REXXTOOLS/MVS

EARL D. HODIL
CHICAGO-SOFT

REXXTOOLS/MVS

A Toolkit for MVS Programmers

REXX Symposium
Annapolis, Maryland
May 5, 1992

Earl D. Hodil
Chicago-Soft, Ltd.
45 Lyme Road, #307
Hanover, New Hampshire 03755
Phone: (603) 643-4002
FAX: (603) 643-4571
MCIMAIL: CHISOFT

Introduction

What is REXXTOOLS/MVS?

- REXXTOOLS is a collection of assembler-based functions and utilities designed to help the REXX programmer be more productive.

Who will use REXXTOOLS/MVS?

- Application programmers - ISPF Dialogs, batch jobs, etc. for end-users.
- System programmers - function packages, utilities for themselves and application programmers.

Introduction

REXXTOOLS Components (REXX perspective):

* REXX function package - 29 new functions
* REXX host command environment - ADDRESS REXX
* REXX compiler - encapsulates REXX programs in standalone load modules
* Sample applications - TSO utilities, programming examples

Introduction

REXX Functions:

- `ABC = MYFUNC(ARG1, ARG2)`
- ARG1 & ARG2 are arguments passed to the function
- MYFUNC returns a value
- MYFUNC could have side-effects (i.e., set other variables in the caller's variable pool)

REXX Subroutines:

- `CALL MYFUNC ARG1 ARG2`
- Arguments have the same meaning as for functions
- Can optionally return a value (RESULT special variable)

Introduction

How are functions and subroutines developed?

- **REXX - internal and external subroutines and functions**
- **Compiled/assembled languages (a la REXXTOOLS)**
- **Search order**
- **Function packages**
  - **Groups related functions together**
  - **Pre-loaded at environment initialization**
  - **Can't be developed in REXX (unless you have REXX compiler!)**

Introduction

REXX Host Commands:

- **any expression not identified as a language construct**

```
"VGET (ZUSER) SHARED"
```

- **expression is evaluated and string is passed to host command environment routine.**
- **ADDRESS instruction is used to switch host command environments.**

```
ADDRESS TSO "LISTA ST H"
```

- **Each REXX environment has a default host command environment**
- **Parameters module host command environment table maps environment names to routines.**

VSAM Functions

Why VSAM?

* **Lots of existing VSAM files**
* **Better for multi-user applications than ISPF Tables**

What is supported?

* **All VSAM dataset organizations**
  - **Key-Sequenced Data Set**
  - **Entry-Sequenced Data Set**
  - **Relative Record Data Set**
  - **Linear Dataset (sort of)**
* **Interface is patterned after DFP macros**

VSAM Functions

Opening and Closing VSAM Datasets:

* **CALL OPEN 'VSAM', ddname [,acboptions]**
* **CALL CLOSE 'VSAM', ddname**
* **ACB options string**

  "(ADR,SEQ,NDF)"

Reading and Writing Records:

* **CALL GET ddname [,key] [,rploptions]**
* **CALL PUT ddname, record [,key] [,rploptions]**
* **RPL options string**

  "(KEY,DIR,GEN)"

Deleting Records:

* **CALL ERASE ddname**

VSAM Functions

Other VSAM Functions

* CALL ENDREQ ddname
* CALL POINT ddname [,key] [,rploptions]
* CALL VERIFYV ddname

VSAM Functions

ACB Options:

- Most ACB options are supported:

  ADR, CNV, KEY, DIR, SEQ, SKP, IN, OUT, DFR, NDF, NIS, SIS, NRM, AIX, NRS, RST

* Stay in effect from OPEN to CLOSE

RPL Options:

* Most RPL options are supported:

  ADR, CNV, KEY, DIR, SEQ, SKP, ARD, LRD, FWD, BWD, NSP, NUP, UPD, KEQ, KGE, FKS, GEN

- Shared between calls for each ddname.

* Stay in effect until changed

VSAM Functions

Special Variables:

* **Used to return information from functions**
* **RC and REASON - usually straight from VSAM**
* **OPEN:**
  - **$RXTTYPE**
  - **$RXTLRCL**
  - **$RXTCNVL**
  - **$RXTKEYO**
  - **$RXTKEYL**
  - **$RXTRECS**
  - **$RXTHRBA**
  - **$RXTERBA**

VSAM Functions

Special Variables (continued)

