"rgf_util2.rex"

2009 International Rexx Symposium

Chilworth, England (May 2009)

Rony G. Flatscher (Rony.Flatscher@wu.ac.at)
Wirtschaftsuniversität Wien, Austria (http://www.wu.ac.at)

Agenda

- Overview
- String-related BIF-Extensions
- Routine "dump2()"
- Parsing of words
- Routines "sort2()" and "stableSort2()"
- New comparators
- Roundup

String-related BIF-Extensions Caseless & Negative Numbers, 1

- Default to caseless comparison modes
- Allow negative numbers
  - Define negative positions
  - Define negative lengths
- Implemented as public routines
  - Named after their BIFs
  - Supplement digit "2" to BIF-name
- Works on 3.x and 4.x versions of ooRexx
  - On 4.x .context~package is used to add them

String-related BIF-Extensions Caseless & Negative Numbers, 2

- Semantics, if negative numbers
  - Negative start position
    - Start from the other side and position in the opposite direction
  - Negative length number
    - Position in the opposite direction

String-related BIF-Extensions Caseless & Negative Numbers, 3

- abbrev2(info, string [,length] [,C|I])
- changeStr2(needle, haystack, newNeedle [,count] [,C|I])
- delStr2(string, start [,length])
- delWord2(string, start [,count])
- lastPos2(needle, haystack [,start] [,C|I])
- left2(string, length [,pad])
- lower2(string [,start] [,length])
- overlay2(new, target [,start] [,length])
- pos2(needle, haystack, [,start] [,C|I])

String-related BIF-Extensions Caseless & Negative Numbers, 4

- right2(string, length [,pad])
- subChar2(string, pos)
- subStr2(string, pos [,length])
- subWord2(string, pos [,count])
- upper2(string [,start] [,length])
- word2(string, pos)
- wordIndex2(string, pos)
- wordLength2(string, pos)
- wordPos2(phrase, string [,start] [,C|I])

String-related BIF-Extensions Caseless & Negative Numbers, 5

- Examples

```
ABBREV2("Print", "Pri") ....................... -> [1]
ABBREV2("PRINT", "Pri", 1) .................... -> [1]

ABBREV2("int", "Pri", -1) ..................... -> [1]

ABBREV2("Print", "PRI", "I") .................. -> [1]
ABBREV2("PRINT", "Pri", "C") .................. -> [0]
------------------------------------------------------------------
CHANGESTR2("I","I0II00","X")    ............... -> [X0XX00]
CHANGESTR2("I","I0II00","X", 1) ............... -> [X0II00]

CHANGESTR2("I","I0II00","X", -1) .............. -> [I0IX00]
CHANGESTR2("I","I0II00","X", -2) .............. -> [I0XX00]

CHANGESTR2("AB","AB0ABBAAB0AB","--",  2) ..... -> [--0--BAAB0AB]
CHANGESTR2("AB","AB0ABBAAB0AB","--", -2) ..... -> [AB0ABBA--0--]

CHANGESTR2("i","I0II00","X",  ,"C") ........... -> [I0II00]
CHANGESTR2("i","I0II00","X", 1,"I") ........... -> [X0II00]
```

String-related BIF-Extensions Caseless & Negative Numbers, 6

- Examples

```
COMPARE2("abc","abc")      ..................... -> [0]
COMPARE2("abc","ABC")      ..................... -> [0]
COMPARE2("abc","ak")       ..................... -> [2]

COMPARE2("Ab-- ","aB","-","I") ................. -> [5]
COMPARE2("Ab-- ","aB","-","C") ................. -> [1]
COMPARE2("Ab-- ","Ab","-","C") ................. -> [5]
-------------------------------------------------------------
COUNTSTR2("1", "101101") ....................... -> [4]
COUNTSTR2("KK","J0KKK0") ....................... -> [1]
COUNTSTR2("KK","J0KKKK0") ...................... -> [2]
COUNTSTR2("KK","J0kkk0") ....................... -> [1]

COUNTSTR2("KK","J0KKK0", "I") .................. -> [1]
COUNTSTR2("kk","J0KKKK0","I") .................. -> [2]
COUNTSTR2("KK","J0kkk0", "I") .................. -> [1]
COUNTSTR2("KK","J0kkk0", "C") .................. -> [0]
```

