Script Files
- Part II -

by Ted Roche

In the last article, I discussed some of the basics of working with script files. For those of you who missed that issue, let me summarize a few key points here. A script is a text file of AmigaDOS commands performed by the EXECUTE command which can function without operator intervention. Script files are great for automating some of the drudgery of working with AmigaDOS by letting you work through the grammar and syntax of a series of commands only once and then calling them back as needed. Script files can be run from an icon using Workbench 1.3's ICONX program, or from a CLI/Shell by using the EXECUTE command. This article examines some of the special commands available only within script files.

Using ED

One frustrating experience I found with Workbench 1.3 was that the ED text editor wipes out the Script protection bit when it edits a text file. I worked out the following script file, called SE, saved in the S: subdirectory, and called up by typing SE <filename>

```
.key filename
; SE - Script Edit
if "<filename>" eq ""
   echo "ERROR: must specify filename."
skip END
endif
ed <filename>
protect <filename> +s
lab END
```

This script illustrates a few features of scripts that I have not previously covered. The first line uses a "dot" command, which allows a script file to be called with parameters specified on the command line. This lets you specify a filename, for example, or a device name. Dot commands have a number of remarkable capabilities, and I plan to devote an entire future article to them.

The second line contains a remark -- a non-executable line of text which lets you put in comments, explanations of "tricky" code, or a credit line. While not required, remarks can be a big help in debugging or in trying to understand someone else's (or your own!) programming techniques. All text following a semicolon is treated as a remark; EXECUTE will ignore the remainder of the line.

The third through sixth lines of code are an IF ... ENDIF structure. The IF command is followed by a condition to be tested, in this case, whether the filename required by this script file has been specified or is a null string (""). If no file was specified, the next line is executed, otherwise, all code is skipped until the ENDIF statement is reached. The IF . . . ENDIF commands will also recognize an ELSE option, for alternate courses of action:

```
IF (condition) (do one set of actions)
ELSE
   (do another set)
ENDIF
```

The ECHO command prints text in the CLI/Shell from which the script was called. In this case, the text informs the operator that a filename must be specified. The SKIP command tells EXECUTE to skip all command lines until it comes across a LABel named END. In this file, SKIP causes EXECUTE to jump to the last command in the file. Note that the ECHO and SKIP commands are indented to make it easier for the reader to follow the flow of logic within the program.

Assuming a file has been specified, the script calls ED, the Workbench 1.3 text editor, specifying the filename to be edited. When editing is complete, control is returned to the script file, which resets the Script protection bit for the file. Voila! A simple solution to a minor bug.

A Script Text File Viewer

Another small aggravation with AmigaDOS comes when using the TYPE command to display a file on the screen. If the file is over 25 lines in length, it just zips by, right off the top of the screen. Using Workbench 1.3's MORE program to read the file fixes that, but the delays while each page loads from disk can be maddeningly long. One solution is to use a "buffer" to hold the upcoming portion of the text file. The S:PTYPE script listed below is called by typing PTYPE <filename>

```
.KEY filename/a
; Ptype - Type a file to a PIPE: and display using MORE
if exists <filename>
```

```
    run type >pipe:ptype<$$> <filename>
    run sys:utilities/more pipe:ptype<$$>
else
    echo "ERROR: File <filename> does not exist."
endif
```

Again, a dot command specifies the KEYword required. The /a option after the keyword tells EXECUTE that this keyword must be supplied. If I forget, and just type "PTYPE" and hit return, the message 'EXECUTE: Parameters unsuitable for key "Filename/a" ' is displayed, reminding me to add the filename, although perhaps the error message is not as clear as in our earlier example.

The third line starts an IF . . . ELSE . . . ENDIF structure that extends to the end of the script file. IF EXISTS tests for the existence of the file specified; if found, the lines following the IF statement are performed. If the file can't be found, the ECHO line following the ELSE is performed instead.

The fourth line starts the new task of running the TYPE command. The output of the TYPE command (the text file) is directed to one of Workbench 1.3's new devices, PIPE:. This device allows you to "pipe" information from one task to another. The name used after PIPE: identifies this particular pipe, so that other tasks may access the pipe. I named this pipe as ptype<$$> -- the <$$> is a special symbol recognized by EXECUTE, and it will substitute the number of the CLI/Shell that the script was called from. If your CLI/Shell prompt says "2>" then the pipe you create will be named PTYPE2. This lets you have multiple CLI's each using the PTYPE command without confusion.

Now that we've started sending our file out into who-knows-where, let's go get it. The fifth line uses RUN to start the MORE program running in its own full-sized window, rather than in the current CLI, which may be a much smaller window. With RUN, control is returned to the original (calling) CLI/Shell, so that you can flip back and forth to the CLI and use other AmigaDOS commands while the text file is being displayed.

I hope I've shown you some of the potential of script files. If you have comments, suggestions, or want to contribute a script file of your own, please drop a note to me in care of the *Amiga Culture* editor.

Ted Roche is co-owner of Computer Resource, *a New Hampshire-based consulting firm specializing in training on Amiga software and hardware applications.*