IBM Software

REXX Portability and Tips

Share Session 8826, Spring
March, 2011

W. David Ashley
dashley@us.ibm.com

ON DEMAND BUSINESS™

© 2011 IBM Corporation

Important REXX Compiler Disclaimer

The information contained in this presentation is provided for informational purposes only.

While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided “as is”, without warranty of any kind, express or implied.

In addition, this information is based on IBM’s current product plans and strategy, which are subject to change by IBM without notice.

IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other documentation.

Nothing contained in this presentation is intended to, or shall have the effect of:

- creating any warranty or representation from IBM (or its affiliates or its or their suppliers and/or licensors); or
- Altering the terms and conditions of the applicable license agreement governing the use of IBM software.

Agenda

- **REXX Compiler**
- **ooRexx**
- **REXX Hints and Tips**
  - Variable Names
  - Style Tips
  - Execution Optimization

ON DEMAND BUSINESS™

REXX History

- **Rexx (Restructured eXtended eXecutor)**
- **1979mar29 Mike Cowlishaw (IBM Fellow) publishes initial specification**
- **Late 1979 first implementation internal to IBM on VM/CMS.**
- **Available to the general public in 1983 VM (3rd release)**
  - 25 years ago, Winter 1983 Share (San Francisco), Mike Cowlishaw and Rich McGuire demonstrated Rexx to the public.
- **1985 first non-IBM version appears.**
- **1987 IBM announces Rexx to be the Procedures Language for SAA (Systems Application Architecture)**
  - MVS/TSO, AS/400, 1989 OS/2 1.2 EE
- **1989 REXX Compiler for MVS and VM released**
- **1990 first (annual) Rexx Symposium**

REXX History

- **1990 Rexx 4.0 Language published. OS/2 1.3 first implementation**
- **Early 1990s versions available for AIX/6000, PC-DOS, Netware, CICS.**
- **1996 ANSI “Programming Language Rexx”, X3.274-1996**
- **1996 NetRexx**
- **1996 Object Rexx released OS/2 version 4. 1997 Windows and Linux, 1998 AIX, 2000 Linux/390, 2002 Solaris.**
- **2003 Rexx Compiler Release 4 (aka Version 1.4)**
- **2005 Open source ooRexx (Open Object Rexx)**
  - http://www.rexxla.org/

ON DEMAND BUSINESS™

REXX Compiler on z/OS and z/VM

- **IBM Compiler for REXX on zSeries Release 4**
  - z/VM, z/OS: PID 5695-013
- **IBM Run Time Library for REXX on zSeries Release 4**
  - z/VM, z/OS: PID 5695-014
  - VSE part of operating system
- **IBM Alternate Library for REXX on zSeries Release 4**
  - Free download
  - http://www-306.ibm.com/software/awdtools/rexx/rexxzseries/altlibrary.html
  - Included in z/OS 1.9 base operating system
- **Continued ongoing support for Release 4**
  - Release 4 has been available since 2003
  - Release 3 is no longer in service

ON DEMAND BUSINESS™

Invoking Compiler

- **Invoke on VM command line**
  - REXXC *source_file* (*options*
  - REXXC or REXXC ? or HELP REXXC
- **Invoke with VM full screen panel**
  - REXXD *source_file*
- **Invoke with VM bath facility (product id 5664-364)**
- **Invoke with z/OS command shell**
  - REXXC *source_file options*
  - ex 'RXT.V140.SFANCMD(REXXC)' 'SHARE.REXX(MORT) XREF ALT‘
- **Invoke with z/OS ISPF primary panel**
- **Invoke with z/OS JCL**

REXX Compiler Libraries

- **A Rexx library is required to execute compiled programs**
- **Compiled Rexx is not a LE language**
- **2 choices: Run-time library and Alternate library**
  - Run-time library. Program product.
  - Alternate library. Free. Uses the native system's REXX interpreter.
- **Compiled and library code runs in 31-bit mode**
  - base/displacement instead of relative addressing
  - BALR and other old opcodes. Can run on old hardware.
  - No z/Architecture in plan today.

Primary and Alternate Libraries

- **REXX Library (PID 5695-014)**

  (aka Primary Library or Run-time Library)

```
rexxc test exec (cexec(test1 exec)
exec test1
```

  - test1 → primary library

- **Alternate Library**

```
rexxc test exec (cexec(test2 exec) alt
  sl
exec test2
```

  - test2 → alternate library → system interpreter

- **Will run whichever library is loaded in memory**
  - Alternate library execution requires ALT and SLINE

Alternate library shipped with R1.9 of z/OS

- **Starting with release 1.9 (Sept 2007) of z/OS the alternate library is shipped with the base OS.**
- **Identical to the free, downloadable, distributable alternate library.**
  - No need for software developers to include the alternate library with their shipped packages.
  - No need for users to download and install the alternate library.