* **GET/PUT:**
  - **$RXTKEY**
  - **$RXTRBA**
  - **$RXTRECL**

VSAM Functions

Sample REXX program:

```
/* REXX */
  ADDRESS TSO "ALLOC FI(RXTKSDS) DA(RXTKSDS.DATA) SHR REU"
  CALL OPEN 'VSAM', 'RXTKSDS', '(KEY,DIR,IN)'
  CALL TPUT "ENTER KEY OR 'END':", 'ASIS'
  KEY = TRANSLATE(TGET(, 'WAIT'))
  DO WHILE KEY <> 'END'
    CALL GET 'RXTKSDS', KEY, '(DIR,GEN,KEY)'
    IF RC <> 0 THEN
      SAY 'NO MATCH FOR KEY='KEY' FOUND.'
    ELSE
      SAY 'RECORD='RESULT
    CALL TPUT "ENTER KEY OR 'END':", 'ASIS'
    KEY = TRANSLATE(TGET(, 'WAIT'))
  END
  CALL CLOSE 'VSAM', 'RXTKSDS'
  ADDRESS TSO "FREE FI(RXTKSDS)"
EXIT
```


MVS Supervisor Services

Access to System Services

- **Patterned after and interfaces to application macros**
- **Problem state only**
- **All are task related**
- **Functional areas:**
  - **Virtual Storage Management**
  - **Resource Control**
  - **Security**
  - **Operator Communication and logging**

MVS Supervisor Services

Virtual Storage Management:

STGAD = GETMAIN(amount [,subpool] [,loc] [,bndry] [,fill])

CALL FREEMAIN addr, amount [,subpool])

Uses:

- **Communicating with non-function asm programs**
- **Multi-tasking REXX application inter-task communication**

MVS Supervisor Services

Resource Control

- **Problem: How to share a resource between asynchronous processes?**
- **Reserving and freeing an arbitrary resource:**

  CALL ENQ major, minor [,control] [,scope] [,reqtype]

  CALL DEQ major, minor [,scope] [,reqtype]
- **Halting execution until conditions are right:**

  CALL WAIT 'ECB', ecbad [,longwait]

  CALL WAIT 'ECBLIST', ecblad [,eventno] [,longwait]

  CALL WAIT 'SEC', seconds
- **Signaling event completion:**

  CALL POST ecbad [,compcode]

MVS Supervisor Services

Securing a resource:

* System Authorization Facilty (SAF)
  - Works with major security packages (RACF, ACF2, etc.)
  - Router Table must be set up
  - SAF must be active
* Problem state only - can't counterfeit userid
* Modelled after RACROUTE macro:

  CALL RACROUTE 'AUTH', entity, [,class] [,attr] [,dstype]
  [,volser] [,oldvol] [,appl] [,owner] [,acclv] [,racfind]
  [,generic] [,reqstor] [,subsys]

MVS Supervisor Services

Operator Communication and Event Logging:

* Single and Multi-line console messages:

  wtoid = WTO(msgtext [,msgcount] [,route] [,desc])
  - msgcount > 0 uses multi-line format
  - no direct control of routing and highlighting

  CALL DOM wtoid
  - removes non-scrollable messages
  - not an error if message is already gone

* Two-way communication:

  CALL WTOR msgtext [,waitsecs] [,route]
  - Does wait internally
  - Handy for batch jobs

MVS Supervisor Services

Operator Communication and Event Logging (continued):

* **Logging events:**

  CALL WTL msgtext

  - **Fast way to keep track of program execution**

TSO Services

Input/Output Functions:

* **REXX SAY instruction is limited**
  - **PUTLINE only**
  - **No formatting control**
* **REXX PULL instruction:**
  - **GETLINE only**
  - **Does have nice parsing**
  - **Complicates matters when using the data stack**
* **ISPF Dialog Manager**
  - **Must be under ISPF command to use**
  - **Can't use in certain environments like TEST**

TSO Services

REXXTOOLS TPUT

* **CALL TPUT string [,tptype] [,tpwait] [,tphold] [,tpbreak]**
* **Line mode:**
  - **tptype: 'ASIS' or 'EDIT'**
  - **ASIS like CLIST WRITENR:**

```
CALL TPUT 'Enter Your Name:', 'ASIS'
```

  - **No echo prompting:**

```
CALL TPUT 'Enter Your Password:'||'24'X,,
   'ASIS'
```


TSO Services

REXXTOOLS TPUT

* **Full-screen mode:**
  - **tptype: 'NOEDIT' or 'FULLSCR'**
  - **string argument contains 3270 data stream:**

```
DS = 'F5C3'X||SBA(1,1,80)||'1DF8'X||'ENTER YOUR NAME ===>'||,
     '1DC813'X||SBA(1,40)||'1DF8'X
CALL TPUT DS, 'FULLSCR'
```

Miscellaneous Services

Stem Handling Functions

- REXX stem variables - i.e., variables with a dot
- Sorting arrays (stems) with numeric subscripts:

  CALL STEMSORT *stemname* [,startsub] [,stemcount] [,sortfields]

  - sortfields like DFSORT or SYNCSORT

    "(start,length,type,direction)"

