ProTech
protechtraining.com

"Charting the Course ...

... to Your Success!"

Comprehensive REXX Programming for z/OS

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.

TSO, Batch, NetView, and z/OS UNIX 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 lab 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.

Topics

- REXX Overview
- First REXX Procedure
- Running REXX
- Expressions and Operators
- REXX Variables
- REXX Keywords
- Calling and Writing Subroutines and Functions
- Built-in Functions
- More Keyword Instructions
- TSO/E Specific Built-In Functions
- File I/O
- NetView REXX
- Using REXX with z/OS UNIX System Services
- Rexx and ISPF
- DB2 and REXX
- REXX and CICS
- VSAM and REXX
- Advanced REXX Topics: The Stack & PARSE
- Advanced TSO/E REXX

Prerequisites

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

Five days

ProTech
protechtraining.com

"Charting the Course ...

... to Your Success!"

Comprehensive REXX Programming for z/OS

Course Outline

I. 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. Comments
- M. Block comments - can span lines
- N. Literal Strings
- O. Literal String Examples
- P. Programming Style
- Q. Example Programming Style
- R. Written Exercise
- S. Written Exercise – Answers

II. 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 Data Stack (External Data Queue)
- F. Using the REXX Data Stack
- G. REXX Keyword Instructions
- H. Trace Output
- I. Lab: Trace
- J. An ISPF Feature: Syntax Hilighting

III. Running REXX
- A. REXX Operating Environments
- B. Running REXX in TSO/E
- C. Where TSO/E REXX execs live
- D. Implicit execution: The easier way
- E. TSO/E Allocation Differences
- F. TSO/E Allocation Examples
- G. TSO/E Allocation Tools - ALTLIB
- H. TSO/E Allocation Tools - DDCONCAT
- I. TSO/E Allocation Tools - RUN
- J. Running REXX from Batch - IKJEFT1A
- K. Running REXX from Batch - IRXJCL
- L. Running OPS/REXX via TSO Cmds
- M. Running MVS System REXX
- N. Running REXX in NetView
- O. REXX in UNIX System Services
- P. Running REXX in VM
- Q. Running ooRexx in Windows
- R. Running a Windows ooRexx exec:
- S. Running REXX execs w/ Regina:

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

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

VI. 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. The Last DO (that you'll ever do!)
- G. Leave [control variable name]
- H. How to Stop Looping in TSO/E
- I. How to Stop a Loop in PC REXX
- J. Lab3: Validating 2 numbers
- K. How to leave a Forever:
- L. REXX Keyword - Iterate
- M. Conditional DO - DO WHILE
- N. Conditional DO - DO UNTIL
- O. Optional Lab4: DO UNTIL
- P. SELECT Keyword
- Q. Selecting cases with If - Then - Else
- R. SELECT Example

Comprehensive REXX Programming for z/OS

Course Outline (cont'd)

S. Guidelines for Using SELECT vs. IF
T. Optional Lab5: SELECT
U. Optional Lab6: SELECT
V. Keyword Instruction - Interpret
W. Example - Interpret Keyword
X. Optional Lab99: Fun with Interpret
Y. REXX Keywords - Host Commands
Z. Basic TSO/REXX Environments
AA. Advanced TSO/REXX Environments
BB. CICS/REXX Environments
CC. TSO Lab: Changing ADDRESS

VII. 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 Arg
N. SAA Functions ARG()
O. REXX Keyword: Parse Value
P. REXX Keyword: Parse Source
Q. Parse Source - Example
R. Recursive Function Calls
S. Overriding Built-in Functions
T. Subroutine & Function Side Effects
U. Exposing Variables in a Routine
V. REXX Keyword: Expose
W. Procedure Quiz
X. REXX Keyword: SIGNAL name

VIII. Built-in Functions

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

IX. More Keyword Instructions

A. REXX Keywords
B. DROP & UPPER Keywords
C. NOP Keyword
D. Error Handling: Call on Error
E. Error Handling: Signal on Error
F. Error Handling Functions
G. Error Handling Lab:
H. Numeric Keyword
I. SAA Functions TRUNC()
J. Parse Keyword
K. Options Keyword

