#SHAREorg

SHARE
Technology · Connections · Results

Rexx Power Tools - The PARSE Command

Session 11751
August 7, 2012

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.rochester.rr.com/pinncons

SHARE in Anaheim 2012

#SHAREorg

SHARE
Technology · Connections · Results

Abstract

Do you need to chop up file records into separate data fields? Do you need to pass arguments to subroutines? Would you like to get better performance from your REXX routines? If you answered yes to any of these questions, then this session is for you! Come to this session to learn about the REXX Parse command. Parse is one of the most powerful and flexible REXX commands for handling strings. It's one-stop shopping for nearly anything you want to do with a string.

SHARE in Anaheim 2012

#SHAREorg

SHARE
Technology · Connections · Results

Agenda

- PARSE Command Flavors
- PARSE ARG
- PARSE EXTERNAL (z/OS)
- PARSE LINEIN (ooRexx)
- PARSE NUMERIC (z/OS)
- PARSE PULL
- PARSE SOURCE

#SHAREorg

SHARE
Technology · Connections · Results

Agenda

- PARSE VALUE
- PARSE VAR
- PARSE VERSION
- PARSE Templates
- Debugging PARSE
- Summary
- Finally…

PARSE ARG

- PARSE ARG used to parse arguments passed to Rexx exec or subroutine
- ARG is a short form for PARSE UPPER ARG – DON'T use ARG, use PARSE UPPER ARG if you really want uppercase arguments
- For the function call `tomstr(string,2,10)`, use the following procedure syntax:

```
tomstr : procedure
parse arg str, startpos, tomstrlen
```

PARSE ARG

- If invoking a main Rexx exec:

```
tomstr string 2 10
```

use the following syntax in exec tomstr:

```
/* rexx */
parse arg str startpos tomstrlen
```

PARSE ARG

- The PARSE command

```
parse arg str startpos tomstrlen
```

  is equivalent to

```
str       = arg(1)
startpos  = arg(2)
tomstrlen = arg(3)
```

- PARSE has better performance and is much easier to read

PARSE EXTERNAL (z/OS)

- PARSE EXTERNAL is used in z/OS to read from the terminal if running interactively, or from DD SYSTSIN if running in batch
- Returns a null string if SYSTSIN is empty, and a Rexx error message if SYSTSIN is not allocated
- Recommend using other features to pass input to Rexx exec
- If SYSTSIN is used to invoke Rexx exec, your results may be unpredictable

PARSE LINEIN (ooRexx)

- PARSE LINEIN in ooRexx will get the next line of the default input stream
- Shorter form of PARSE VALUE LINEIN() WITH…
- Only recommended if direct access to input stream is required, otherwise, use PARSE PULL

PARSE NUMERIC (z/OS)

- PARSE NUMERIC displays current numeric controls in effect in format DIGITS FUZZ FORM
- On TSO/E:

```
parse numeric numstr
```

  puts “9 0 SCIENTIFIC” into numstr (defaults)
- “9” indicates 9-digit precision, “0” is how many digits are ignored during numeric conversions (“fuzz”) and “SCIENTIFIC” is the exponential notation method

SHARE
Technology · Connections · Results

PARSE PULL

- PARSE PULL is used to pull data from the stack (z/OS) or external data queue (ooRexx)
- Can also pull data from a terminal if the Rexx is running interactively in z/OS, or default input stream if the external data queue is empty in ooRexx
- PULL is short for PARSE UPPER PULL
- DON'T use PULL, use PARSE UPPER PULL if you really want to uppercase input

SHARE
Technology · Connections · Results

PARSE PULL

- Common error is to use PULL for file I/O, which will corrupt lowercase text, packed, and binary data (BTDTGTTS)
- From TSO/E, use this instead:

```
"EXECIO 1 DISKR INFILE"
parse pull inrec
```

- INREC will hold the record read from INFILE

#SHAREorg

SHARE
Technology · Connections · Results

PARSE SOURCE

- PARSE SOURCE shows information about environment and how exec was invoked
- From ISPF, issue

```
tso exec 'ibmuser.tso.jcl(testsrc)'
```

PARSE SOURCE shows:

```
TSO COMMAND TESTSRC SYS00004
IBMUSER.TSO.JCL ? TSO ISPF ?
```

PARSE SOURCE

- From TSO READY prompt, issue:

```
exec 'ibmuser.tso.jcl(testsrc)'
```

- PARSE SOURCE shows:

```
TSO COMMAND TESTSRC SYS00007
IBMUSER.TSO.JCL ? TSO TSO/E ?
```

PARSE SOURCE

- From TSO READY prompt (or batch IKJEFT01), issue:

```
%testsrc
```

- PARSE SOURCE shows:

```
TSO COMMAND TESTSRC SYSPROC ? ? TSO
TSO/E ?
```

PARSE SOURCE

- From IRXJCL, use PARM=('TESTSRC')
- PARSE SOURCE shows:

```
TSO COMMAND TESTSRC SYSEXEC ?
TESTSRC MVS MVS ?
```

PARSE SOURCE

- From Windows/XP Command Window, issue:

```
C:\download>rexx version
```

- PARSE SOURCE shows:

```
WindowsNT COMMAND
C:\download\version.CMD
```

PARSE VALUE

- PARSE VALUE will parse the result of an expression without having to use an intermediate variable

```
parse value time() with hours ':' mins ':' secs
```