String-related BIF-Extensions Caseless & Negative Numbers, 7

- Examples

```
DELSTR2("abcd",   3    ) ..................... -> [ab]
DELSTR2("abcde",  3,  2) ..................... -> [abe]
DELSTR2("abcde",  6    ) ..................... -> [abcde]

DELSTR2("abcd",  -3    ) ..................... -> [a]
DELSTR2("abcde", -3, -2) ..................... -> [ade]

DELSTR2("abc",    1    ) ..................... -> []
DELSTR2("abc",   -1    ) ..................... -> [ab]

DELSTR2("abc",    3    ) ..................... -> [ab]
DELSTR2("abc",   -3    ) ..................... -> []
```

String-related BIF-Extensions Caseless & Negative Numbers, 8

- Examples

```
DELWORD2("   eins zwei drei ", 2)          ..... -> [   eins ]
DELWORD2("   eins zwei drei ", 2, 1)       ..... -> [   eins drei ]

DELWORD2("   eins zwei drei ",-1)          ..... -> [   eins zwei ]
DELWORD2("   eins zwei drei ",-2)          ..... -> [   eins ]

DELWORD2("   eins zwei drei ",-2, 1)       ..... -> [   eins drei ]
DELWORD2("   eins zwei drei ",-2,-1)       ..... -> [   eins drei ]

DELWORD2("   eins zwei drei ",-2,-2)       ..... -> [   drei ]
DELWORD2("   eins zwei drei ", 2,-2)       ..... -> [   drei ]
DELWORD2("   eins zwei drei ",-2, 2)       ..... -> [   eins ]
```

String-related BIF-Extensions Caseless & Negative Numbers, 9

- Examples

```
LASTPOS2(" ",  "abc def ghi"   ) ..............  -> [8]
LASTPOS2(" ",  "abc def ghi", 8) ..............  -> [8]
LASTPOS2(" ",  "abc def ghi", -1) .............  -> [8]
LASTPOS2(" ",  "abc def ghi", -8) .............  -> [4]
LASTPOS2("xY", "efGXYZXYXY",  9)      .........  -> [7]
LASTPOS2("xY", "efGXYZXYXY",  9, "I") .........  -> [7]
LASTPOS2("xY", "efGXYZXYXY",  9, "C") .........  -> [0]
-------------------------------------------------------------
LEFT2("abc d"  , 8     ) ......................  -> [abc d   ]
LEFT2("abc d"  , 8, ".") ......................  -> [abc d...]
LEFT2("abc d"  , -8    ) ......................  -> [   abc d]
LEFT2("abc d"  , -8, ".") .....................  -> [...abc d]
-------------------------------------------------------------
lower2("ABCDEF"         ,  4)     .............  -> [ABCdef]
lower2("ABCDEF"         ,  3,  2) .............  -> [ABcdEF]
lower2("ABCDEF"         , -4)     .............  -> [ABcdef]
lower2("ABCDEF"         , -3, -2) .............  -> [ABcdEF]
```

