

# Rexx Parse Templates

---

exposed



---

Chip Davis



[chip@aresti.com](mailto:chip@aresti.com)

Aresti Systems, LLC



[www.aresti.com](http://www.aresti.com)

# Instruction Format - Source

---

PARSE [UPPER] *source* *template*

- *source*
  - ARG invocation argument(s)
  - LINEIN next line from STDIN
  - PULL top line of stack
  - SOURCE info about running program
  - VALUE expression
  - VAR contents of a variable
  - VERSION info about interpreter

# Instruction Format - Template

PARSE [UPPER] *source* *template*

## ■ *template*

- data variables `rec2 rlen food bard`
- placeholder periods `.`
- explicit patterns
  - ▶ absolute position `12 =42`
  - ▶ relative position `+8 -4`
  - ▶ literal `'/' "total=" '09'x`
- variable reference patterns `(delim)(patrn3)`

# Parsing Hierarchy

{ Template Selection }

PARSE PULL

*template*

PARSE ARG *template1, template2, template3*

{ Pattern Matching }

*variables*

*pattern*

*variables*

*pattern*

*variables*

{ Word Parsing }

*var1 var2*

*var3 var4 var5 var6*

*var7*

# Instruction Format - Template

---

PARSE [UPPER] *source* *template*

- *template*

- data variables

*rec2 rlen food bard*

- placeholder periods

*.*

# Word Parsing

---

- Each variable in template is assigned a word of the data string
  - All leading blanks are removed
  - One trailing blank is removed
  - **Exception** - last variable in template:
    - ▶ No blanks are removed
    - ▶ Rest of data string assigned to last variable
- If no data, null string is assigned to variable
- Placeholder periods ignore data word
- Every variable will get a new value

# Word Parsing (1)

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

\*-\* Parse Var str v1 v2 v3 . v4 v5 . v6 v7 v8

v1 > \_\_\_\_\_

v2 > \_\_\_\_\_

v3 > \_\_\_\_\_

. > \_\_\_\_\_

v4 > \_\_\_\_\_

v5 > \_\_\_\_\_

. > \_\_\_\_\_

v6 > \_\_\_\_\_

v7 > \_\_\_\_\_

v8 > \_\_\_\_\_

# Word Parsing (1)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

\*-\* Parse Var str v1 v2 v3 . v4 v5 . v6 v7 v8

v1 > "We"

v2 > "have"

v3 > "met"

. > "the"

v4 > "enemy, "

v5 > "and"

. > "he"

v6 > "is"

v7 > "us."

v8 > ""

# Word Parsing (2)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

\*-\* Parse Var str v1 v2 v3 . v4 v5 v6

v1 > \_\_\_\_\_

v2 > \_\_\_\_\_

v3 > \_\_\_\_\_

. > \_\_\_\_\_

v4 > \_\_\_\_\_

v5 > \_\_\_\_\_

v6 > \_\_\_\_\_

# Word Parsing (2)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

\*-\* Parse Var str v1 v2 v3 . v4 v5 v6

v1 > "We"

v2 > "have"

v3 > "met"

. > "the"

v4 > "enemy, "

v5 > "and"

v6 > " he is us. "

# Word Parsing (3)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

\*-\* Parse Var str v1

v1 > \_\_\_\_\_

# Word Parsing (3)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

\*-\* Parse Var str v1

v1 > " We have met the enemy, and he is us. "

# Parsing Hierarchy

{ Template Selection }

PARSE PULL

*template*

PARSE ARG *template1, template2, template3*

{ Pattern Matching }

*variables*

*pattern*

*variables*

*pattern*

*variables*

{ Word Parsing }

*var1 var2*

*var3 var4 var5 var6*

*var7*

# Instruction Format - Template

---

PARSE [UPPER] source *template*

## ■ *template*

- data variables `rec2 rlen food bard`
- placeholder periods `.`
- explicit patterns
  - ▶ absolute position `12 =42`

# Position Pattern Parsing

---

- 1. Find Start Point in the data string
- 2. Find Match Point in the data string
- 3. WordParse the data substring into the variables between the template patterns

# Find Start Point

---

- If beginning of template
  - SP = first character of data
- If a previous MP position pattern
  - SP = that character position in the data

# Find Match Point

---

- If no more patterns in the template
  - $MP = \text{end of data} + 1$
- If another pattern in the template
  - $MP = \text{char position of } \underline{\text{start}} \text{ of next pattern}$

# WordParse the Data

---

- Extract the data string from the StartPoint to, but not including, the MatchPoint
- WordParse this string into the variable(s) between the template patterns

