Learn to Program in Rexx
Hands-On Lab Part 2 of 2

Session 7486
August 5, 2010

Thomas Conley
Pinnacle Consulting Group, Inc. (PCG)
59 Applewood Drive
Rochester, NY 14612-3501
P: (585)720-0012
F: (585)723-3713
pinncons@rochester.rr.com
http://home.roadrunner.com/~pinncons/

TSO/Rexx execs are members of a PDS allocated to SYSPROC or SYSEXEC DD statements in TSO. Rexx is automatically invoked for execs in SYSEXEC, but to distinguish Rexx execs from CLISTs in SYSPROC, the first line of exec must contain the comment:

```
/* rexx */
```

The ALTLIB command is used to dynamically add a dataset to the SYSPROC or SYSEXEC concatenation:

```
address tso "ALTLIB ACT APPL(CLIST) DA('SYS1.SEDGEXE1')"
```

TSO execs can run in batch using program IKJEFT01. Following is sample JCL to execute exec STRVSPAR:

```
//IBMUSERT JOB 'CONLEY',CLASS=A,NOTIFY=&SYSUID,MSGCLASS=X
//TSO      EXEC PGM=IKJEFT01,
//              REGION=0M
//SYSPROC  DD  DISP=SHR,DSN=IBMUSER.REXX.EXEC
//SYSTSPRT DD  SYSOUT=*
//SYSTSIN  DD  *
%STRVSPAR
/*
```

Rexx uses "address" to issue commands to various external environments. Some of the command environments in TSO include: tso, ispexec, isredit, console, mvs, link, and attach. Use the tso environment to issue TSO commands:

```
address tso "ALLOC FI(OUTPUT) DA('USER.OUTPUT') SHR"
```

BPXWDYN is an alternate method for allocating files, which allows batch REXX execs to run under IRXJCL instead of IKJEFT01 (IRXJCL can perform better if you don't require other TSO services). To use BPXWDYN, set a variable to the TSO allocation command string, then call BPXWDYN passing the TSO allocation command string as the argument:

```
allocstr = "ALLOC FI(OUTPUT) DA('USER.OUTPUT') SHR"
call bpxwdyn(allocstr)
```

EXECIO performs file I/O (open, read, write, close, etc.). The syntax is:

"EXECIO <rec-cnt> <diskop> <ddname> (<options>)"

Some examples:

```
"EXECIO 0 DISKR INFILE (OPEN)"  /* read no recs, open file
*/
"EXECIO 0 DISKR INFILE (FINIS)" /* read no recs, close
file */
"EXECIO 1 DISKR INFILE (STEM INREC.)" /* read 1 record and
place in stem inrec. */
"EXECIO * DISKW OUTFILE (OPEN STEM OUTRECS. FINIS)" /* open
file, write all records in stem outrecs. and close file */
```

This code will display a file:

```
/* rexx */
address tso "ALLOC FI(INFILE) DA('USER.DATASET') SHR"
"EXECIO * DISKR INFILE (OPEN STEM INFILE. FINIS)"
address tso "FREE FI(INFILE)"
do i = 1 to infile.0
   say infile.i
end
```

Outtrap traps command output stores the output in a stem variable. This code uses outtrap to process a LISTCAT command:

```
x = outtrap('listcato.')
address tso "LISTCAT ENT('SYS1.PARMLIB') ALL"
x = outtrap('OFF')
do i = 1 to listcato.0
   say listcato.i
end
drop listcato.
```

ALWAYS turn off outtrap after issuing the command, and DROP outtrap variable when finished to save storage .

SYSVAR and MVSVAR are TSO/REXX functions that return various information about the operating system and TSO user environment:

`x = sysvar('SYSJES')` returns the JES type and release
`x = mvsvar('SYSMVS')` returns the MVS level of the system
`x = mvsvar('SYSNAME')` returns the SMFID of the system

A full list and description of each SYSVAR and MVSVAR can be found in the TSO/REXX Reference.

LISTDSI is a TSO/REXX function that returns all the data regarding a dataset and stores the data in variables:

`x = LISTDSI('SYS1.PARMLIB')` lists SYS1.PARMLIB data

SYSDSNAME holds the dataset name
SYSVOLUME holds the volume name
SYSBLKSIZE holds the blocksize

A full list and description of each LISTDSI variable can be found in the TSO/REXX Reference.

For manuals, go to:

http://www-1.ibm.com/servers/eserver/zseries/zos/bkserv/

If you have any questions, raise your hand and we'll come to you

Before you leave:

- DO NOT CHANGE THE TSO PASSWORD!!
- If you want a copy of your work, you must have a diskette or USB thumb drive
- Logoff Windows – Do NOT power off!
- Fill out a session evaluation sheet and leave it at the front of the lab
- Pick up an answer set at the front of the lab

Getting Started

1) Logon to TSO (userid and password provided by instructor)
2) Get into ISPF
3) Get into ISPF Edit
4) Split your Screen
    a) Use START or PF2
5) Enter ISPF Option 6
6) Then issue:

  %SETREXX
    (issues altlib activate appl(exec) dataset(rexx.exec))
7) You will create/edit the Lab Exercises in the Edit session and Test them in the Option 6 session
8) Note: This library will only work in the ISPF Session (split) in which the ALTLIB is issued) so you may need to do this on each ISPF split.
9) Create and edit your Rexx exec's in <userid>.REXX.EXEC.

Exercises

These exercises will build upon the information in the session handout that you can use for reference. Solutions will be given out at the end of the session.

Exercise 1

For this exercise create member EX1 in your REXX library.
Goal: Display the name of the current system name and level of OS/390.
Hint: Use the Say command and the MVSVAR function.
Save the REXX program and then run it by issuing from the ISPF Command Prompt TSO %EX1 or split the screen and option ISPF option 6 and issue %EX1.

Exercise 2
For this exercise create member EX2 in your REXX Library.
Goal: Issue any TSO Command from within a REXX Program.
e.g. LISTD 'SYS1.PARMLIB'
Hint: Use ARG or PARSE ARG to get the desired TSO command and options from the command line.
Note: A variable will be processed before it is executed.
Save the REXX program and run it.

Exercise 3
For this exercise create member EX3 in your REXX Library.
Goal: The same as Exercise 2 but this time capture the output and display it inside a loop.
Hint: Use OUTTRAP and a DO END loop.
Save the REXX program and run it.

Exercise 4
For this exercise create member EX4 in your REXX Library.
Goal: The same as Exercise 3 but this time instead of displaying the results using a loop we will use the ISPF Browse service.
Hint: Use TSO Allocate and Free. Use EXECIO to write the trapped results. Use Address ISPEXEC Browse to view it.
Save the REXX program and run it.

Exercise 5
For this exercise create member EX5 in your REXX Library.
Goal: Test the existence of a data set passed to the program and display information about it.
Hint: Save the status (from SYSDSN) in a variable to test and then use LISTDSI for more information.
Save the REXX program and run it.

Exercise 6
For this exercise create member EX6 in your REXX Library.
Goal: Allocate and read in the file SHAREAxx.REXX.DATAFILE (where xx is your 2-digit userid suffix). The file is in the following format:

Lastname Firstname areacode phone number

Parse the data and echo it back in the following format:

Lastname, Firstname (xxx) yyy-yyyy

Hint: Use either TSO ALLOC or BPXWDYN to allocate the file, EXECIO * or EXECIO 1 DISKR to read the file, and “say” to echo the output.
Save the REXX program and run it.