Advanced REXX Programming
Course Summary

Description

This course is designed to enable the attendee to write advanced REXX procedures which interface to the Operating System in various ways, including File I/O, the use of ISPF panels and dialogs, edit macros, JCL modifications, submitting JCL using the stack and the MVS internal reader and complex parsing templates using table variables. A direct focus is placed on advanced REXX coding and debugging techniques. The student will learn to create table arrays and the data stack to manage and sort data. The student will also be able to utilize the ISPF environment to control menus and dynamic areas, and to operate the edit macro environment. This course covers REXX interfaces to other MVS environments, such as CICS, DB2 and UNIX System Services. In addition, use of REXX in Object REXX environments is provided as an optional topic.

Topics

- Reviewing REXX Basics
- Understanding the REXX Stack & Advanced Parse
- Building & Using Subroutines
- REXX and ISPF
    - Dialog Development
    - Edit macros
- REXX and TSO/E
- REXX and UNIX System Services
- Object REXX-specific Functions
- TCP/IP Socket Programming in REXX
- REXX and VSAM
- REXX and DB2
- REXX and CICS
- REXX and SAS
- IBM REXX Compiler
- Using REXX with Pro/JCL

Prerequisites

Students should have completed a basic REXX course, such as the ProTech *REXX Programming in a Multi-platform Environment* class or possess equivalent knowledge.

Duration

Two days (can easily be extended to three days to allow more lab time, facilitate additional review, and allow more depth to the coverage of optional topics).

Advanced REXX Programming - Course Outline

1. **Reviewing REXX Basics**
   A. REXX Syntax
   B. REXX Clause Types
   C. REXX Keywords
   D. Expressions
   E. Functions and Subroutines
   F. Simple and Stem variables
   G. The REXX Stack
   H. Command Interface
   I. File I/O
   J. Error handling

2. **Advanced Topics: The Stack & PARSE**
   A. Data Stack (External Data Queue)
   B. Keywords to Stack & Unstack Data
   C. Using the QUEUED() Function
   D. Stack Mgmt Commands (TSO/E)
   E. Stack Mgmt Commands in TSO/E, VM
   F. TSO/E I/O Using the Stack
   G. WinNT/9x/OS/2 RxQueue Function
   H. The RXQUEUE Filter
   I. RXQUEUE() Example
   J. Preserving the Stack (Portable)
   K. PARSE Keyword: Data Sources
   L. Advanced PARSE
   M. Technique: Destructive Parse

3. **Building & Using Subroutines**
   A. Functions vs Subroutines
   B. Types of Routines
   C. Structured Programming
   D. Internal vs External Routines
   E. Recursive Function Calls
   F. Subroutine & Function Calls
   G. Characteristics of Well Designed Routines
   H. Building a Subroutine Toolkit
   I. Validating a Value Against a List
   J. Handling Strings with Punctuation
   K. Host Command Wrappers
   L. General Message Send Routines
   M. Strings with Embedded Quotes
   N. Checking for Existence of a Table
   O. Checking for Dataset Existence

4. **REXX and ISPF**
   A. ISPF Considerations
   B. ISPF Edit Models
   C. ISPF Panels
   D. ISPF Development Tools
   E. ISPF Dialog Example - Math
   F. Example ISPF Panels
   G. ISPF Messages - (ISPMLIB)
   H. ISPF Edit Macros
   I. ISPF Edit Macro Example
   J. ISPF Edit Macros - Line labels
   K. ISPF Editor-Assigned Line labels
   L. ISPF Edit Macro Cmds
   M. ISPF Edit Macros - Example
   N. ISPF Edit Macros - ISPF Msgs
   O. ISPF Generic Msg Example

5. **REXX and TSO/E**
   A. TSO/E EXECUTIL cmd
   B. Non-SAA Routines for TSO/E, VM
   C. TSO/E Built-in Functions
   D. OUTTRAP() Example
   E. Optional Lab: Using OUTTRAP()
   F. Optional Lab: Find VOLSER
   G. TSO/E Built-in Functions
   H. Example: LISTDSI() Function
   I. Example: MVSVAR()
   J. TSO/E Built-in Functions: SYSVAR
   K. Example: MVSVAR(), SYSVAR(), etc.
   L. TSO/E Functions - GETMSG()
   M. Issuing Console Cmds via AutoOperator
   N. Running REXX from Batch
   O. Submitting JCL From REXX
   P. Submitting JCL with Substitution
   Q. Submitting JCL without Data sets
   R. Basic TSO/REXX Environments
   S. Advanced TSO/REXX Environments
   T. Program Linkage Using REXX
   U. REXX Program Linkage Options
   V. REXX Program Linkage Details

