ARx_UcrTutor1.ag

COLLABORATORS			
	TITLE : ARx_UcrTutor1.ag		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 8, 2025	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

ARx_UcrTutor1.ag

1.1 main

```
   AN AMIGAGUIDE® TO ARexx                 Edition: 1.0a
   by Robin Evans

      Note: This is a subsidiary file to ARexxGuide.guide. We recommend
      using that file as the entry point to this and other parts of the
      full guide.
```

1.2 arg FileName UCdir .

Return to program listing

```
The  keyword   ARG  is an abbreviation for the instruction PARSE UPPER ARG.
 PARSE  is an extraordinary instruction that gives ARexx power to handle
text strings that is unmatched in most programming languages.

As we’ve used it here, the instruction picks up the first two words (which
are anything with a space on either side) typed on the CLI after the
program name. In this case, we’ve picked up the filename and uncrunch
directory, if it’s included.

The ARG instruction also translates values into uppercase (capital
letters). That’s useful in this program, but there are times when it is
undesirable. In those cases, use the unabbreviated instruction  PARSE ARG .

As it is used here, the ARG command won’t recognize either a file name or
a directory specification that includes spaces. It might be an interesting
exercise to add that capability, but we won’t do it in this tutorial.
```

1.3 if FileName = ’?’ then

Return to program listing

Shell users are accustomed to getting a template of options after typing a command name followed by '?'. It works like this with the copy command:

```
   >> copy ?
   FROM/M,TO/A,ALL/S,QUIET/S,BUF=BUFFER/K/N,CLONE/S,DATES/S,NOPRO/S,
   COM/S,NOREQ/S:
```

This line gives UnCrunch.rexx the same facility by recognizing the cry for help. If the user has typed { UnCrunch ? }, the following lines will be executed.

1.4 do

Return to program listing

The keyword DO groups the following clauses into what ARexx considers a single instruction . It performs a function similar to the block identifier '{' in C or 'BEGIN' in Pascal.

DO is often used in conjunction with an IF/THEN instruction because IF will execute only the clause that immediately follows it. Using DO turns multiple clauses into a single instruction so that all of them will be executed when the IF condition is true.

Like the '}' closing brace in C, END is a subkeyword that must always accompany DO.

1.5 options prompt 'UCR FILENAME/A, DESTINATION/F: '

Return to program listing

The OPTIONS keyword is used for a number of unrelated tasks in ARexx. Here, it sets up a prompt string to be used later by another instruction. Note that OPTIONS PROMPT doesn't actually present a prompt to the user. That is done with the PULL keyword on the next line.

1.6 pull FileName UCdir

Return to program listing

Like ARG , PULL is an abbreviation for a variation of the PARSE instruction. The full instruction in this case is PARSE UPPER PULL . This instruction echos the ARG instruction used at the beginning of the main program listing. That's because it does the same thing except that, instead of taking its arguments from the command line, it pulls them interactively from the user. Here's the effect of this command:

```
         > rx uncrunch ?
         UCR FILENAME/A, DESTINATION/F:
```

The template defined by the OPTIONS PROMPT instruction is presented to the user with the cursor positioned one space after the colon. (There is a space before the final quotation mark in the OPTIONS PROMPT parameter.) Once the user presses the return key, the PULL instruction will retrieve whatever was entered.

1.7 if FileName = ” then

Return to program listing

Since the program can't do anything without a filename, this section makes sure that we don't continue without one. This IF instruction compares [FileName] to an empty string, which is like saying, 'If there isn't a FileName, then...'.

The '=' sign here is a comparison operator that means something like 'is the same as'. Using the single '=' comparison operator means that any leading or trailing blanks on the value being compared will not be significant. In other words, { '   foo   ' } will be the same as { 'foo' }.

In cases where exact comparison is desired, use the comparative operator '=='.

1.8 if UCdir = ” then

Return to program listing

The IF instruction here, unlike the one above that checks for a blank filename, is not followed by an 'end' or 'endif' keyword. In many languages, 'endif' is a required part of any 'if' command. In ARexx, however, the IF/THEN instruction will automatically execute the one clause following THEN, so 'endif' is not required.

On the other hand, the secondary keyword THEN is required whenever IF is used.

1.9 UCdir = 'RAM:'

Return to program listing

We've used the '=' sign in two previous clauses, but this '=' sign means something different. In the line directly above this one {if UCdir = '' } the '=' sign is a comparison operator .

Here, however, the '=' sign performs a significantly different task; it identifies an assignment clause which associates the value of the expression on the right side of the sign, 'RAM:', with the variable to its left, [UCDir].

