RexxLA Symposium 2019

Multi-line Strings and ooRexx:
A discussion

Gil Barmwater

Overview

- What is a “multi-line string”?
- Multi-line strings in ooRexx
- A tool to help create them
- Summary

What is a multi-line string?

- Depends on who you ask!
  - Could be just a string that displays on multiple lines:
    - say “This string”.endofline”shows as 2 lines”
  - More likely to be a long string that is defined on multiple lines
    - say “This string, while not that long,” -
      “spans 3 lines by using the ooRexx” -
      “continuation syntax.”
  - For this discussion, the latter case will be what is meant (but the former is not excluded)

Multi-line strings in ooRexx

- The previous example shows (oo)Rexx supports strings that are defined on multiple lines
  - Must follow the rules:
    - Statement continuation must be indicated
    - Literals can NOT span lines
- Sometimes the “rules” make it difficult to define the string data
  - Usually when creating a string meant to be processed by a non-Rexx program

Multi-line strings in ooRexx (2)

- An example of a partial Rexx statement to create an SQL command:

  -
    ```
    sql_cmd = "INSERT INTO wp_posts (",
              "`ID`,",
              "`post_author`,",
              "`post_date`,"
    …
              "VALUES(",
              "'"gp.ID.1"',",
              "'"gp.post_author.1"',",
              "'"gp.post_date.1"',",
    ...
    ```

Multi-line strings in ooRexx (3)

- Other Languages
  - Many implement a syntax based on Here Documents: c.f. https://en.wikipedia.org/wiki/Here_document
  - Python has a "verbatim" string delimited by triple quotes
  - Resulting strings may contain variables and include the end-of-line character(s)
- ooRexx can provide the functionality without adding additional syntax to the language

The Tool - MLString

- Based on work spanning several years
  - Additions made as requirements changed
  - Rewrites done as “new” features of ooRexx discovered
- Consists of a Class file and a Native Routine library
  - Easy to use:
    - Add a ::REQUIRES directive for the class file to your program
    - Ensure the .cls file and library are accessible

The Tool - MLString(2)

- Design requirements
  - What does the object need to know?
    - The data that will be part of of the object
    - Implemented as Attributes
  - What must the object be able to do?
    - The methods that the object can execute
    - Only job is to create a string

The Tool - MLString(3)

- The Attributes
  - The “lines” that are to be the parts of the string
    - Since the order is important, an array holds the collection
    - May be specified as the first argument of the “New” message or set using the attribute name - mlsParts
    - E.g. .array~of(“This string, while not that long,” , “spans 3 lines by using the ooRexx” , “continuation syntax.”)

The Tool - MLString(4)

- The Attributes (cont.)
  - How the “lines” are to be joined together to form the string
    - Should the “lines” be joined by abuttal concatenation, blank concatenation, .endofline, or some other string
    - May be specified as the second argument of the “New” message or set using the attribute name – join
    - Note that Rexx continuation syntax is actually blank concatenation

The Tool - MLString(5)

- The Attributes (cont.)
  - The variable delimiter
    - The character used to enclose variables to distinguish them from literal data
    - May be specified as the third argument of the “New” message or set using the attribute name – varFlag
    - As opposed to Rexx syntax, the “lines” are considered literal data, continuation is implied (the end of the array indicates end of the data), and variables are enclosed by the varFlag character

The Tool - MLString(6)

- The Attributes (cont.)
  - The variable values
    - Because of encapsulation, the object cannot “see” the variables of the calling program
    - Supplied as a directory whose indexes are the variable names and whose items are the corresponding values
    - May be specified as the fourth argument of the “New” message or set using the attribute name – var_values
    - Easily done using .context~variables

The Tool - MLString(7)

- The Methods
  - toString
    - Takes the object data “line” by “line”, strips leading and trailing “whitespace”, searches for variables and substitutes their values, and then joins the “lines” together using the “join” string
    - Accepts up to three arguments to supply/override the values for the join, varFlag and var_values attributes
  - makeString
    - Simply forwards to toString; no arguments

Using MLString

- Specifying the array of “lines” for mlsParts
  - OoRexx 5.0.0 introduces the Resource directive
    - Very similar to a Here Document
    - The lines between the directive and the end delimiter may be almost any data
    - An example using earlier data

