REXX Library Collection

By Alex Taylor

Background

- The more I use REXX, the more often I find myself asking “I wish I could do this…”
  - Frequently, my solution is to write a REXX library that does it.
- I’ve written several different REXX libraries for OS/2
  - Some are highly specialized and not publicly released
  - Most are available (with full source) on NetLabs:

http://trac.netlabs.org/rexxlibs

RXLVM - REXX Interface to LVM

- My first REXX library (?)
- Provides read-only access to Logical Volume Manager functions
  - Open and close LVM Engine
  - Query IBM Boot Manager information
  - Get physical disk information
  - Get logical volume information
  - Get partition information
  - Refresh removable media
- Requires WSeB/MCP or later (obviously)
- Requires understanding certain LVM concepts (e.g. using handles, opening/closing the LVM engine, etc.)

RXLVM Example

```
CALL RxFuncAdd 'RxLvmLoadFuncs', 'RXLVM', 'RxLvmLoadFuncs'
CALL RxLvmLoadFuncs

result = RxLvmEngineOpen()
IF Left( result, 6 ) == 'ERROR:' THEN DO
    PARSE VAR result 'ERROR:' lvm_error_code
    SAY 'LVM returned error:' Strip( lvm_error_code )
    RETURN lvm_error_code
END

CALL RxLvmGetVolumes 'volumes.'
SAY '========' volumes.0 'VOLUMES FOUND ========'
DO i = 1 TO volumes.0
    SAY Word( volumes.i, 3 ) || '->' || Word( volumes.i, 2 ),
        SubWord( volumes.i, 4 )
END
```

RXULS - REXX Unicode Library

- Interface to the OS/2 Unicode (“Universal Language Support”) API
- Tries to simplify usage of the ULS API…
  - …still requires some understanding of Unicode and ULS concepts
- Includes functions to:
  - Convert text between different codepages
  - Convert text to/from Unicode formats (e.g. UCS-2, UTF-8)
  - Transform text
  - Query locale information
  - Format time & date strings
  - Read & write Unicode clipboard text

Important (RX)ULS Concepts

- OS/2 codepages (INF file includes a reasonably complete list)
- Unicode text encodings:
  - UCS-2 (≈UTF-16) : codepage 1200
  - UTF-8 : codepage 1208
- Common parameters:
  - subchar – substitution character (not for all codepages)
  - controls – how to interpret ASCII control bytes (D | G | C | L)
  - path – controls conversion of DBCS '\' and '~' (Y | N)
  - locale – name of locale to use when localization applies

  (These can be confusing so read the documentation!)

RXULS Example

```
CALL RxFuncAdd 'ULSLoadFuncs', 'RXULS', 'ULSLoadFuncs'
CALL ULSLoadFuncs

string = "A café in Trois-Rivières"
SAY 'In codepage 850:' string

string_conv = ULSConvertCodepage( string, 850, 862, '3f')
IF ULSERR <> '0' THEN SAY ULSERR
ELSE SAY 'In codepage 862:' string_conv

string_upper = ULSTransform( string, 'upper', 850 )
IF ULSERR <> '0' THEN SAY ULSERR
ELSE SAY 'Uppercase:' string_upper
```

RXPRTUTL - Printer Management

- Originally written to support PM Printer Manager
- Allows manipulation of printers and print queues from REXX:
  - List installed drivers and printer models
  - List printer ports
  - Query, create, delete or configure a printer port
  - List printers (queues)
  - Query, create or delete a printer
  - Open a printer's WPS object
  - Set the system default printer
  - Hold or release a printer queue
- Properly cleans up printer queues when deleting printers

RXPRTUTL Example

```
CALL RxFuncAdd "RPULoadFuncs", "RXPRTUTL", "RPULoadFuncs"
CALL RPULoadFuncs

rc = RPUEnumPrinters('printers.')
IF rc == 0 THEN SAY RPUERROR
ELSE DO
     SAY printers.0 'printers are defined:'
     default_printer = ''
     DO i = 1 TO printers.0
          CALL CHAROUT, '['printers.i.!name']' printers.i.!description
          IF POS('D', printers.i.!flags ) > 0 THEN DO
               default_printer = printers.i.!queue
               CALL CHAROUT '  <- DEFAULT'
          END
          SAY
     END
END
IF default_printer <> '' THEN DO
     rc = RPUOpenView( default_printer, 'O')
     IF rc == 0 THEN SAY RPUERROR
END
```

RXUTILEX - RexxUtil Extensions

- Useful functions of a general nature,

  a.k.a. “Stuff that should have been in RexxUtil”

- Currently includes:
  - Clipboard access
  - Process management
  - System information (e.g. RAM)
  - Replacing locked files
  - Number, time & currency string formatting
  - Named pipe creation & deletion
  - Large (>2GB) file access

- Plus other things I (or others) think of…

Some RXUTILEX Examples

```
CALL RxFuncAdd 'Sys2LoadFuncs', 'RXUTILEX', 'Sys2LoadFuncs'
CALL Sys2LoadFuncs

pc = Sys2QueryProcessList('procs.')
SAY pc 'processes running:'
DO i = 1 to procs.0
    SAY procs.i
END

SAY
SAY 'System RAM:' Sys2FormatNumber( Sys2QueryPhysicalMemory() ) 'kB'
SAY
SAY 'Screen resolution:' Sys2QuerySysValue('CXSCREEN') ,
                         Sys2QuerySysValue('CYSCREEN')
```

VX-REXX Extras

https://altsan.org/programming/os2/#vrobjex

- An extension library for VX-REXX
- Currently provides:
  - Tooltips (with any type of control)
  - Progress bar (supporting a few different visual styles)
  - Enhanced text+image button control
  - Warp-4-style notebook control
  - Colour selection dialog
  - A directory selection dialog
  - Clipboard functions (that also work in text-mode programs)
- Source is available at http://trac.netlabs.org/vxapps/browser/projects/vrobjex

Library Framework

- Common functions for handling parameters and return values: http://trac.netlabs.org/rexxlibs/shared
  - REXX string memory allocation
  - Supports creating stem variables
  - Standardized error code handling
- Mostly written by me, some improvements by Steven Levine
- Can be used for other REXX libraries

Questions?