String-related BIF-Extensions Caseless & Negative Numbers, 10

- Examples

```
overlay2("12", "abc", 2        ) ..............  -> [a12]
overlay2("12", "abc", 2, 1     ) ..............  -> [a1c]
overlay2("12", "abc", 2, 2     ) ..............  -> [a12]
overlay2("12", "abc", 2, 3     ) ..............  -> [a12 ]
overlay2("12", "abc", 2, 4     ) ..............  -> [a12  ]
overlay2("12", "abc", 2,-1     ) ..............  -> [a2c]
overlay2("12", "abc", 2,-2     ) ..............  -> [a12]
overlay2("12", "abc", 2,-3     ) ..............  -> [a 12]
overlay2("12", "abc", 2,-4     ) ..............  -> [a  12]
overlay2("12", "abc", 2,-3, ".") ..............  -> [a.12]
overlay2("12", "abc", 2,-4, ".") ..............  -> [a..12]
overlay2("12", "abc",-4,-1     ) ..............  -> [2abc]
overlay2("12", "abc",-4,-2     ) ..............  -> [12bc]
overlay2("12", "abc",-4,-3     ) ..............  -> [ 12c]
overlay2("12", "abc",-4,-4     ) ..............  -> [  12]
overlay2("12", "abc",-4,-5     ) ..............  -> [   12]
```

String-related BIF-Extensions Caseless & Negative Numbers, 11

- Examples

```
POS2("day","Saturday"   ) ..................... -> [6]
POS2("Day","Saturday"   ) ..................... -> [6]

------------------------------------------------------------

RIGHT2("abc d"  , 8     ) ..................... -> [   abc d]
RIGHT2("abc d"  , 8, ".") ..................... -> [...abc d]
RIGHT2("abc def", 7     ) ..................... -> [abc def]
RIGHT2("12",5,"0")        ..................... -> [00012]

RIGHT2("abc d"  , -8     ) .................... -> [abc d   ]
RIGHT2("abc d"  , -8, ".") .................... -> [abc d...]
RIGHT2("12",-5,"0")       ..................... -> [12000]
```

String-related BIF-Extensions Caseless & Negative Numbers, 12

- Examples

```
SUBCHAR2("abc",  3         ) ..................... -> [c]
SUBCHAR2("abc",  4         ) ..................... -> []

SUBCHAR2("abc", -3         ) ..................... -> [a]
SUBCHAR2("abc", -4         ) ..................... -> []
-------------------------------------------------------------

SUBSTR2("abc", -2          ) ..................... -> [bc]
SUBSTR2("abc", -2, -4      ) ..................... -> [  ab]
SUBSTR2("abc", -2, -6, ".") ..................... -> [....ab]
SUBSTR2('ab', -1, -3, .)  ..................... -> [.ab]
substr2("abc", -4         ) ..................... -> [ abc]
substr2("abc", -4,  , ".") ..................... -> [.abc]
substr2("abc", -4, 1, ".") ..................... -> [.]
substr2("abc", -4,-1, ".") ..................... -> [.]
```

String-related BIF-Extensions Caseless & Negative Numbers, 13

- Examples

```
SUBWORD2("   eins zwei drei ", 2)          ..... -> [zwei drei]
SUBWORD2("   eins zwei drei ", 3)          ..... -> [drei]
SUBWORD2("   eins zwei drei ", 2, 1)       ..... -> [zwei]
SUBWORD2("   eins zwei drei ", 2, 2)       ..... -> [zwei drei]

SUBWORD2("   eins zwei drei ",-2)          ..... -> [zwei drei]
SUBWORD2("   eins zwei drei ",-3)          ..... -> [eins zwei drei]
SUBWORD2("   eins zwei drei ", 2,-1)       ..... -> [zwei]
SUBWORD2("   eins zwei drei ",-2, 1)       ..... -> [zwei]
SUBWORD2("   eins zwei drei ",-2,-2)       ..... -> [eins zwei]
SUBWORD2("   eins zwei drei ", 2,-2)       ..... -> [eins zwei]
SUBWORD2("   eins zwei drei ",-2, 2)       ..... -> [zwei drei]
-------------------------------------------------------------------
UPPER2("abcdef"          ,  4)     ............. -> [abcDEF]
UPPER2("abcdef"          ,  3,  2) ............. -> [abCDef]
UPPER2("abcdef"          , -4)     ............. -> [abCDEF]
UPPER2("abcdef"          , -3, -2) ............. -> [abCDef]
```