```
::resource theLines
This string, while not that long,
spans 3 lines by using the ooRexx
continuation syntax.
::END
```

Using MLString(2)

- Specifying the array of “lines” for mlsParts (cont.)
  - Retrieve the data as an array using .resources[thelines]
  - Pass that as the first argument of the “New” method
    - .mlstring~new(.resources[theLines])
  - Alternatively
    - .mlstring~new(theLines)

Using MLString(3)

- Specifying the array of “lines” for mlsParts (cont.)
  - Including the data “inline”
    - Simply enclose the block of lines in a single block comment
    - Pass an “offset” (integer) to the first data line as the first argument of the “New” method
    - An example

```
.mlstring~new(1) /*
 this is line 1 and
 this is line 2.
 */ ---
```

Using MLString(4)

- Specifying the “join” string
  - For blank concatenation, use “ ”
  - For abuttal concatenation, use “”
  - For multiple output lines, use .endofline
    - Special case: “lines” are NOT stripped
    - Output is WYSIWYG (variables may still be included and will be replaced by their values)
    - Typical usage is “help text”

Using MLString(5)

- Working with variables
  - MLString is NOT an expression evaluator
    - No function calls – date()
    - No calculations – n / 3
  - Either simple or compound variables will be replaced by their values
  - Must specify both the delimiter/flag AND the directory of values to MLString
  - Choose a flag character that is NOT in your data (or, at least, occurs sparingly)

Using MLString(6)

- Working with variables (cont.)
  - A data “line” must contain an even number of varFlag characters or they will be ignored
  - If varFlag is \ and fred has the value 3:
    - \fred\ → 3
    - \ferd\ → FERD
    - \ \ → a blank
    - \\ → \
  - Alternate syntax for supplying var_values:
    - .mlstring~new(, , , .context)

Using MLString(7)

- Eliminating the need to pass .context
  - Routines written in C++ have access to an API that will return the calling program's context
  - The MLString function takes 3 optional arguments
    - Same as the first three arguments of the “New” method of the .mlstring class
  - Using the native APIs
    - Determines the caller's context
    - Interfaces to the .mlstring class using the 4 args

Using MLString(8)

- Eliminating the need to pass .context (cont.)
  - Rather than return an instance of .mlstring, the function returns the result of sending that instance the toString method, i.e. a string
  - An example:

    -
      ```
      fullStop = '.';msg = mlstring(1, ' ', '\') /*
        This is line 1 and
         this is line 2\fullStop\
        --- */
        say msg
      ```
    - Yields
      ```
        This is line 1 and this is line 2.
      ```

Using MLString(9)

- Another example
  - The sql command from before:

```
• sql_cmd = "INSERT INTO wp_posts (",
                "`ID`,",
                "`post_author`,",
                "`post_date`,",
   …
                "VALUES(",
                "'"gp.ID.1"',",
                "'"gp.post_author.1"',",
                "'"gp.post_date.1"',",
   ...
```

Using MLString(10)

- Another example (cont.)
  - Using the mlstring routine
    - sql_cmd = mlstring(1, '', '\') /*

```
INSERT INTO wp_posts (
            `ID`,
            `post_author`,
            `post_date`,
…
            VALUES(
            '\gp.ID.1\',
            '\gp.post_author.1\',
            '\gp.post_date.1\',
…
*/ ---
```

Summary

- Rexx permits the definition of a string to span multiple lines
  - The syntax required can obscure the contents of that string, especially for complex strings used to communicate with non-Rexx programs
- The MLString tool can help
  - Allows the programmer to focus on the syntax of the string he is trying to create without being concerned with making it “legal” Rexx as well
- Questions/feedback: gbarmwater@alum.rpi.edu