X. TSO/E Specific Built-In Functions

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. TSO/E Built-in Functions
G. Example: LISTDSI() Function
H. Example: LISTDSIX Output
I. TSO/E Built-in Functions
J. Example: MVSVAR()
K. TSO/E Built-in Functions: SYSVAR
L. Example: MVSVAR(), SYSVAR(), etc.
M. TSO/E Built-in Functions
N. TSO/E Functions - GETMSG()

XI. File I/O

A. REXX I/O Characteristics
B. I/O Handling for TSO: EXECIO
C. EXECIO Examples
D. Update in place with EXECIO
E. Using EXECIO w/ large files
F. I/O Handling for PCs & UNIX

Comprehensive REXX Programming for z/OS

Course Outline (cont'd)

G. I/O for PCs & UNIX: LINEIN()
H. I/O for PCs & UNIX: LINEOUT()
I. I/O for PCs & UNIX: LINES()
J. I/O for PCs & UNIX: CHARIN()
K. I/O for PCs & UNIX: CHAROUT()
L. I/O for PCs & UNIX: CHARS()
M. High-Speed Character I/O
N. REXX w/ Redirection & Piping
O. I/O for PCs & UNIX: STREAM()
P. STREAM() Function Examples
Q. STREAM() Function Sample Program
R. Optional MVS Lab: Writing JCL
S. Optional MVS Lab: Reading JCL
T. Optional Lab: Employee Database

XII. NetView REXX

A. NetView REXX Overview
B. NetView ADDRESS environments
C. NetView Command Processors
D. Pipe Processing in NetView
E. Pipeline Processing Concepts
F. NetView pipe Command
G. Pipe Command Example
H. Overview of Pipe Stages
I. Pipe Stages: Filtering
J. Pipe Stages: Data Manipulation
K. Pipe Stages: Plumbing
L. Pipe Stages: Commands
M. Pipe Stages: File I/O
N. Pipe Stages: Output and Logging
O. Pipe Stages: Programming
P. Pipe Stages: NetView Related
Q. Pipe Stages: Miscellaneous
R. Selected NetView Functions

XIII. Using REXX with z/OS 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: File Manipulation
I. ADDRESS SYSCALL: File I/O
J. ADDRESS SYSCALL: File System Information
K. ADDRESS SYSCALL: Directory Manipulation
L. ADDRESS SYSCALL: Process Manipulation
M. ADDRESS SYSCALL: Signal handling
N. ADDRESS SYSCALL: Security-Related
O. ADDRESS SYSCALL: Miscellaneous

XIV. Rexx and ISPF

A. ISPF Overview
B. ISPF Documentation
C. ISPF Development Tools
D. ISPF Dialog Elements
E. Dialog Elements: Variables
F. Dialog Elements: Variable Pools
G. Dialog Elements: Panels
H. Dialog Elements: Message Definitions
I. Dialog Elements: File Tailoring Skeletons
J. Dialog Elements: Tables
K. ISPF Dialog Example - Math
L. ISPF Edit Models
M. ISPF Generic Messages
N. SETMSG Service
O. ISPF Generic Msg Example
P. Edit Model lab
Q. ISPF Return Code Processing
R. ISPF Data Definition (DD) Names
S. ISPF Facilities: LIBDEF command
T. ISPF DISPLAY Service
U. TBDISPL Service
V. Menu Panel Example: EMPMENU
W. Input Panel Example: EMPIN
X. Table Display Panel Example: EMPLIST
Y. ISPMLIB Messages Example: EMP00
Z. ISPF Dialog lab
AA. ISPF Edit Macros
BB. ISPF Edit Macro Example
CC. ISPF Edit Macros - Line labels
DD. ISPF Editor-Assigned Line labels
EE. ISPF Edit Macro Cmds
FF. ISPF Edit Macros – Example

Comprehensive REXX Programming for z/OS

Course Outline (cont'd)

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

XVI. 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

XVII. 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()

XVIII. Advanced REXX Topics: The Stack & PARSE
- 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. Windows 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

XIX. Advanced TSO/E REXX
- A. TSO/E EXECUTIL cmd
- B. TSO/E Example
- C. Running REXX from Batch
- D. Submitting JCL From REXX
- E. Submitting JCL with Substitution
- F. Submitting JCL without Data sets
- G. Basic TSO/REXX Environments
- H. Advanced TSO/REXX Environments
- I. Program Linkage Using REXX
- J. REXX Program Linkage Options
- K. REXX Program Linkage Details