String-related BIF-Extensions Caseless & Negative Numbers, 14

- Examples

```
WORD2("   eins zwei drei ", 2)          ........ -> [zwei]
WORD2("   eins zwei drei ", 3)          ........ -> [drei]

WORD2("   eins zwei drei ",-2)          ........ -> [zwei]
WORD2("   eins zwei drei ",-3)          ........ -> [eins]
-------------------------------------------------------------

WORDINDEX2("   eins zwei drei ", 2)           ... -> [9]
WORDINDEX2("   eins zwei drei ", 3)           ... -> [14]

WORDINDEX2("   eins zwei drei ",-2)           ... -> [9]
WORDINDEX2("   eins zwei drei ",-3)           ... -> [4]
```

String-related BIF-Extensions Caseless & Negative Numbers, 15

- Examples

```
WORDPOS2("EINS", " eins zwei drei "     )        -> [1]
WORDPOS2("eins", " EINS zwei drei "     )        -> [1]
WORDPOS2("EINS", " eins zwei drei ", "C")        -> [0]
WORDPOS2("eins", " EINS zwei drei ", "C")        -> [0]
WORDPOS2("EINS", " eins zwei drei ", "I")        -> [1]
WORDPOS2("eins", " EINS zwei drei ", "I")        -> [1]

WORDPOS2("  eins ", " eins zwei drei ",-1)       -> [0]
WORDPOS2("  eins ", " eins zwei drei ",-4)       -> [1]

WORDPOS2("  eins ", " eins zwei drei ",-1, "C")  -> [0]
WORDPOS2("  eins ", " eins zwei drei ",-1, "I")  -> [0]
WORDPOS2("  eins ", " eins zwei drei ",-4, "C")  -> [1]
```

Public Routine "dump2()"

- What is the "content" of a collection?
- Collection classes
  - List content in a ledgible ("human-centric") way
  - Allow any collection to be processed
  - Allow optional heading
  - Allow optional comparator (for sorting)
- dump2(collection [,heading] [,comparator])

Public Routine "dump2()" Example, 1

```
a2=.array~new
a2[1,2]="x"
a2[3,4]="a"
a2[3,5]="z"
a2[4,1]="A"
a2[7,5]="m"
call dump2 a2

------------------

type: The Array class: (5 items)

    # 1: index=[1,2] -> item=[x]
    # 2: index=[3,4] -> item=[a]
    # 3: index=[3,5] -> item=[z]
    # 4: index=[4,1] -> item=[A]
    # 5: index=[7,5] -> item=[m]
```

Public Routine "dump2()" Example, 2

```
d1=.directory~new
d1["x"]="x-value"
d1["a"]="a-value"
d1["z"]="z-value"
d1["A"]="A-value"
d1["m-index"]="m-index-value"
call dump2 d1

------------------

type: The Directory class: (5 items)

    # 1: index=[A]       -> item=[A-value]
    # 2: index=[a]       -> item=[a-value]
    # 3: index=[m-index] -> item=[m-index-value]
    # 4: index=[x]       -> item=[x-value]
    # 5: index=[z]       -> item=[z-value]
```

Public Routine "dump2()" Example, 3

```
r=.relation~new
r["x"]="x-value"
r["x"]="xyz-value"
r["x"]="x-value"
r["a"]="a-value"
r["z"]="z-value"
r["m-index"]="m-index-value"
r["A"]="A-value"
call dump2 r

------------------

type: The Relation class: (7 items)

    # 1: index=[A]       -> item=[A-value]
    # 2: index=[a]       -> item=[a-value]
    # 3: index=[m-index] -> item=[m-index-value]
    # 4: index=[x]       -> item=[an Array (3 items) id#_266390774]
    # 5: index=[z]       -> item=[z-value]
```

Routine "parseWords2"