# Absolute Position Patterns (1)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

S M

\*-\* Parse Var str 1 v1 =10 v2 v3 v4 17 v5 =25 v6

v1 > \_\_\_\_\_

v2 > \_\_\_\_\_

v3 > \_\_\_\_\_

v4 > \_\_\_\_\_

v5 > \_\_\_\_\_

v6 > \_\_\_\_\_

# Absolute Position Patterns (1)

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
      S           M
*-* Parse Var str 1 v1 =10 v2 v3 v4 17 v5 =25 v6
v1 > " We have "
v2 > _____
v3 > _____
v4 > _____
v5 > _____
v6 > _____
```

# Absolute Position Patterns (1)

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
          S          M
*-* Parse Var str 1 v1 =10 v2 v3 v4 17 v5 =25 v6
v1 > " We have "
v2 > "met"
v3 > "th"
v4 > ""
v5 > _____
v6 > _____
```

# Absolute Position Patterns (1)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

S M

```
*-* Parse Var str 1 v1 =10 v2 v3 v4 17 v5 =25 v6
```

```
v1 > " We have "
```

```
v2 > "met"
```

```
v3 > "th"
```

```
v4 > ""
```

```
v5 > "e enemy, "
```

```
v6 > _____
```

# Absolute Position Patterns (1)

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
                                     S           M
*-* Parse Var str 1 v1 =10 v2 v3 v4 17 v5 =25 v6
v1 > " We have "
v2 > "met"
v3 > "th"
v4 > ""
v5 > "e enemy,"
v6 > " and he is us. "
```

# Instruction Format - Template

---

PARSE [UPPER] *source* *template*

## ■ *template*

- data variables
- placeholder periods
- explicit patterns
  - ▶ absolute position
  - ▶ relative position

rec2 rlen food bard

.

12 =42

+8 -4

# Find Start Point

---

- If beginning of template
  - SP = first character of data
- If absolute position pattern (8 =42)
  - SP = that character position in the data
- If relative position pattern (+57 -2)
  - SP = (previous MP + pattern) char position

# Relative Position Patterns (1)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

S M

\*-\* Parse Var str 1 v1 +9 v2 v3 v4 +7 v5 +8 v6

v1 > \_\_\_\_\_

v2 > \_\_\_\_\_

v3 > \_\_\_\_\_

v4 > \_\_\_\_\_

v5 > \_\_\_\_\_

v6 > \_\_\_\_\_

# Relative Position Patterns (1)<sub>></sub>

---

```
      .....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
      S           M
*-* Parse Var str 1 v1 +9 v2 v3 v4 +7 v5 +8 v6
v1 > " We have "
v2 > _____
v3 > _____
v4 > _____
v5 > _____
v6 > _____
```

# Relative Position Patterns (1)

---

```
      .....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
           S           M
*-* Parse Var str 1 v1 +9 v2 v3 v4 +7 v5 +8 v6
v1 > " We have "
v2 > "met"
v3 > "th"
v4 > ""
v5 > _____
v6 > _____
```

# Relative Position Patterns (1)<sub>></sub>

---

```
      .....|.....1.....|.....2.....|.....3.....|.....4
str> " We have  met the enemy, and  he is us. "
                S          M
*-* Parse Var str 1 v1 +9 v2 v3 v4 +7 v5 +8 v6
v1 > " We have "
v2 > "met"
v3 > "th"
v4 > ""
v5 > "e enemy, "
v6 > _____
```

# Relative Position Patterns (1)

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
                                     S           M
*-* Parse Var str 1 v1 +9 v2 v3 v4 +7 v5 +8 v6
v1 > " We have "
v2 > "met"
v3 > "th"
v4 > ""
v5 > "e enemy,"
v6 > " and he is us. "
```

# Find Match Point

---

- If no more patterns in the template
  - $MP = \text{end of data} + 1$
- If another pattern in the template
  - $MP = \text{char position of } \underline{\text{start}} \text{ of next pattern}$
- If  $MP \leq SP$  (not moving forward in the data)
  - ▶  $MP = \text{end of data} + 1$

# Absolute Position Patterns (2)

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
                                     S           M
*-* Parse Var str 1 v1 =10 v2 v3 v4 17 v5 =25 v6 1 v7 v8
v1 > " We have "
v2 > "met"
v3 > "th"
v4 > ""
v5 > "e enemy,"
v6 > " and he is us. "
v7 > _____
v8 > _____
```

# Absolute Position Patterns (2)

