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 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 filespec 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.

Inside ARexx

A Directory Opus LHA Display Utility

by Merrill Callaway

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

```
version REXX: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:

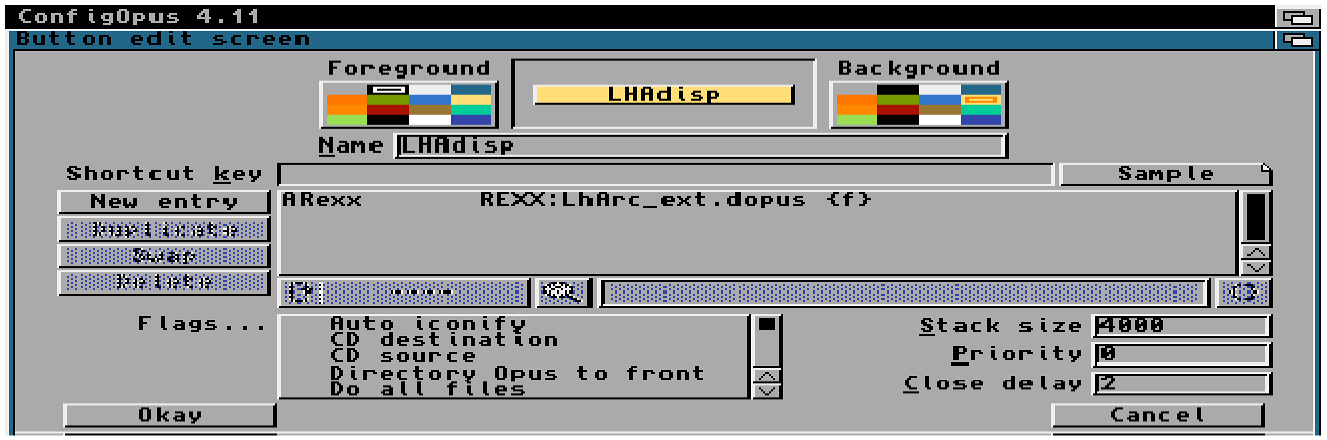
```
rx doit ?
```

then if the first part of the program is coded,

```
/* DOIT.rexx with help */
ARG help

IF help='?' THEN DO
 /* display help lines...*/
 SAY 'help line 1...'
 SAY 'template maybe...'
 SAY 'etc...'
END
/* rest of program */
```

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



Name LHAdisp

Shortcut key

```
ARexx          REXX:LhArc_ext.dopus {f}
```

Auto iconify
CD destination
CD source
Directory Opus to front
Do all files

Stack size 4000
Priority 0
Close delay 2