REXX Parsing

For today's slides and other webinars visit:

www.themisinc.com/webinars

PARSE Forms

PARSE ARG *template*	Parameters passed to program or subroutine
PARSE EXTERNAL *template*	Read from Terminal (TSO/E, VM only)
PARSE NUMERIC *template*	Current NUMERIC settings (TSO/E, VM only)
PARSE PULL *template*	Remove data from REXX STACK
PARSE SOURCE *template*	Information about the current program
PARSE VALUE *expression* WITH *template*	Information comes from expression
PARSE VAR *name template*	Parse one variable into other variables
PARSE VERSION *template*	Information about the REXX interpreter

General Rules for Parsing

- Parsing processes the data string from left to right
- If there is more data than defined variables, the last variable receives ALL the remaining data
- If there are more variables than data, the remaining variables are set as null
- A period (.) may be used as a “placeholder” to bypass setting a variable

PARSE VAR Keyword

PARSE [UPPER] VAR *origin template*

Use designated variable as input to template

PARSING Example

```
origin_data = ‘This is the original data’

PARSE VAR origin_data var1 var2 var3

var1 = This
var2 = is
var3 = the original data
```

PARSING Example #2

```
origin_data = 'This is the original data'

PARSE VAR origin_data var1 . . var3

var1 = This
var3 = original data
```

PARSING Example #3

origin_data = ‘This is the original data’

PARSE VAR origin_data var1 var2 var3 .

var1 = This

var2 = is

var3 = the

PARSING Example #4

origin_data = ‘This is the original data’

PARSE VAR origin_data var1 “the” var3 .

var1 = This is

var3 = original

NOTE: The placeholder (.) removes the last bit of data as space-delimited.

Parsing Example

Evaluate the following PARSE template:

- What will *dsn* and *member* contain?

```
dsname = "'SYS1.PROCLIB(JES2)'"
PARSE VAR dsname "'" dsn '(' member ')' .
```

```
dsn = SYS1.PROCLIB
member = JES2
```

PARSE EXTERNAL Keyword

PARSE [UPPER] EXTERNAL | LINEIN *template*

- Reads directly from terminal input
  - EXTERNAL - TSO or VM only
  - LINEIN - Windows, UNIX
  - UPPER - convert data to upper case

- The *template* controls how the input should be divided up by PARSE

PARSE PULL Keyword

PARSE [UPPER] PULL template

- PARSE PULL reads data from the Stack
- If the stack is empty then PULL will read from the terminal

PARSE VALUE Keyword

PARSE VALUE *expression* WITH *template*

```
PARSE VALUE TIME() WITH hh ":" mm ":" ss
```

PARSE ARG Keyword

PARSE [UPPER] ARG *template*

or

ARG *template* (PARSE UPPER is implied)

```
ARG n1,n2

PARSE UPPER ARG n1,n2
```

PARSE ARG Example

```
 /*   REXX  */
"CLRSCRN"
var1 = ‘parm1 parm1a parm1b'
var2 = 'parm2'
var3 = 'parm3'

CALL SUBRTN var1,var2,var3
EXIT 0

SUBRTN:
   PARSE ARG arg1 arg2 arg3,arg4,arg5
   SAY arg1
   SAY arg2
   SAY arg3
   SAY arg4
   SAY arg5
   RETURN
```

```
 parm1
 parm1a
 parm1b
 parm2
 parm3
```

PARSE ARG Example #2

```
 /*   REXX  */
"CLRSCRN"
var1 = 'This is the original data'
var2 = 'parm2'
var3 = 'parm3'

CALL SUBRTN var1,var2,var3
EXIT 0

SUBRTN:
   PARSE ARG arg1 'the' arg2 arg3,arg4,arg5
   SAY arg1
   SAY arg2
   SAY arg3
   SAY arg4
   SAY arg5
   RETURN
```

```
This is
original
data
parm2
parm3
```

More Parse Keywords

PARSE NUMERIC template (only applicable to TSO/E & VM)

```
PARSE NUMERIC data
SAY data                         ==>> 9 0 SCIENTIFIC
```

PARSE VERSION template

Information regarding language level

```
PARSE VERSION data
SAY data  ➔ REXX-ooRexx_4.2.0(MT)_64-bit 6.04 22 Feb 2014
```

Advanced PARSE

- PARSE can use absolute and relative positioning

```
alpha = "abcdefghijklmnopqrstuvwxyz"
```

Unsigned number moves cursor to absolute column

```
PARSE VAR alpha 8 c1 9 5 c2 6 12 c3 13 16 c4 17
SAY c1 c2 c3 c4            ==>> h e l p
```

Signed number moves cursor relative to current column

```
PARSE VAR alpha +13 instruction +3 .
SAY instruction            ==>> nop
```

Advanced PARSE

- Absolute & Relative positioning useful for extracting fields from I/O records
  - Variables past end of record (variable) set to null
  - Example:

```
alpha = “123456789"
PARSE VAR alpha 3 w1 +3 w2 3 w3


SAY “W1=“w1                         ==>> W1=‘345’
SAY “W2=“w2                         ==>> W2=‘6789’
SAY “W3=“w3                         ==>> W3=‘3456789’
```

Advanced PARSE

Parse can evaluate the same string multiple times

```
alpha = "abcdefghijklmnopqrstuvwxyz"

PARSE VAR alpha 1 s1 +3 1 s2 +5  .
SAY "S1="s1                      ==>> S1=abc
SAY "S2="s2                      ==>> S2=abcde

PARSE VAR alpha . 'ijk' -3 found +3 .
SAY 'found='found                ==>> found=fgh
PARSE VAR alpha . 'ijk' +0 found +3 .
SAY 'found='found                ==>> found=ijk
```

Advanced Parse

Parse can contain variables in the template to indicate literal strings, absolute or relative positioning.

```
alpha = "abcdefghijklmnopqrstuvwxyz"
needle = 'ijk'
len = LENGTH(needle)
PARSE VAR alpha (needle) found +3 .
SAY 'found='found                   ==>>
  found=ijk
```

Using Variables

- + move to the right
- - move to the left
- = absolute column

```
alpha = "abcdefghijklmnopqrstuvwxyz"

movec = 3
PARSE VAR alpha 1 s1 +(movec) 1 s2 +5  .
SAY "S1="s1                           ==>> S1=abc
SAY "S2="s2                           ==>> S2=abcde
```

Advanced PARSE Example

- Read the file ADDRESS.FILE
  - Extract:

NAME	1 - 16
ADDR	17 – 35
CITY	36 – 48
STATE	49 - 50

Advanced PARSE Code Example

```
 /*  REXX  */
var1 = 'Jimi Hendrix    1234 1st Street    Seattle      WA'
var2 = 'Edward Van Halen2435 Mullholland DrLos Angeles  CA'
var3 = 'Steve Vai       1179 Main Street   Denver       CO'
var4 = 'Frank Zappa     29735 Laural CanyonLos Angeles  CA'

movec = 19
col=36

PARSE VAR var1 1 name 17 17 addr +19 36 city 49
SAY name addr city
PARSE VAR var1 1 name 17 17 addr +(movec) 36 city 49
SAY name addr city
PARSE VAR var1 1 name 17 17 addr =(col) 36 city 49
SAY name addr city
PARSE VAR var2 1 name 17 17 addr 36 36 city 49
SAY name addr city
PARSE VAR var3 1 name 17 17 addr 36 36 city 49
SAY name addr city
PARSE VAR var4 1 name 17 17 addr 36 36 city 49
SAY name  addr city
```