- Definition of words differs from language to language
- Sometimes defining delimiters is easier
- Returns an array of parsed words
  - Either a single dimensioned array with words, or
  - A two-dimensional array containing the starting position, the length of the word and the next starting position

Routine "parseWords2" Example

```
str="Über den Wölkchen - oder Woelkchen - muss. die Freiheit wohl grenzenlos sein?"

deli=" -/\,:!?." || "09"x    -- delimiters
arr=parseWords2(str, deli)
do w over arr
   say "["w"]"
end

::requires "rgf_util2.rex"
--------

[Über]
[den]
[Wölkchen]
[oder]
[Woelkchen]
[muss]
[die]
[Freiheit]
[wohl]
[grenzenlos]
[sein]
```

Class "StringOfWords"

- Definition of words differs from language to language
- Sometimes defining delimiters is easier
- Submit a
  - String to be parsed
  - Optionally a string of reference characters
    - Defaults to the delimiter characters blank and tab
  - Optionally a character indicating how to use the reference characters
    - Defaults to "D"elimiter

Class "StringOfWords" Example

```
str="Über den Wölkchen - oder Woelkchen - muss. die Freiheit wohl grenzenlos sein?"

deli=" -/\,:!?." || "09"x    -- delimiters
sow=.StringOfWords~new(str, deli)
do w over sow  -- makeArray=wordArray, positionArray
   say "["w"]"
end
--------

[Über]
[den]
[Wölkchen]
[oder]
[Woelkchen]
[muss]
[die]
[Freiheit]
[wohl]
[grenzenlos]
[sein]
```

"sort2()" and "stableSort2()"

- Makes sorting available as a routine
- Makes it easy to indicate the sorting options
  - Allows the implicit usage of comparators defined in "rgf_util2.rex"
- Should be easier to use, than the array methods
  - sort(),
  - sortWith(),
  - stableSort() and
  - stableSortWith()

"sort2()" and "stableSort2()" Example, 1

```
mmyArray=.array~of(0, -1, " -1", "aNton", "AntoN", "BERta", "Anton", "berta", 99, 1E1)
tmpArray=sort2(myArray~copy)
call dump2 tmpArray, "sort"
---

sort: (10 items)

    #  1: index=[1]  -> item=[ -1]
    #  2: index=[2]  -> item=[-1]
    #  3: index=[3]  -> item=[0]
    #  4: index=[4]  -> item=[1E1]
    #  5: index=[5]  -> item=[99]
    #  6: index=[6]  -> item=[AntoN]
    #  7: index=[7]  -> item=[Anton]
    #  8: index=[8]  -> item=[aNton]
    #  9: index=[9]  -> item=[berta]
    # 10: index=[10] -> item=[BERta]
------------------------------------------------
```

"sort2()" and "stableSort2()" Example, 2

```
mmyArray=.array~of(0, -1, " -1", "aNton", "AntoN", "BERta", "Anton", "berta", 99, 1E1)
tmpArray=stableSort2(myArray~copy)
call dump2 tmpArray, "stableSort"
---

stableSort: (10 items)

    #  1: index=[1]  -> item=[-1]
    #  2: index=[2]  -> item=[ -1]
    #  3: index=[3]  -> item=[0]
    #  4: index=[4]  -> item=[1E1]
    #  5: index=[5]  -> item=[99]
    #  6: index=[6]  -> item=[aNton]
    #  7: index=[7]  -> item=[AntoN]
    #  8: index=[8]  -> item=[Anton]
    #  9: index=[9]  -> item=[BERta]
    # 10: index=[10] -> item=[berta]
------------------------------------------------
```