PARSE VAR

- PARSE VAR <variable> <template> parses the variable according to the patterns defined in the template

```
timevar = time()
parse var timevar hours ':' mins ':'
secs
```

PARSE VERSION

- PARSE VERSION displays information regarding the Rexx language level and modification date of the Rexx processor
- From TSO/E:

```
REXX370 3.48 01 May 1992
```

- From ooRexx:

```
REXX-ooRexx_4.1.0(MT) 6.03 5 Dec 2010
```

PARSE TEMPLATES

- PARSE templates are used by the PARSE VALUE and PARSE VAR commands to split a source string into multiple components
- The simplest template is a list of variable names, which will parse the source string into a series of blank delimited words
- The next examples will use the following string:

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'
```

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'

parse var string w1 w2 w3 w4 w5 .
w1 => 'This'
w2 => 'is'
w3 => 'a'
w4 => 'blank-delimited'
w5 => 'string'
```

PARSE TEMPLATES

- The period '.' in a template is a placeholder which acts as a dummy variable in the middle of a template, or collects the remainder of the source string when used at the end of a template
- If you do not include a period '.' at the end of your template, the last variable in your template will receive whatever is left of the source string

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'

parse var string w1 w2 w3 w4
w1 => 'This'
w2 => 'is'
w3 => 'a'
w4 => 'blank-delimited   string'
```

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'

parse var string w1 w2 w3 w4 .
w1 => 'This'
w2 => 'is'
w3 => 'a'
w4 => 'blank-delimited'
```

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'

parse var string w1 . w2 . w3 .
w1 => 'This'
w2 => 'a'
w3 => 'string'
```

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'

parse var string w1 . . w2 w3 .
w1 => 'This'
w2 => 'blank-delimited'
w3 => 'string'
```

PARSE TEMPLATES

- The next type of PARSE template is the string pattern template
- The string pattern is a literal or variable string pattern indicating where the source string should be split
- Literal: `parse var string w1 '-' w2 .`
- Variable:

```
dash = '-'
parse var string w1 (dash) w2 .
```

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'

parse var string w1 '-' w2 .
w1 => '  This is  a blank'
w2 => 'delimited'
```

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'
dash = '-'

parse var string w1 (dash) w2 .
w1 => '  This is  a blank'
w2 => 'delimited'
```

PARSE TEMPLATES

- The positional pattern template allows the source string to be parsed by character position
- An absolute positional pattern template is an absolute number or a number with an equal sign '=' preceding it

```
parse var string 5 w1 =10 w2 .
```

- Most commonly used template for processing fixed-format file records

#SHAREorg

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5...
string = 'Conley          Tom              Rexx Guru'

parse var string 1 lname 15 fname 30 title 39 .
lname => 'Conley          '
fname => 'Tom              '
title => 'Rexx Guru'
```

PARSE TEMPLATES

- A relative positional pattern template is an absolute number preceded by a plus sign '+' or a minus sign '-'

```
parse var string 5 w1 +5 w2 .
```

PARSE TEMPLATES

```
          1...5....1....5....2....5....3....5...
string = 'Conley         Tom              Rexx Guru'
parse var string 1 lname +14 fname +15 title +9 .
lname => 'Conley         '
fname => 'Tom             '
title => 'Rexx Guru'
```

PARSE TEMPLATES

- Mixing absolute and relative positional patterns can create some "interesting" situations

```
          1...5....1....5....2....5....3
string = 'REstructured eXtended eXecutor'
parse var string var1 3 junk 'X' var2 +1 junk 'X' var3 +1 junk
say var1||var2||var3
var1 => 'RE'
var2 => 'X'
var3 => 'X'
```

- Check out PARSE flow chart on pp. 172-174 of the TSO/E Rexx Reference to get the gory details of mixing absolute, relative, and literal templates

Debugging PARSE

- Best way to debug PARSE is to use "trace i"
- The following PARSE gives an unexpected result for w3

```
          1...5....1....5....2....5....3....5..
string = '  This is  a blank-delimited   string'
parse var string w1 . . w2 w3
w1 => 'This'
w2 => 'blank-delimited'
w3 => '  string'
```

Debugging PARSE

- trace i shows

```
*-* parse var string w1 . . w2 w3
>>>   "This"
>.>   "is"
>.>   "a"
>>>   "blank-delimited"
>>>   "  string"
```

SHARE
Technology · Connections · Results

Debugging PARSE

- After adding a period '.' to the end of the template, trace i shows

```
*-* parse var string w1 . . w2 w3 .
>>>    "This"
>.>    "is"
>.>    "a"
>>>    "blank-delimited"
>>>    "string"
>.>    ""
```

Summary

- The various flavors of the PARSE command were discussed
- Several template examples were shown and explained
- Debugging PARSE by using "trace i"

#SHAREorg

SHARE
Technology · Connections · Results

Finally….

- I'm always interested to hear about your experiences with PARSE, so if you have questions or come up with a neat solution to a problem, please feel free to drop me an Email pinncons@rochester.rr.com
- The TSO/E Rexx Reference can be found at http://publibz.boulder.ibm.com/epubs/pdf/ikj4a370.pdf
- Grab ooRexx at http://rexxla.org