ON DEMAND BUSINESS™

Compiler Advantages

- **Program performance**
  - Known value propagation
  - Assign constants at compile time
  - Common sub-expression elimination
  - stem.i processing
- **Source code protection**
  - Source code not in deliverables
- **Improved productivity and quality**
  - Syntax checks all code statements
  - Source and cross reference listings
- **Compiler control directives**
  - %include, %page, %copyright, %stub, %sysdate, %systime, %testhalt

Source Code Protection

- **Protects your intellectual property**
- **Protects your code from manipulation**
- **Keeps your code maintainable**

ON DEMAND BUSINESS™

Improved Productivity and Quality

- **Debugging: cross reference listing**
- **Syntax check of all statements**
- **Syntax check without code execution**
- **Compiler error messages**
- **Lists all errors – no stopping at first error**

OoRexx 4.1.0

- **Released November 2010**
- **Major new version, many internals changed**
- **Backwards compatible**
- **New C++ API is supplied. The old API set will be maintained for the foreseeable future**
- **Major update to the Programming Guide**

OoRexx 4.1.0 (cont.)

- **Many performance enhancements**
- **Additional new classes including Socket, StreamSocket, MIME, and SMTP**
- **New RXAPI interface (now a sockets interface, no shared memory usage)**
- **HostEmu (ExecIO, HI, TE, TS)**
- **csvStream class**
- **Support for many platforms**

ON DEMAND BUSINESS™

ooRexx 4.2.0

- **A 4.1.1 release may happen to fix an rxapi problem**
- **Probable release in November 2011**
- **Additional new platform specific function and class libraries**
- **Complete update to ooDialog**
- **RexxGTK will be released at the same time**
- **OrxSQL will be released at the same time**

ON DEMAND BUSINESS™

Rexx Tips

Bad Variable Names

- **X – can cause problems with string constants in abbutal concatenations.**

```
x = 2010

bad = '0101'x' CE'   /* unexpected results */
```

- **B – can cause problems with string constants in abbutal concatenations (see above example).**
- **Result – value could be modified after a subroutine call.**
- **Rc – value is not stable between host commands.**

Rexx Tips

Bad Variable Names (cont.)

- **Don't use uninitialized variable names. Always quote literals!**

```
/* this is really a bad idea */

mydate = date() ce  /* ce is unitialized */
```

Rexx Tips

General Style Tips

- **Capitalizing variable names will be frowned on by most classic Unix programmers.**
- **Be consistent with indentation.**
- **Don't go crazy with extra blanks as it actually makes your code tricky to read.**
- **Always put a blank before the comma line continuation character for readability.**
- **Be consistent when quoting string constants (single or double quotes).**

Rexx Tips

General Style Tips (cont.)

- **In ooRexx, avoid nesting function invocations. Use chained method invocations instead. This can save a lot of typing mistakes.**
- **If using ooRexx, use arrays instead of stems with numeric indexes.**
- **Avoid variables names longer than 15 characters for readability.**
- **Avoid using the ¬ operator as it is not portable.**

Rexx Tips

General Style Tips (cont.)

- **Don't cross line boundaries with string literals as this is not portable to non-mainframe Rexx implementations.**
- **Don't put parenthesis around arguments to CALL.**
- **Be careful how you code the vertical bar (|). This character is sometimes not portable between the mainframe and the Windows and Unix platforms. When moving code between platforms be sure to check that the character has been translated correctly.**

Rexx Tips

General Style Tips (cont.)

- **Try not to use the characters (!) and (?) in Rexx variable names. When the code is moved to a Unix platform it can cause readability problems with programmers of other languages in that environment.**
- **Block comments are always a good thing, until taken to the extreme. Large block comments can make your indentation scheme harder to follow.**

Rexx Tips

Execution Optimization Tips

- **Don't code semicolons at the end of statement lines. This actually adds nil statements to your program! (not valid for the Rexx Compiler)**
- **Avoid a long list of EXPOSE variables on PROCEDURE clauses (does not apply to ooRexx method definitions).**
- **If possible use the ooRexx ROUTINE directive instead of PROCEDURE. It is faster since ooRexx caches those definitions.**
- **For mainframe interpreted Rexx try to reduce the number of code files as this can cause an initial performance hit.**

Join RexxLA

- **If you have interest in the Rexx language, join the Rexx Language Association.**
- **Large member base all devoted to the Rexx language and its promotion.**
- **Product agnostic (mainframe REXX, Regina, ooRexx, NetRexx, RexxIMC, etc.).**

ON DEMAND BUSINESS™