  - Uses Heapsort algorithm (see Wirth)
- Displaying arrays with numeric subscripts:

  CALL STEMDISP 'BROWSE', stemname [,startsub] [,stemcount] [,title] [,panel]

  - Uses BRIF service for display (no dataset)

 23

Miscellaneous Services

String Handling Functions:

- Difficult parsing
  - Parsing where the location and frequency of the delimiters is difficult to predict.
  - Example:

    DSN('abc.efg(one)') KEYWORD2(two)

  - CALL PARSETOK string, stemname [,nbd] [,blankopt] [,dropopt]

 24

Miscellaneous Services

PARSETOK Example:

```
STRING = 'DSN(ABC) NONAME'
CALL PARSETOK STRING, "TOK.", "()", "BLANKS"
/* TOK.0 = 6; TOK.1 = 'DSN'; TOK.2 = '(';
TOK.3 = 'ABC'; TOK.4 = ')'; TOK.5 = ' ';
TOK.6 = 'NONAME' */
```


Miscellaneous Services

String Handling Functions (continued):

- **Sorting words**
  - **CALL WORDSORT string [,diropt]**
  - **diropt - Asending or Descending**
  - **Useful for sorting indexes into arrays:**

```
A.C = 5
A.A = 1
A.B = 2
INDEX = 'C A B'
INDEX = WORDSORT(INDEX)
/* INDEX = 'A B C' */
DO I = 1 TO WORDS(INDEX)
  SAY VALUE("A."WORD(INDEX,I))
END
```

Miscellaneous Services

MVS/Quick-Ref Function:

CALL QWIKREF fastpath, stemname [,maxlines] [dropopt]

- **fastpath just like QW command:**

  topic=item

  "M=IEF450I"

- **Possible use: trouble ticket automation**

Miscellaneous Services

Conversions:

- **Useful when working with existing VSAM files**
- **370 Packed decimal to REXX decimal**

  CALL P2D packnum [,scale]

```
PACKNUM = '1020000C'X
PRINTNUM = P2D(PACKNUM,2)
/* PRINTNUM = 10200.00 */
```

- **REXX decimal to Packed decimal**

  CALL D2P printnum [,n]

```
PACKNUM = D2P(100.45,5)
/* PACKNUM = '000010042C'X */
```

ADDRESS REXX

* **Issuing a command:**

```
ADDRESS REXX
"THIS IS A HOST COMMAND"
```

* **RXTADDRX REXX program**

```
/* REXX */
SAY ARG(1)
RETURN 4
```

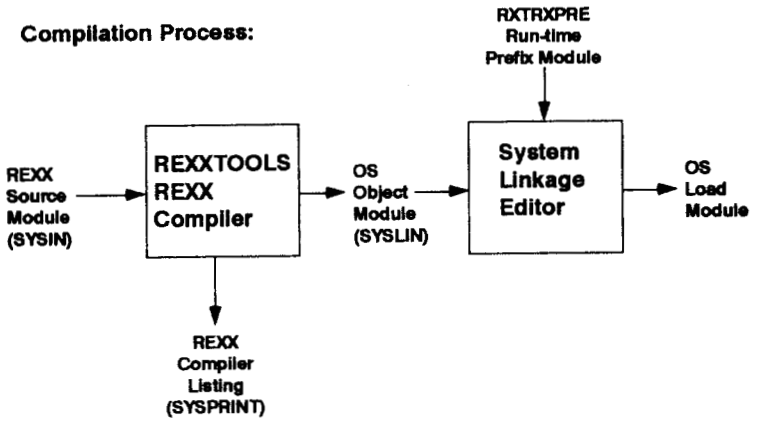
* **Argument: host command string**
* **Must return numeric return code**
* **Limitation: no way to access calling program's variables**

REXX Compiler

* **Compiles REXX programs into standalone, 31 bit, load modules.**
* **Full REXX language supported (including INTERPRET)**
* **No transient library, and no licensing for object modules**
* **Load modules can be used for:**
  - **REXX functions (function packages)**
  - **TSO commands**
  - **Batch programs**
* **Parmlist type is determined dynamically**
* **Program source is included in the load module:**
  - **Source can be compressed (50-80+% compression)**
  - **Source can be encoded (renders it unreadable)**

