Introduction to Multi-Platform REXX Programming

Course Summary

Description

This course is designed to enable the attendee to create, execute, debug and modify programs in the REXX programming language. The course includes execution under MVS, both from TSO/E and non-TSO/E address spaces. Comparisons to similar CLIST capabilities are provided where reasonable. Quizzes and labs are used to reinforce presented topics.

Linux/UNIX, Windows, NetView, VM/CMS, PC/DOS, OS/2, as well as execution with automation products will be addressed, as needed, or the course may be tailored to those environments for site-specific needs.

A principle advantage of REXX is the relative ease of creating well-structured programs. Development techniques will be described which allow rapid design and execution of REXX programs while providing maintainability and allowing upgrades in a logically layered fashion.

The extensive practical experience of the instructor will emphasize 'real world' applications of REXX. Provided example programs cover a wide range of topics, including automated operations and conditionally executing TSO/E and ISPF commands.

At the conclusion of the course, students will understand all the primary programming constructs in REXX and how to meld these constructs into effective REXX programs. Students will also have knowledge of many advanced techniques, both those reviewed in class and additional sample programs. Each student will receive a floppy disk containing tested solutions to all the quizzes and labs, all examples and the advanced techniques sample programs.

Optionally, an instructor can be retained for additional days to provide onsite programming and systems expertise. The instructor can show hands-on techniques for providing solutions in a rapid manner, assisting in a one on one fashion with the more complicated aspects of the local environment. All instructors have extensive backgrounds in providing solutions to the systems needs of corporate clients and possess a diversity of skills, experience and industry contacts.

Prerequisite

Students must be familiar with use of the text editor on the programming platform. No prior programming experience is required, but students should have a basic understanding of data processing concepts.

Duration

Three days

Multi-Platform REXX Programming
Course Outline

1. REXX Overview
- A. What is REXX?
- B. REXX, Object REXX, NetRexx
- C. REXX Language Design Goals
- D. Portability: REXX Environments
- E. REXX vs. TSO CLIST
- F. REXX vs. BAT file
- G. Languages: Compiled vs. Scripting
- H. What is a REXX Procedure?
- I. REXX Syntax
- J. REXX Delimiters
- K. REXX Has Five Clause Types
- L. Types of Tokens Within A Clause
- M. REXX Comments
- N. Literal Strings
- O. Programming Style

2. First REXX Procedure
- A. REXX Keyword Instructions
- B. REXX Built-in Functions
- C. TSO/E REXX Keyword Instructions
- D. PC REXX Keyword Instructions
- E. The REXX Data Stack
- F. REXX Keyword Instructions
- G. Trace Output
- H. Lab: Trace
- I. An ISPF Feature: Syntax Hilighting

3. Running REXX
- A. REXX Operating Environments
- B. Running REXX in TSO/E
- C. Implicit execution: The easier way
- D. TSO/E Allocation Examples
- E. TSO/E Allocation Tools - ALTLIB, DDCONCAT, RUN
- F. Running REXX in NetView
- G. REXX in UNIX System Services
- H. Running REXX in VM
- I. Running REXX in Windows 9x etc.
- J. Running REXX execs w/ Regina:
- K. Running REXX in OS/2

4. Expressions and Operators
- A. Terms
- B. Operators
- C. Arithmetic Operators
- D. Standard Comparison Operators
- E. Strict Comparison Operators
- F. Logical (Boolean) Operators
- G. Operator Precedence
- H. String Concatenation
- I. Concatenation Operators

5. Assignments and Variables
- A. Variable Syntax
- B. Lab2: Adding 2 numbers
- C. Special REXX Variables
- D. REXX Keywords - PARSE VAR
- E. General Rules for Parsing
- F. PARSE Quiz
- G. Compound (Stem) Variables
- H. Stem Variables With Numeric Tails
- I. Stem Variables With Character Tails
- J. Variable related REXX Functions

6. REXX Keywords
- A. IF - THEN - ELSE
- B. Basic DO - The DO Group
- C. Repetitive DO
- D. More Iterative DOs
- E. Optional Lab: Counting with DO
- F. DO Forever
- G. Leave [control variable name]
- H. How to Stop Looping
- I. Lab3: Validating 2 numbers
- J. How to leave a Forever:
- K. REXX Keyword - Iterate
- L. Conditional DO - DO WHILE
- M. Conditional DO - DO UNTIL
- N. Optional Lab4: DO UNTIL
- O. SELECT Keyword
- P. Selecting cases with If - Then - Else
- Q. Optional Lab5: SELECT
- R. Optional Lab6: SELECT
- S. Keyword Instruction - Interpret
- T. Optional Lab99: Fun with Interpret
- U. REXX Keywords - Host Commands
- V. Basic TSO/REXX Environments
- W. Advanced TSO/REXX Environments
- X. CICS/REXX Environments
- Y. TSO Lab: Changing ADDRESS