```
      .....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
      S                                     M
*-* Parse Var str 1 v1 =10 v2 v3 v4 17 v5 =25 v6 1 v7 v8
v1 > " We have "
v2 > "met"
v3 > "th"
v4 > ""
v5 > "e enemy,"
v6 > " and he is us. "
v7 > "We"
v8 > "have met the enemy, and he is us. "
```

# Absolute Position Patterns (3)

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
      S                               M
*-* Parse Var str 1 v1 =25 v2 =15 . v3 =19 v4 =5 v5
v1 > " We have met the enemy,"
v2 > _____
v3 > _____
v4 > _____
v5 > _____
```

# Absolute Position Patterns (3)

---

```

      .....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
                                     S           M
*-* Parse Var str 1 v1 =25 v2 =15 . v3 =19 v4 =5 v5
v1 > " We have met the enemy,"
v2 > " and he is us. "
v3 > _____
v4 > _____
v5 > _____
```

# Absolute Position Patterns (3)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

S M

\*-\* Parse Var str 1 v1 =25 v2 =15 . v3 =19 v4 =5 v5

v1 > " We have met the enemy, "

v2 > " and he is us. "

v3 > ""

v4 >

---

v5 >

---

# Absolute Position Patterns (3)

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
                                     S           M
*-* Parse Var str 1 v1 =25 v2 =15 . v3 =19 v4 =5 v5
v1 > " We have met the enemy, "
v2 > " and he is us. "
v3 > ""
v4 > "enemy, and he is us. "
v5 > _____
```

# Absolute Position Patterns (3)

---

```

      .....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
      S                                     M
*-* Parse Var str 1 v1 =25 v2 =15 . v3 =19 v4 =5 v5
v1 > " We have met the enemy, "
v2 > " and he is us. "
v3 > ""
v4 > "enemy, and he is us. "
v5 > "have met the enemy, and he is us. "
```

# Relative Position Patterns (3)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

\*-\* Parse Var str 1 v1 +24 v2 -10 . v3 +4 v4 -14 v5

v1 > \_\_\_\_\_

v2 > \_\_\_\_\_

v3 > \_\_\_\_\_

v4 > \_\_\_\_\_

v5 > \_\_\_\_\_

# Relative Position Patterns (3)

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
```

```
str> " We have met the enemy, and he is us. "
```

```
*-* Parse Var str 1 v1 +24 v2 -10 . v3 +4 v4 -14 v5
```

```
v1 > " We have met the enemy, "
```

```
v2 > " and he is us. "
```

```
v3 > ""
```

```
v4 > "enemy, and he is us. "
```

```
v5 > "have met the enemy, and he is us. "
```

# Fields - Absolute

.....|.....1.....|.....2.....|.....3.....|.....4

```
str> " We have met the enemy, and he is us. "
```

```
*-* Parse Var str =2 v1 =4 . ,  
                  =5 v2 =9 . ,  
                  =11 v3 =14 . ,  
                  =15 v4 =18 . ,  
                  =19 v5 =24 . ,  
                  =26 v6 =29 . ,  
                  =31 v7
```

```
v1 > "We"
```

```
v2 > "have"
```

```
v3 > "met"
```

```
v4 > "the"
```

```
v5 > "enemy"
```

```
v6 > "and"
```

```
v7 > "he is us. "
```

# Fields - Absolute

.....|.....1.....|.....2.....|.....3.....|.....4

```
str> " We have met the enemy, and he is us. "
```

```
*-* Parse Var str =2 v1 +2 ,  
                  =5 v2 +4 ,  
                  =11 v3 +3 ,  
                  =15 v4 +3 ,  
                  =19 v5 +5 ,  
                  =26 v6 +3 ,  
                  =31 v7
```

```
v1 > "We"
```

```
v2 > "have"
```

```
v3 > "met"
```

```
v4 > "the"
```

```
v5 > "enemy"
```

```
v6 > "and"
```

```
v7 > "he is us. "
```

# String Pattern Parsing

---

- 1. Find Start Point in the data string
- 2. Find Match Point in the data string
- 3. WordParse the data substring into the variables between the template patterns

# Find Start Point

---

- If beginning of template
  - SP = first character of data
- If literal pattern (" , " 'Type: ' )
  - SP = first character following pattern

# Find Match Point

---

- If no more patterns in the template
  - $MP = \text{end of data} + 1$
- If another pattern in the template
  - If part of remaining data matches pattern
    - ▶  $MP = \text{char position of } \underline{\text{start}} \text{ of next pattern}$
  - If no remaining data matches the pattern
    - ▶  $MP = \text{end of data} + 1$

# WordParse the Data

---

- Extract the data string from the StartPoint to, but not including, the MatchPoint
- WordParse this string into the variable(s) between the template patterns