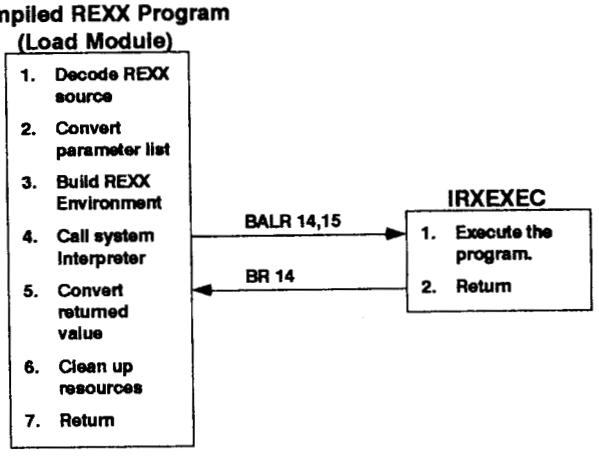
REXX Compiler

Compilation Process:



REXX Compiler

Relationship of compiled REXX programs to system interpreter:



REXX Compiler

Compiler Listing

```
1
-
ORXC 01.02.01          DSN=BI22EDH.USER.EXEC NAME=SAMPREXX
0
                REXXTOOLS/MVS REXX COMPILER  V01.02.01   18 Jan 1992 12:11:48
                COMPILING FROM BI22EDH.USER.EXEC ON VOLUME 780091 (3390)
                CURRENT USERID IS BI22EDH
                OPTIONS ARE: COMPRESS XREF VERSION(01.01.01) NAME(SAMPREXX)
```


REXX Compiler

Compiler Listing (continued)

```
1
-
ORXC 01.02.01          - A sample REXX program
0                      SOURCE LISTING
0               LINE
                   1   /* REXX - A sample REXX program */
                   2   ADDRESS TSO       /* establish host command environ.*/
                   3   /* Get the current date and
                   4      write it to the terminal */
                   5   today = date()
                   6   say 'today is 'today
                   7   /* now loop for awhile */
                   8   Do i = 1 to 30
```


REXX Compiler

Compiler Listing (continued)

```
 9     Say 'The time is now:     'time()
10     Select
11       When (i = 10) Then
12          Say "going..."
13       When (i = 20) Then
14          Say "going..."
15       When (i = 30) Then
16          Say "gone."
17       Otherwise
18          NOP
19     End    /* end of Select */
20   End   /* end of do i = 1 to 30 */
21   /* return to our caller */
22     exit
```


REXX Compiler

Compiler Listing (continued)

```
1
-
ORXC 01.02.01          - A sample REXX program
0                      COMPRESSED SOURCE LISTING
0              LINE
                  1    ADDRESS TSO
                  2    today=date()
                  3    say 'today is 'today
                  4    Do i=1 to 30
                  5    Say 'The time is now:     'time()
                  6    Select
                  7    When (i=10) Then
                  8    Say "going..."
```


REXX Compiler

Compiler Listing (continued)

```
 9   When (i=20) Then
10   Say "going..."
11   When (i=30) Then
12   Say "gone."
13   Otherwise
14   NOP
15   End
16   End
17   exit 0
```


REXX Compiler

Compiler Listing (continued)

```
1
-
ORXC 01.02.01          - A sample REXX program
0                      SYMBOL CROSS-REFERENCE LISTING
0              SYMBOL                  REFERENCES
               ADDRESS                 2
               DATE                    5
               DO                      8
               END                     19 20
               EXIT                    22
               I                       8 11 13 15
               NOP                     18
```

REXX Compiler

Compiler Listing (continued)

```
OTHERWISE           17
SAY                 6 9 12 14 16
SELECT              10
THEN                11 13 15
TIME                9
TO                  8
TODAY               5 6
TSO                 2
WHEN                11 13 15
```


REXX Compiler

Compiler Listing (continued)

```
1
-
ORXC 01.02.01          - A sample REXX program
0
               COMPILATION FINISHED
               ELAPSED TIME: 0.938795 (SEC)  CPU TIME: 0.60 (SEC)
               COMPRESSION: 523 BYTES COMPRESSED TO 204 BYTES. 60.99%
   COMPRESSION
               SOURCE RECORDS READ       22
               OBJECT RECORDS WRITTEN    12
               LIST   RECORDS WRITTEN    88
```

REXX Compiler

Compiler Benefits:

* Prevents unauthorized modifications to distributed REXX programs
* Saves DASD space
* Reduces run times
  - Load time reductions of 70+% (100% for function packages)
  - Execution time reductions 10-15%
  - Best profile for reducing time:
    + medium-to-large REXX program
    + executed frequently
    + short execution path

 41

REXX Compiler

PROCTSO EXEC

* Utility function for parsing arguments like CLIST PROC statement
* Before and after compilation comparison:

ITEM	BEFORE	AFTER	COMMENT
Bytes of code	21349	4578	(78.56% compression)
CPU secs/call	0.08	0.02	(0.06 sec. saved; %75 reduction)

* Executed approx. 1000/day (savings of 60 seconds)
* Assuming $1000.00/hour CPU time:
  - Saves $16.00/day
  - Saves $6080.00/year

 42