Rexx for Shell Scripting

or

"We don't need no stinkin' bashes..."

Chip Davis
chip@aresti.com
2013 VM Workshop
UIPUI

Topics

- Comparison of Rexx and bash
- Brief overview of Rexx syntax
- Delightful features of Rexx
- A couple of demos
- Where to get Rexx for Linux

Rexx

- Designed to be easy
- ANSI standard
- Independent of OS
- Runs anywhere
- Sole designer (+ARB)
- Human-oriented
- Few limits
- Few special characters
- Single T/F value
- Easy to read, write, debug

bash

- Designed to be dense
- based on sh, ksh, ...
- Integral to OS
- Runs on Unix
- “Growed like Topsy”
- Machine-oriented
- Many different limits
- Many modal special characters
- Multiple T/F values
- Not so easy...

```
validate()
{
  varname=$1    varvalue=$2

  if [ ! -z $varvalue ] ; then
    if [ "${varvalue%${varvalue#?}}" = "/" ] ; then
      if [ ! -x $varvalue ] ; then
        echo "** $varname set to $varvalue, but I cannot \
             find executable."
        errors=$(( $errors + 1 ))
      fi
    else
      if in_path $varvalue $PATH ; then
        echo "** $varname set to $varvalue, but I cannot \
             find it in PATH."
        errors=$(( $errors + 1 ))
      fi
    fi
  fi
}
```

```
validate:
parse arg varname varvalue
if left(varvalue,1) = '/' then do
  'test -x' varvalue
  if rc then do
    say "**" varname "set to" varvalue", but I cannot" ,
        "find executable."
    errors = errors + 1
    end
  end
else
  if \ in_path(varvalue, path) then do
    say "**" varname "set to" varvalue", but I cannot" ,
        "find it in PATH."
    errors = errors + 1
  end
return 0
```

```
for directory in $PATH
do
  if [ ! -d $directory ] ; then
      echo "** PATH contains invalid directory $directory"
      errors=$(( $errors + 1 ))
  fi
done
```

```
path = value('PATH',,'SYSTEM')
do while path \= ""
  parse var path directory ':' path
  'test -d' directory
  if rc then do
    say "** PATH contains invalid directory" directory
    errors = errors + 1
  end
end
```

Rexx Instructions

1. Null — `/* comment */` `-- another comment`

   [*or just a blank line*]

2. Label — ***`symbol`***`:`

   `Get_Args:` `exit:`

3. Assignment — ***`symbol = expression`***

   `foo = length(bar'/'baz) + 2`

4. Keyword — **`do say if parse select trace numeric address call interpret`**

5. Command — ***`expression`***

   `'chmod' perms files`

Rexx Symbols

Allowed Characters:	[**a** - **z**] [**A** - **Z**] [**0** - **9**] **_** **.** **?** **!**
Format:	Must not start with **.** or [**0** - **9**]
Case-sensitive:	No
Length Limits:	ooRexx: unlimited Regina: 100,000 chars

Rexx Expressions

- Any sensible combination of:
  - Constants
  - Variables
  - Operators
  - Functions
- Blanks may be used between terms for readability
- Evaluated L-to-R, inner-to-outer parens
- Result is a character string

```
Say "February has" 28 + (yr // 4 = 0) "days"
```

Rexx Constants

- Stored as character strings
- Length limited to: 100,000 chars (Regina) [1]
  available memory (ooRexx)
- Numeric

```
2.54          -.088        6.02e-23
```

- String

```
'Say "Hi"'   ''  ""       "Don't look"
```

- Hex

```
'F8'x    'D0A'X           "c3c8 c9d7"x
```

- Binary

```
'111111000'b    '11'B   "1111 0011"b
```

Rexx Variables

symbol

- Length of value limited only by available memory
- Initial value is *symbol* in uppercase
- Simple: No '.'s in *symbol*

```
x15     !0_1?   Last_Matching_Value_Found
```

- Compound: Periods separate *stem* and *tails* in *symbol*

```
matrix.12.4     tax.item.st     A4.?
```

  - Values of *tail* variables are substituted and derived name is used to access the value

```
matrix.12.4     tax.book.NC     A4.Why not?
```

  - *stem.* = *expr* sets all *stem.tail.tail...* to value of *expr*

```
matrix. = 0     line. = ''      set.="none"
```

Rexx Operators

1. Prefix (monadic)	`+` `-` `\`
2. Power	`**`
3. Multiplicative	`*` `/` `%` `//`
4. Additive	`+` `-`
5. Concatenation	`||` [*abbutal*] [*blank*]
6. Comparison	`=` `<` `>` `<=` `>=`
	`==` `<<` `>>` `<<=` `>==`
7. AND	`&`
8. OR/XOR	`|` `&&`

An *expression* may contain <u>any</u> combination of operators

Rexx Functions

***symbol*([*expr* [, *expr*] ...])**

- Built-in
  - ANSI Functions
  - Platform Extensions
  - External Function Packages

```
SubStr(fn, LastPos('/', fn) + 1)
```

- Your Own
  - Subprocedure that returns a value
  - Returned value replaces invocation
  - Internal vs. External to executing program

```
MyFunc(ifn, ofn, recno, '*')
```

Trace

- Displays interpreted line before execution
- Many levels of detail: ACEFILNOR (may be set dynamically)
- Keyword instruction in code or interpreter invocation option
- Interactive tracing
  - Pauses for input after tracing instruction
  - Anything entered will be executed as if it were at that line in the program
  - Allows significant debugging at the point of failure
  - Last instruction traced may be re-executed
  - Trace setting saved across subprocedure boundaries
  - May skip over uninteresting instructions

Trace Identifiers

- Trace output line format: *line# trace_id trace_data*

`*-*`	Rexx instruction as found in program
`>>>`	String result of executing instruction
`>.>`	String ignored in Parse template
`>C>`	Derived name of compound variable
`>F>`	String returned from a Function
`>L>`	Literal string encountered
`>O>`	Result of a dyadic operation
`>P>`	Result of a monadic (prefix) operation
`>V>`	String retrieved from a variable
`+++`	Trace message

Address

- Controls to which environment a command is sent
- Redirects STDIO for one command or permanently

ADDRESS [*environment* [*command*] [**WITH** *redirects*]]

ADDRESS [*environment* [**WITH** *redirects*]]

environment: [**SYSTEM** | **COMMAND** | **REXX**]

command: [*external command*]

redirects: [**INPUT** *inp_redir*] [**OUTPUT** *out_redir*] [**ERROR** *err_redir*]

inp_redir: [**NORMAL**] | *where*

out_redir: [**NORMAL**] | [**REPLACE** | **APPEND**] *where*

err_redir: [**NORMAL**] | [**REPLACE** | **APPEND**] *where*

where: [**STREAM** | **STEM** | **LIFO** | **FIFO**] *name*

```
Address System 'grep ^chip:' With Input Stream 'etc_pw' ,
                                Output Stem users.
```

Demos

- validator
- bashit
- mychmod

Installing Rexx

- Regina Rexx

  sourceforge.net/projects/regina-rexx/files/regina-rexx/

- Open Object Rexx

  sourceforge.net/projects/oorexx/files/oorexx/

- Rexx Language Association

  www.rexxla.org

Finally...

- Questions?
- Comments?
- Brickbats?