# Literal Patterns (1)<sub>></sub>

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

S

M

\*-\* Parse Var str v1 v2 v3 ',' v4 v5

v1 > \_\_\_\_\_

v2 > \_\_\_\_\_

v3 > \_\_\_\_\_

v4 > \_\_\_\_\_

v5 > \_\_\_\_\_

# Literal Patterns (1)>

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
      S                               M
*-* Parse Var str v1 v2 v3 ',' v4 v5
v1 > "We"
v2 >
v3 >
v4 >
v5 >
```

# Literal Patterns (1)>

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
      S                               M
*-* Parse Var str v1 v2 v3 ',' v4 v5
v1 > "We"
v2 > "have"
v3 >
v4 >
v5 >
```

# Literal Patterns (1)

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
      S                               M
*-* Parse Var str v1 v2 v3 ',' v4 v5
v1 > "We"
v2 > "have"
v3 > " met the enemy"
v4 >
v5 >
```

# Literal Patterns (1)

---

.....|.....1.....|.....2.....|.....3.....|.....4

```
str> " We have met the enemy, and he is us. "
```

S

M

```
*-* Parse Var str v1 v2 v3 ',' v4 v5
```

```
v1 > "We"
```

```
v2 > "have"
```

```
v3 > " met the enemy"
```

```
v4 >
```

```
v5 >
```

# Literal Patterns (1)

---

.....|.....1.....|.....2.....|.....3.....|.....4

```
str> " We have met the enemy, and he is us. "
```

S

M

```
*-* Parse Var str v1 v2 v3 ',' v4 v5
```

```
v1 > "We"
```

```
v2 > "have"
```

```
v3 > " met the enemy"
```

```
v4 > "and"
```

```
v5 >
```

# Literal Patterns (1)

---

.....|.....1.....|.....2.....|.....3.....|.....4

```
str> " We have met the enemy, and he is us. "
```

S

M

```
*-* Parse Var str v1 v2 v3 ',' v4 v5
```

```
v1 > "We"
```

```
v2 > "have"
```

```
v3 > " met the enemy"
```

```
v4 > "and"
```

```
v5 > " he is us. "
```

# Variable Patterns (1)

---

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
```

```
*-* dlim = ','
*-* Parse Var str v1 v2 v3 (dlim)v4 v5
v1 > "We"
v2 > "have"
v3 > " met the enemy"
v4 > "and"
v5 > " he is us. "
```

# Variable Patterns (2)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

```
*-* dlim = 24
```

```
*-* Parse Var str v1 v2 v3 (dlim)v4 v5
```

```
v1 > "We"
```

```
v2 > "have"
```

```
v3 > " met the enemy, and he is us. "      !!!
```

```
v4 > ""
```

```
v5 > ""
```

# Variable Patterns (3)

---

.....|.....1.....|.....2.....|.....3.....|.....4

```
str> " We have met the enemy, and he is us. "
```

```
*-* dlim = 24
```

```
*-* Parse Var str v1 v2 v3 =(dlim) v4 v5
```

```
v1 > "We"
```

```
v2 > "have"
```

```
v3 > " met the enemy"
```

```
v4 > ", "
```

```
v5 > "and he is us. "
```

# Variable Patterns (4)

```
line.1 = 'Fannie Mae;1;19;2:55;'
line.2 = 'Midnight Special/2/23/2:54/'
line.3 = 'Wang Dang Doodle,2,10,2:59,'
line.4 = 'St. Louis Blues-1-10-3:02-'
Do i = 1 To 4
  lastchar = Length(line.i)
  Parse Var line.i =(lastchar) dlim ,
              =1 title (dlim) ,
              cdnum (dlim) ,
              trk (dlim) ,
              time (dlim) .
  Say Left(title,20) Right(cdnum,2) ,
      Right(trk,3) Left(time,8)
End i
```

---

Fannie Mae	1	19	2:55
Midnight Special	2	23	2:54
Wang Dang Doodle	2	10	2:59
St. Louis Blues	1	10	3:02

# Find Start Point

---

- If beginning of template
  - SP = first character of data
- If literal pattern ( " , " 'Type: ' )
  - SP = first character following pattern ★
- ★ unless the MP is a relative position pattern, in which case  
SP = first character of pattern

# Literal+Relative Patterns (1)

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

S

M

\*-\* Parse Var str v1 v2 v3 ',' v4 +1 v5 v6

v1 > \_\_\_\_\_

v2 > \_\_\_\_\_

v3 > \_\_\_\_\_

v4 > \_\_\_\_\_

v5 > \_\_\_\_\_

v6 > \_\_\_\_\_

# Literal+Relative Patterns (1)

---

.....|.....1.....|.....2.....|.....3.....|.....4

str> " We have met the enemy, and he is us. "

