Interesting Corners of REXX

Mike Cowlishaw
IBM
Interesting Corners of REXX
REXX Symposium

Mike Cowlishaw
IBM UK Laboratories
Hursley

IBM
May 1994
Outline

- Instructions
- Built-in Functions
- Miscellaneous
- Questions?
Multi-way CALL

var='FRED'
call jumper var, firstarg

jumper: procedure
signal value arg(1)

fred:       /* do whatever */

return      /* to the CALL */
DO FOREVER—can be clearer

```
do forever
    if something then leave
end /* forever loop */
```
DROP—extra state for a variable

drop var
do i=1 to howmany
  if whatever then var=somevalue
end i

if symbol('var')\='LIT' then say 'Found!'
NUMERIC FORM and FUZZ

With NUMERIC FORM ENGINEERING:

\[
\begin{align*}
\text{var} &= 1234 \\
\text{say } \text{var} \times 1e10 & \quad \text{var} \times 1e11 & \quad \text{var} \times 1e12 \\
\rightarrow & \\
12.34E+12 & \quad 123.4E+12 & \quad 1.234E+15
\end{align*}
\]

... and don't forget NUMERIC FUZZ for fuzzy comparisons.
Most implementations have variable column patterns:

```plaintext
namecol=pos('Name', header)
do i=1 to entries
    parse var entry.i =(namecol) name.i
end i
```

Use "." placeholder to strip blanks:

```plaintext
line='modemspeed = 9600 ' |
parse var line key . '=' value .
```
More PARSE

Use relative patterns to include strings in results:

parse var line pre 'START' 'SLIP' +0 post
if pre='' & post='SLIP' then do
  /* found 'start ... slip' */
  end

or...

parse var line pre 'WAIT' +0 key num post
if pre='' & key='WAIT' & post='' then do
  /* found 'wait [num]' */
  end

May 1994

Mike Cowlishaw
Parsing field-oriented data

/* Set up template matching structure */
/* (perhaps read from a file). */
template='socsecnum + 9',
    'name +40 -40',
    ' last +20',
    ' first +20',
    'balance + 4'

| record=charin(myfile, 80)
interpret 'parse var record' template
balance=c2d(balance, 4)

May 1994
Using PARSE for POS/SUBSTR

/* Change all "old" to "new" in string */
/* If "old" is null,"new" is prefixed */

Change: procedure
   parse arg string, old, new
   if old=='' then return new||string
   out=''
   do while pos(old, string)
    =0
    parse var string prefix (old) string
    out=out||prefix||new
   end
   return out||string

May 1994 Mike Cowlishaw
PROCEDURE EXPOSE lists

Lists can be very useful with *PROCEDURE EXPOSE*:

```plaintext
errors='sigl rc CleanupFlag'
shared='masterlist. CurName CurCount'
```

subfunction:
```plaintext
procedure expose (errors) (shared)
```

return
TRACE

Don’t forget:

trace Labels
... lets you check the flow in a program

and...

trace Intermediates
... lets you check expression evaluation in detail.

Note: The TRACE instruction is completely ignored during interactive tracing—but the TRACE() built-in function is not.

May 1994

Mike Cowlishaw
Function names in quotes

System-dependent function names can be useful:

\texttt{say 'e:\tools\testit.cmd'()}

or...

\texttt{say 'EXEC PROFILE'()}

... and they work with CALL, too.
Built-in Functions

**ABBREV** allows default match to null string:

```plaintext
say abbrev('PRINT', 'PRINT')
say abbrev('PRINT', 'PRI')
say abbrev('PRINT', '')
```

... all say 1

**CENTER** can be spelled properly, too:

```plaintext
say centre('goal kick', 25)
```
More Built-in Functions

COMPARE is often overlooked:

\[
\text{alpha} = \text{'abcdefgijklmnopqrstuvwxyz'} \\
\text{say compare(} \text{alpha, 'abcdefgijklmnoqps'} \text{)}
\]

... says 16

DATE lets you find the day-of-the week as a number:

\[
\text{say date('Base')} // 7
\]

... says 0 for Monday, 1 for Tuesday, etc.
INSERT and OVERLAY

Powerful, when you need them:
say insert('needle', 'haystack', 3)
... says “hayneedlestack”

and...
say overlay('12:30', 'It is hh:mm', 7)
... says “It is 12:30”
Removing character(s) from a string

Use *SPACE* (with a little help from *TRANSLATE*):

```
string='Hoppy floppy'
string=translate(string, 'p', ' p')
string=space(string, 0)
string=translate(string, 'p', ' p')
```

... sets *STRING* to "Hoy floy"

For multiple characters, use (for example):

```
string=translate(string, 'a', ' aeiou')
string=space(string, 0)
string=translate(string, 'a', ' a')
```
Testing for parity

Use SPACE — with a little help from X2B, TRANSLATE, and LENGTH:

bits=x2b('C7')  /* 11000111 */
ones=translate(bits, ' ', '0')
ones=length(space(ones, 0))
parity=ones//2

... sets PARITY to "1"
SUBSTR, LEFT, and RIGHT

SUBSTR or LEFT can take a pad character:

say substr('Fred', 1, 8, '?')
say left('Fred', 8, '?')

... both say “Fred?????”

RIGHT can pad on the left, or return rightmost characters:

say right(12, 6, 0)
say right('e:\extra.cmd', 3)

... says “000012” and “cmd”
TRANSLATE

As well as character substitution, *TRANSLATE* can be used to reformat (lay out) strings:

\[
\text{in} = 'abcdefgh' \\
\text{pattern} = 'gh.ef.abcd' \\
\text{say translate(pattern, '19940827', in)}
\]

... says "27.08.1994"
VERIFY can look for "the odd one in", as well as "the odd one out":

\[
\text{say verify('123.456', '0123456789')}\]

... says "4"

but...

\[
\text{say verify("It's 1994", '13579', 'Match')}\]

... says "6"
And finally...

/* Shuffle the numbers in range 1->max */
shuffle: procedure
    signal off novalue
    max=arg(1)
    out=''
    do i=1 to max
        sub=random(i,max)
        out=out substr(ar.sub,4)
        if sub=i then iterate
        ar.sub=ar.i
    end
    return out