1.10 if right(UCdir, 1) ~= ':' & right(UCdir, 1) ~= '/' then

Return to program listing

Here we find a new aspect of the REXX language called functions . Functions are self-contained programs that perform an operation and return a value of some sort. Although it would be redundant, an add() function might take the arguments 2 and 3. It would return the value 5.

There are three classes of functions. RIGHT() is a built-in function that is always available to any ARexx program. This line shows a standard form of the function. The values inside the parentheses are arguments that are sent to the function.

In this case, the RIGHT() function gets the value of the variable [UcDir] and the number 1. It then 'returns' the right-most character of [UcDir] There is no assignment clause here because the value returned by the function is used directly in the IF clause.

ARexx has several functions that perform similar tasks. In line 79 , the verify() function is used along with right() to accomplish the same thing done here with the two comparative operations.

1.11 ArcExt = upper(right(FileName, 3))

Return to program listing

Two functions are nested in this clause. The value returned by the inner function, RIGHT() , becomes an argument to the outer function, UPPER() .

The RIGHT() function retrieves the last (or right-most) three characters of the variable [FileName]. The UPPER() function translates those characters to upper case, which would make the comparisons in the lines below more accurate if the [FileName] variable contained a value in mixed-case.*

The result or value returned by UPPER() is assigned to the variable to the left of the '=' sign -- [ArcExt]. The assignment to a variable will allow us to use the result of this function again without needing to call the function a second time.

* The UPPER() function is included here to demonstrate how it can be used, even though it is redundant in this case since the [FileName] was already translated to upper case in line 3 by the ARG instruction.

1.12 address command

Return to program listing

The ADDRESS instruction changes the host of subsequent commands. The 'COMMAND' option indicates that AmigaDOS should serve as the host. Now that this instruction has been issued, any commands issued later in the program will be sent to AmigaDOS.

1.13 select

Return to program listing

SELECT is a powerful cousin of the IF instruction. It precedes a list of possible conditions each of which is identified by the WHEN keyword. ARexx makes its way through the list and executes the instruction following the first conditional that is TRUE. If there is no match, the OTHERWISE clause, which is required, will be executed.

1.14 when ArcExt = 'LZH' | ArcExt = 'LHA' then

Return to program listing

The syntax for WHEN is similar to that of IF except that it will not take an ELSE clause since each successive WHEN clause already acts like ELSE. Only the first WHEN clause that tests true will be executed. The following WHEN clauses as well as the concluding OTHERWISE will be skipped.

Here, the clause checks the variable [ArcExt] against either of two possible values. The '|' in the middle of the clause is an ARexx logical operator that means OR. This clause will test TRUE if [ArcExt] is equal to either of the supplied values.

1.15 'Lha -x x' FileName '#?' UcDir

Return to program listing

The command 'Lha' has nothing to do with ARexx, but its presence here begins to hint at the extraordinary power of ARexx as an interprocess communication language.

The quotation marks surrounding the command tell ARexx that it should not interpret anything inside the string . For instance, without the quotation marks, ARexx would try to subtract a variable [x] from a variable [Lha] when encountering the terms { Lha -x }. It wouldn't work. Instead, the quotation marks identify the clause as a command that should be passed to the host address.

Because of the ADDRESS COMMAND instruction above, the host address for ARexx is now the shell. The command -- Lha with all the options -- will be sent to AmigaDOS and executed there as it would be if it was typed it in directly.

1.16 call SetDest('ZOO x//')

Return to program listing

Although the syntax is similar, the function used here is different than those used previously. SetDest() is an internal function defined within this script.

When it encounters the function call, the interpreter looks for a label matching the function name. The colon after SetDest: tells ARexx that the following program following lines define the function.

Functions can be used in either of two forms that make them easy to spot. The symbol or word used to identify the function is either followed by a set of parentheses or it is preceded by the keyword CALL . (Parentheses may be used even with the CALL keyword, as they are in this script, but are not necessary.)

1.17 otherwise

Return to program listing

OTHERWISE is a keyword that must always appear as the final clause in the list of WHEN conditions associated with a SELECT instruction. The clause after OTHERWISE is executed only if all of the preceding WHEN conditions failed.

Even if there's nothing to do, OTHERWISE must be used, but need not be followed by anything other than the END that closes the SELECT range.

```
   select
      when ...
      when ...
      otherwise
   end
```

1.18 end

Return to program listing

This is an example of the only situation in ARexx where END is not paired with the DO keyword. In this case, it closes the range of clauses associated with SELECT . END must always be used with SELECT.