6. Using REXX with UNIX System Services
- A. Why Use REXX in z/OS UNIX?
- B. REXX in UNIX System Services
- C. USS REXX Address Environments
- D. Sample USS REXX Program
- E. REXX in TSO vs. REXX in Shell
- F. Understanding SYSCALLS()
- G. Bi-Modal USS REXX Program
- H. ADDRESS SYSCALL: Files & File I/O
- I. ADDRESS SYSCALL: File System Info
- J. ADDRESS SYSCALL: Directories
- K. ADDRESS SYSCALL: Processes
- L. ADDRESS SYSCALL: Signal handling
- M. ADDRESS SYSCALL: Security
- N. ADDRESS SYSCALL: Miscellaneous

7. IBM Object REXX Built-In Functions
- A. Built-in Functions: Beep
- B. Built-in Functions: Directory
- C. Built-in Functions: Filespec
- D. REXX API Functions
- E. Understanding REXXUTIL
- F. REXXUTIL Character UI Functions
- G. REXXUTIL Disk & File Functions
- H. Optional Lab: Finding a Program
- I. Optional Lab: Tallying disk information
- J. REXXUTIL Misc. Functions
- K. REXXUTIL REXX Variable Functions
- L. REXXUTIL Windows API Functions
- M. REXXUTIL Windows Semaphores
- N. REXXUTIL REXX Macro Functions
- O. REXXUTIL OS/2 Workplace Shell
- P. REXXUTIL Misc. OS/2 Functions
- Q. REXXUTIL - SYSINI() in OS/2

8. TCP/IP Socket Programming in REXX
- A. TCP/IP Application Functionality
- B. Two Similar Packet Delivery Systems
- C. Packet Routing
- D. Network Physical Layer, IP Layer
- E. OS/390 TCP/IP Sockets
- F. Well Known UDP & TCP Ports
- G. TCP/IP Services
- H. TCP/IP Applications
- I. Telnet Access
- J. TCP/IP Applications: rsh, rexec
- K. FTP (File Transfer Protocol)
- L. Socket Programming Overview
- M. Basic Socket Send/Receive Functions
- N. Resolver, Port number
- O. Basic Socket Connection Functions
- P. Sample TCP/IP Socket Application

9. VSAM and REXX
- A. VSAM Dataset Overview
- B. VSAM & REXX- Possible Solutions
- C. IDCAMS Example
- D. Third-party VSAM Interfaces
- E. MAX/REXX Example
- F. REXXTOOLS/MVS Example
- G. OPS/MVS OPSVSAM() Function
- H. REXX Freeware: RXVSAM()
- I. RXVSAM() Syntax
- J. RXVSAM() Example: Load a KSDS
- K. RXVSAM() Example: Read a KSDS
- L. Installing RXVSAM()

10. DB2 and REXX
- A. DB2 & REXX - The Problem
- B. DB2 Version 6 Refresh - the Solution
- C. DB2 REXX: ADDRESS DSNREXX
- D. DB2 REXX: Defining DSNREXX
- E. DB2 Definitions
- F. Selected SQLCA Fields
- G. Error Checking using the SQLCA
- H. Understanding rc vs. sqlcode
- I. Embedding SQL into REXX
- J. More DB2 Definitions
- K. Using an SQL Cursor
- L. Pre-defined SQL Cursors
- M. Example using cursor, SQLDA
- N. DB2 Isolation Levels
- O. Writing a REXX stored procedure
- P. Additional DSNREXX Information
- Q. DB2 & REXX - Additional Solutions
- R. Roll-your-own DB2 REXX API
- S. $DB2CMD REXX function
- T. Third-party REXX-DB2 Interfaces
- U. MAX/REXX Example

11. REXX and CICS
- A. REXX/CICS Overview
- B. REXX/CICS Native CICS Editor
- C. REXX/CICS Client/Server Support
- D. REXX/CICS DB2 Client Example
- E. REXX/CICS DB2 Server Example
- F. System and User Profile Execs
- G. Shared Execs in Virtual Storage
- H. REXX/CICS Environments
- I. REXX/CICS DB2 Interface Example
- J. REXX/CICS EXECIO Example

12. SAS and REXX
- A. SAS - REXX Interaction
- B. REXX and SAS
- C. SAS Options

13. IBM REXX Compiler
- A. REXX Compiler Features
- B. REXX Compiler PROCs
- C. Example Compile, Link, Go

14. Using REXX with Pro/JCL
- A. What is PRO/JCL?
- B. PRO/JCL and MVS
- C. JCL - Before
- D. JCL - After PRO/JCL Scanning
- E. JCL - After RESET
- F. PRO/JCL Product Interfaces
- G. Utility Program Syntax & Emulation
- H. PRO/JCL Edit Macro Summary
- I. JJ Edit Macro Parameters
- J. PRO/JCL REXX Interface
- K. Sample PRO/JCL REXX exec
- L. REXX: Statement Data Extraction
- M. REXX: Statement Data Change
- N. REXX: Control and Test