1 Entries Added to the ".local" Directory

- `.rgf.alpha` defined as: `.rgf.alpha.low || .rgf.alpha.upper`
- `.rgf.alphanumeric` the ASCII characters: `.rgf.alpha || .rgf.digits`
- `.rgf.alpha.low` the lowercase ASCII English characters `"abcdefghijklmnopqrstuvwxyz"`
- `.rgf.alpha.upper` the uppercase ASCII English characters: `.rgf.alpha.low~upper`
- `.rgf.digits` the ASCII characters defined as: `"0123456789"`
- `.rgf.non.printable` the non-printing ASCII characters: `xrange("00"x,"1F"x) || "FF"x`
- `.rgf.symbol.chars` the ASCII characters that may start a Rexx symbol defined as: `".!_?"`

2 New Routines for String Related BIFs

2.1 Ignoring the Case of English Letters

- `abbrev2(information, info [,[n-length] [,"I"|"C"]] )`
- `changeStr2(needle, haystack, newNeedle [,[n-count] [,"I"|"C"]] )`
- `compare2(string1, string2 [,[pad] [,"I"|"C"]] )`
- `countStr2(needle, haystack [,"I"|"C"] )`
- `lastPos2(needle, haystack [, [n-start] [,[n-length] [,"I"|"C"]] ] )`
- `pos2(needle, haystack [, [n-start] [,[n-length] [,"I"|"C"]] ] )`
- `wordPos2(phrase, string [,[n-start] [,"I"|"C"]] )`

2.2 Introducing Negative Numeric Arguments to BIFs

- `abbrev2(information, info [,[n-length] [,"I"|"C"]] )`
- `changeStr2(needle, haystack, newNeedle [,[n-count] [,"I"|"C"]] )`
- `delStr2(string, n-start [,n-length] )`
- `delWord2(string, n-start [,n-length] )`
- `lastPos2(needle, haystack [, [n-start] [,[n-length] [,"I"|"C"]] ] )`
- `left2(string, n-length [, pad] )`
- `lower2(string [, [n-start] [,[n-length]] ] )`
- `overlay2(new, target [, [n-start] [,[n-length] [, pad]] ] )`
- `pos2(needle, haystack [, [n-start] [,[n-length] [,"I"|"C"]] ] )`
- `right2(string, n-length [, pad] )`
- `subChar2(string [, [n-position] )`
- `subStr2(string, n-start [,[n-length] [,pad]] )`
- `subWord2(string, n-start [,n-count] )`
- `upper2(string [, [n-start] [,[n-length]] ] )`
- `word(string, n-position )`
- `wordIndex(string, n-position )`
- `wordLength(string, n-position )`
- `wordPos2(phrase, string [, [n-start] [,"I"|"C"]] )`

3 Making Sorting Easier (More "Rexxish")

3.1 New Public Routines (SORT2, STABLESORT2)

```
      sort2(array [,[A|D] [,[I|C]] ])
stableSort2(array [,[A|D] [,[I|C]] ])
```

```
     sort2(array [,comparator [,A|D] ])
stableSort2(array [,comparator [,A|D] ])
```

3.2 Creating Additional Comparator Classes for Sorting

3.2.1 The "NumberComparator" Class

The syntax for the .NumberComparator constructor is:

```
init( [.true|.false] [,[A|D] [,[IC][N]] ] )
```

```
      sort2(array [,[A|D] [,[I|C][N] ])
stableSort2(array [,[A|D] [,[I|C][N] ])
```

3.2.2 The "StringComparator" Class

The syntax for the .StringComparator constructor is:

```
init([A|D] [,[IC][N]] )
```

```
      sort2(array [,[A|D] [,[I|C][N]] ])
stableSort2(array [,[A|D] [,[I|C][N]] ])
```

3.2.3 The "StringColumnComparator" Class

The syntax for the .StringColumnComparator constructor is one of:

```
init( {pos [,length [,[A|D] [,[I|C|N]] ] ]}[,...] )
init( orderedCollection [, [defaultAD], [defaultICN] )
```

```
[stable]Sort2( array , {pos [,length [,[A|D] ,[I|C|N] ] ] }[,...] )
[stable]Sort2( array, orderedCollection )
```

3.2.4 The "MessageComparator" Class

The .MessageComparator is devised with the following syntax for its constructor:

```
init( messageName|messageObject [,bCached=.false] )
init( orderedCollection )
```

A messageName may be appended with a slash ("/") followed by a blank delimited list of options: "a[scending]" (default) or "d[escending]", optionally followed by "n[umeric]", "i[gnoreCase]" or "c[aseDependent]", if the returned value is of type .String.

```
[stable]Sort2( array, "M", messageName|messageObject[, ...] )
[stable]Sort2( array, "M", orderedCollection )
```

4 Parsing a String Into Words

4.1 The Public Routine "parseWords2"

```
parseWords2(string [,[ref=" "||"09"x] [,[kind="D"|"W"] [,returns="W"|"P"]])
```

4.2 The "StringOfWords" Class

The following methods are defined for the .StringOfWords class:

- `init( string [,[ref=(" "||"09"x)] [,[kind="D"|"W"]] ] )`
  The optional argument kind has either the value "D[elimiter]" (default) or "W[ord]" and determines whether the ref characters are used to delimit words or define the characters that constitute a word.
- `delWord(position [,count] )`
- `kind([newKind])`
- `makeArray`
- `positionArray`
  Returns a two-dimensional array, where the first dimension denotes the i^{th} word position and the second dimension denotes the start position ("[i,1]") and the length ("[i,2]") of the i^{th} parsed word.

- `reference([newReference])`
- `string([newString])`
- `subWord(position [,count] )`
- `word(position)`
- `words`
- `wordArray`
- `wordIndex(n)`
- `wordLength(n)`
- `wordPos(phrase [, [start] [, C|I]] )`

---

5 Helpful Routines for Debug Output

5.1 Routines for Creating Easier Legible Strings

- `enquote2(string [, quote='"'])`
- `escape2(string)`
- `pp2(object)`
- `ppIndex2(object)`
- `ppMethod2(methodObject [, indent=""])`

5.2 Routines to Ease the Dumping of Collections

- `dump2(collection [,[title] [,comparator] ] )`
- `makeRelation2(collection [,messageName|messageObject] )`