*/

OPTIONS RESULTS

/* ADDRESS 'DOPUS.1' *//* unnecessary */

ID_Comment='**Extracted from:'

/*
**Modified/fixed by MerrillCallaway
**Fixes problem with a space in path name.
*/

/* Get the active window. */
STATUS 3
window=RESULT
destwind=ABS(window-1)
/* 0 or 1 */

/* Set the path name to active window path. */
STATUS 13 window
path=RESULT

/* Set the path name to the destination window. */
STATUS 13 destwind
destpath=RESULT

/* Get rid of path(s) with a space in it! */
IF LEFT(path,8)='RamDisk' THEN DO
 PARSE UPPER VAR path ':' rest
 path='RAM:'||rest
 END

IF LEFT(destpath,8)='RamDisk' THEN DO
 PARSE UPPER VAR destpath ':' drest
 destpath='RAM:'||drest
 END

/* End Callaway fix. */

GETNEXTSELECTED
ArcNm=RESULT /* file name of next selected */
IF ArcNm=0 THEN DO
 /* Callaway addition below */
 /* rescans window if nothing selected. */
 RESCAN window
 TOPTEXT 'No file was selected!'
 EXIT
END

FILEINFO ArcNm
FInfo=RESULT
ID_Index=INDEX(FInfo,ID_Comment)
IF ID_Index=0 THEN DO
 /* Callaway: rename ArcPath as ArcExt for less confusion. */
 ArcExt=UPPER(RIGHT(ArcNm,4))
 IF ArcExt~='.LZH' & ArcExt~='.LHA' THEN DO
  TOPTEXT 'ERROR: Improper extension!'
    EXIT
  END
 TOPTEXT 'Reading LhArc Archive....'

 /* Callaway mods follow: */
 ArcPath=path
 STATUS 6 window /* Find out how many entries to remove... */
 /* End Callaway mods. */

 DO RESULT   /* Clear window to work with... */
   GETENTRY 1
   REMOVEFILE result
 END

 ADDRESS COMMAND LhAPath||'LhA >t:RexxTemp vv 'ArcPath||ArcNm
 CALL OPEN('LhaList','t:RexxTemp','R')/* logical names quoted! */

 /* Read to end of header. */
 DOUNTIL left(FileNm,8)='--------'
   FileNm=READLN('LhaList')
 END

 /* First line of file descriptions. */
 FileNm=READLN('LhaList')

 /* Create a file list of all the files in the LhArc file. */
 DOUNTIL LEFT(FileNm,8)='--------'
   FileSpec=READLN('LhaList')
   /* Improvement to template by M.Callaway */
   PARSE VAR FileSpec FileSz . . date time prot .
   /* Improvement to date calculation by M.Callaway. */
   PARSE VAR date day '-' month '-' year
   IF year<78 THEN century=20;ELSE century=19
    /* Make a DOpus date from a LhArc one */
  month=INDEX('JanFebMarAprMayJunJulAugSepOctNovDec',month)+2
  date=century||year||RIGHT(month/3,2,'0')||RIGHT(day,2,'0')
    /* end of Callaway date mod */

   seconds=DATE('I',date,'S')*86400
   DirStat='-1' /* A file not a directory. */
   IF FileSz=0 THEN DirStat=1 /* A directory not a file. */
  ADDFILE FileNm FileSz DirStat seconds time||ID_Comment||ArcNm prot
'01'
   FileNm=READLN('LhaList')
   IF LEFT(FileNm,1)=':' THEN FileNm=READLN('LhaList')
 END

 CALL CLOSE('LhaList')
 ADDRESS COMMAND 'delete t:RexxTemp'
 /* Callaway mod */
 TOPTEXT 'Now select a file and hit |LhAdisp| to extract'
 END

ELSE DO
 /* Archive file and protection bits information. */
 FInfo=SUBSTR(FInfo,ID_Index+17)

 /* Callaway mods */
 /* Get only file spec w/o prot bits. */
 FileNm=path||LEFT(FInfo,LENGTH(FInfo)-10)
 ArcPath=destpath
 /* End Callaway */

 ComString=LhAPath||'LhA x' FileNm
 DO UNTIL ArcNm=0
  SELECTFILE ArcNm 0 1 /* 0=unselect file; 1=update display. */
   /* Build up a command string to a max of 254 characters. */
   IF LENGTH(ComString ArcNm ArcPath)>254 THEN DO
     address command ComString ArcPath
     ComString=LhAPath||'LhA x' FileNm
     END
   ComString=ComString ArcNm
   GETNEXTSELECTED
    ArcNm=RESULT
 END
 ADDRESS COMMAND ComString ArcPath
 RESCAN 1
 RESCAN 0 /* Well, I originally kept this line out on purpose! */
END
EXIT 0
```

·AC·