SM

\*-\* Parse Var str v1 v2 v3 ',' v4 +1 v5 v6

v1 > "We"

v2 > "have"

v3 > " met the enemy"

v4 > \_\_\_\_\_

v5 > \_\_\_\_\_

v6 > \_\_\_\_\_

# Literal+Relative Patterns (1)

```
.....|.....1.....|.....2.....|.....3.....|.....4
str> " We have met the enemy, and he is us. "
                                     S           M
*-* Parse Var str v1 v2 v3 ',' v4 +1 v5 v6
v1 > "We"
v2 > "have"
v3 > " met the enemy"
v4 > ","
v5 > "and"
v6 > " he is us. "
```

# Parsing Hierarchy

{ Template Selection }

PARSE PULL

*template*

PARSE ARG *template1, template2, template3*

{ Pattern Matching }

*variables*

*pattern*

*variables*

*pattern*

*variables*

{ Word Parsing }

*var1 var2*

*var3 var4 var5 var6*

*var7*

# Argument Strings

---

- Only one data string parsed at a time
- **Exception -PARSE ARG :**
  - Each argument is a separate string
  - Argument data strings separated by commas
  - PARSE ARG templates separated by commas
  - Omitted arg same as null string
- OpSys passes only one argument string

# Parse Arg (1)

---

```
*-* call r1 'A Practical', 'Approach', 'to Programming'  
*-* ...  
*-* r1: parse arg arg1, arg2, arg3  
   arg1 > "A Practical"  
   arg2 > "Approach"  
   arg3 > "to Programming"  
  
*-* call r1 'A Practical Approach',, 'to Programming'  
*-* ...  
*-* r1: parse arg v1 v2 . v3, v4 v5 . , . v6  
   v1    > "A"  
   v2    > "Practical"  
   v3    > ""  
   v4    > ""  
   v5    > ""  
   v6    > "Programming"
```

# Parse Arg (2)

---

```
*-* call r2 'A Practical Approach',, 'to Programming'  
*-* ...  
*-* r2: parse arg v1 'Pr' v2 v3 +99, . , =7 v4 +4 . v5  
v1 > "A "  
v2 > "Practical"  
v3 > "Approach"  
v4 > "gram"  
v5 > ""  
  
*-* call r3 'A Practical Approach',, 'to Programming'  
*-* ...  
*-* r3: parse arg ,,v4 v5 v6  
v4 > "to"  
v5 > "Programming"  
v6 > ""
```

# So In Summary ...

---



# Instruction Format - Template

PARSE [UPPER] *source* *template*

## ■ *template*

- data variables

rec2 rlen food bard

- placeholder periods

.

- explicit patterns

- ▶ absolute position

12 =42

- ▶ relative position

+8 -4

- ▶ literal

'/' "total=" '09'x

- variable reference patterns

(delim)(patrn3)

# Parsing Hierarchy

{ Template Selection }

PARSE PULL

*template*

PARSE ARG *template1, template2, template3*

{ Pattern Matching }

*variables*

*pattern*

*variables*

*pattern*

*variables*

{ Word Parsing }

*var1 var2*

*var3 var4 var5 var6*

*var7*

# Find Start Point

- If beginning of template
  - SP = first character of data
- If previous absolute position pattern (8=42)
  - SP = pattern as a character position
- If previous relative position pattern (+57-2)
  - SP = previous MP + pattern as char position
- If previous literal pattern (" , " 'Type: ' )
  - SP = first character following pattern



# Find Match Point

---

- If no more patterns in the template
  - $MP = \text{end of data} + 1$
- If another pattern in the template
  - If part of remaining data matches pattern
    - ▶  $MP = \text{char position of } \underline{\text{start}} \text{ of next pattern}$
  - If no remaining data matches the template
    - ▶  $MP = \text{end of data} + 1$
- If  $MP \leq SP$  (not moving forward in the data)
  - ▶  $MP = \text{end of data} + 1$

# Word Parsing

---

- Each variable in template is assigned a word of the data string
  - All leading blanks are removed
  - One trailing blank is removed
  - **Exception** - last variable in template:
    - ▶ No blanks are removed
    - ▶ Rest of data string assigned to last variable
- If no data, null string is assigned to variable
- Placeholder periods ignore data word
- Every variable will get a new value

# Conclusion

---

- Rexx Parse Templates are:
  - Very Powerful
  - Easy to Use (now that you understand them!)
  - Flexible
  - Efficient
  - Consistent
  - Human-oriented