7. Calling and Writing Subroutines and Functions
- A. Why write Functions & Subroutines?
- B. Characteristics of Well Designed Routines
- C. REXX Keywords
- D. Functions vs Subroutines
- E. Ways to Invoke a Routine
- F. Types of Routines
- G. Structured Programming
- H. Internal vs External Routines
- I. An Internal Function Call
- J. An Internal Subroutine Call
- K. REXX Keyword: RETURN
- L. Passing Data to/from a Routine
- M. REXX Keyword: Parse Argument
- N. SAA Functions ARG()
- O. REXX Keyword: Parse Value
- P. REXX Keyword: Parse Source
- Q. Parse Source - Example
- R. Recursive Function Calls
- S. Subroutine & Function Calls
- T. REXX Keyword: Expose
- U. Expose Quiz
- V. REXX Keyword: SIGNAL name

8. The Built-in SAA REXX Functions
- A. SAA Functions - Date
- B. SAA Functions - Time
- C. SAA Functions - Words
- D. SAA Functions - Characters
- E. SAA Functions - String Formatting
- F. SAA Functions - String Manipulation
- G. SAA Functions - New String Functions
- H. SAA Functions - String Manipulation
- I. SAA Functions - Data Validation
- J. SAA Functions - Report Formatting
- K. SAA Functions - Math
- L. SAA Functions - Min, Max
- M. SAA Functions - Random
- N. SAA Functions - Reverse
- O. SAA Functions - Character Translation
- P. SAA Functions - Misc
- Q. SAA Functions - Bit Manipulation
- R. SAA Functions - Data Conversion
- S. SAA Functions - Data Generation
- T. Optional Lab: The String Functions
- U. Optional Lab: Formatting a Report

9. More Keyword Instructions (Optional Chapter)
- A. REXX Keywords
- B. DROP & UPPER Keywords
- C. Nop Keyword
- D. REXX error handling
- E. Error Handling Functions
- F. Numeric Keyword
- G. SAA Functions - TRUNC()
- H. Parse Keyword
- I. Options Keyword

10. TSO/E Built-In Functions (Optional Chapter)
- A. Non-SAA Routines for TSO/E, VM
- B. TSO/E Built-in Functions
- C. OUTTRAP() Example
- D. Optional Lab: Using OUTTRAP()
- E. Optional Lab: Find VOLSER
- F. Example: LISTDSI() Function
- G. Example: MVSVAR()
- H. TSO/E Built-in Functions: SYSVAR
- I. Example: MVSVAR(), SYSVAR(), etc.
- J. TSO/E Functions - GETMSG()

11. IBM Object REXX for Windows Functions (Optional Chapter)
- A. Built-in Functions: Beep
- B. Built-in Functions: Directory
- C. Built-in Functions: Filespec
- D. Built-in Functions: Var
- E. REXX API Functions
- F. REXX Function Packages
- G. Understanding REXXUTIL
- H. REXXUTIL Character UI Functions
- I. REXXUTIL Disk & File Functions
- J. Optional Lab: Finding a Program
- K. Optional Lab: Tallying disk information
- L. REXXUTIL Misc. Functions
- M. REXXUTIL REXX Variable Functions
- N. REXXUTIL Windows API Functions
- O. REXXUTIL Printer Access Functions
- P. REXXUTIL - SYSINI()
- Q. REXXUTIL REXX Macro Functions
- R. Understanding Unicode
- S. Unicode Conversion Functions
- T. Windows Encryption Functions
- U. REXXUTIL Windows Semaphore Functions
- V. REXXUTIL OS/2 Workplace Shell Functions
- W. REXXUTIL Misc. OS/2 Functions

12. File I/O
- A. REXX I/O Characteristics
- B. I/O Handling for TSO
- C. Using EXECIO w/ large files
- D. I/O Handling for PCs & UNIX
- E. LINEIN()
- F. LINEOUT()
- G. LINES()
- H. CHARIN()
- I. CHAROUT()
- J. CHARS()
- K. High-Speed Character I/O
- L. REXX w/ Redirection & Piping
- M. I/O for PCs & UNIX: STREAM()
- N. STREAM() Function Examples
- O. Optional OS/390 Lab: Writing JCL
- P. Optional OS/390 Lab: Reading JCL
- Q. Optional Lab: Employee Database

13. PC-DOS REXX (Optional Chapter)
- A. Running PC-DOS REXX
- B. Testing PC-DOS REXX
- C. PC-DOS REXX Tips & Tricks
- D. Redirecting DOS cmd output
- E. Redirecting DOS command: Sample Output
- F. PC-DOS Built-in External Functions
- G. Directories
- H. Drives
- I. Files
- J. Cursor
- K. Screen
- L. Misc

14. NetView REXX Overview (Optional Chapter)
- A. NetView ADDRESS environments
- B. NetView Command Processors
- C. Selected NetView Functions

15. Advanced REXX: Stack & PARSE (Optional)
- A. The Data Stack (External Data Queue)
- B. Using the Data Stack and the EDQ
- C. Keywords to Stack & Unstack Data
- D. Using the QUEUED() Function
- E. Stack Mgmt Commands (TSO/E)
- F. Stack Mgmt Commands in TSO/E, VM
- G. TSO/E I/O Using the Stack
- H. WinNT/9x/OS/2 RxQueue Function
- I. The RXQUEUE Filter
- J. RXQUEUE() Example
- K. Preserving the Stack
- L. Preserving the Stack (Portable Solution)
- M. PARSE Keyword: Data Sources
- N. General Rules for Parsing
- O. PARSE VAR
- P. Advanced Parse
- Q. Technique: Destructive Parse