"sort2()" and "stableSort2()" Example, 3

```
myArray = .array~of("Zoe    24", "Fred   41", "Xavier 52", "Andy   40")
myArray~sortWith(.ColumnComparator~new(8, 2))
call dump2 myArray

---
type: The Array class: (4 items)

    # 1: index=[1] -> item=[Zoe    24]
    # 2: index=[2] -> item=[Andy   40]
    # 3: index=[3] -> item=[Fred   41]
    # 4: index=[4] -> item=[Xavier 52]
------------------------------------------------

myArray~sortWith(.InvertingComparator~new(.ColumnComparator~new(8, 2)))
call dump2 myArray
---
type: The Array class: (4 items)

    # 1: index=[1] -> item=[Xavier 52]
    # 2: index=[2] -> item=[Fred   41]
    # 3: index=[3] -> item=[Andy   40]
    # 4: index=[4] -> item=[Zoe    24]
------------------------------------------------
```

"sort2()" and "stableSort2()" Example, 4

```
myArray = .array~of("Zoe    24", "Fred   41", "Xavier 52", "Andy   40")
myArray=sort2(myArray, 8, 2)
call dump2 myArray

---
type: The Array class: (4 items)

    # 1: index=[1] -> item=[Zoe    24]
    # 2: index=[2] -> item=[Andy   40]
    # 3: index=[3] -> item=[Fred   41]
    # 4: index=[4] -> item=[Xavier 52]
------------------------------------------------

myArray=sort2(myArray, 8, 2, "D")
call dump2 myArray
---
type: The Array class: (4 items)

    # 1: index=[1] -> item=[Xavier 52]
    # 2: index=[2] -> item=[Fred   41]
    # 3: index=[3] -> item=[Andy   40]
    # 4: index=[4] -> item=[Zoe    24]
------------------------------------------------
```

"sort2()" and "stableSort2()" Example, 5 (Multiple Columns!)

```
myArray = .array~of("Zoe    24", "Fred   41", "Xavier 52", "Andy   40")
myArray=sort2(myArray, 8, 2, "N", 1, "I")
call dump2 myArray

---
type: The Array class: (4 items)

    # 1: index=[1] -> item=[Zoe    24]
    # 2: index=[2] -> item=[Andy   40]
    # 3: index=[3] -> item=[Fred   41]
    # 4: index=[4] -> item=[Xavier 52]
------------------------------------------------

myArray=sort2(myArray, 8, 2, "ND", 1, "ID")
call dump2 myArray
---
type: The Array class: (4 items)

    # 1: index=[1] -> item=[Xavier 52]
    # 2: index=[2] -> item=[Fred   41]
    # 3: index=[3] -> item=[Andy   40]
    # 4: index=[4] -> item=[Zoe    24]
------------------------------------------------
```

New Comparators "MessageComparator"

- Allows for determining a message name or a message object to retrieve the comparator value, e.g.
  - Attributes
  - Results of invoking methods
  - If collection, either message name, a message object or an array object with three elements:
    - name/message object, a flag [A|D], a flag [C|I|N]
- Optionally allows for caching the retrieved values

New Comparators "NumberComparator"

- Allows Rexx digital numbers as input
- Compares according to Rexx decimal arithmetic rules, e.g.
  - 1E2<123456

New Comparators "StringComparator"

- Allows to determine sorting type and order in an encoded string
  - Sorting order
    - A[scending]
    - D[escending]
  - Sorting type
    - C[ase dependent]
    - I[gnore case]
    - N[umeric]
  - E.g. encoded (in any order): "DN"

New Comparators "StringColumnComparator"

- Allows to state
  - starting position and optionally the length
  - Flag for Ascending or Descending order
  - Flag for Case dependent, Ignore case, and Numeric
- Alternatively, may supply a collection of
  - Column sort definitions

Roundup

- String-related BIF-Extensions
  - By default case-insensitive
  - Adding negative positioning and counts
- Routine "dump2()"
  - Allow a sorted dump of any collection
- Parsing words of any kind
  - Routine "parseWords2()", Class "StringOfWords"
- Routines "sort2()" and "stableSort2()"
  - Using new comparators