Operators

Operators are listed in precedence order, from highest to lowest. Operators from the same grouping are evaluated left to right.

\ ¬ Logical NOT (as a prefix)
+ Indicates a positive number (as a prefix)
- Indicates a negative (as a prefix)
** Raise to a whole number power
* Multiply
/ Divide
% Integer divide (return integer part)
// Remainder divide (return remainder)
+ Add
- Subtract
|| (abuttal) Concatenate (with no space)
( blank) Concatenate with blank in-between
== Strictly equal
=== \== Not strictly equal
>>() Strictly greater than
<< Strictly less than
>= >> =<> \<> Strictly greater than or equal to
( strictly not less than)
<= Vousette or equal to
( strictly not greater than)
= Equal to (also used for assignment)
/= \/= Not equal to
/> <= Greater than
< Less than
>= \< Greater than or equal to
( not less than)
<= -> <= Less than or equal to
( not greater than)
& And (both are true)
| Or (either is true)
&& Exclusive Or (either but not both are true)

Notes: not all implementations support all notations (for example: \¬). You can always ensure a higher precedence for a clause by enclosing it in parentheses: ( ).

Syntax

Symbols include: A-Z, a-z, 0-9, ? ! _
(many implementations add @ # $ .)
; (semi-colon) Statement separator
, (comma) Continues a line
/* */ Encloses comments
( ) Enclose function arguments, or groups for precedence

symbol = expression ; Assignment of value to variable
stem.tail Compound variable (tail may be multiple simple symbols separated by periods)
stem.index Simulating arrays with compound variables
'abc' or "abc" Character string notation
'0C'X or "0c"x Hexadecimal notation
'0101'b or "0101"B Binary string notation

Exceptions

NOVALUE Reference to an unitialized variable
ERROR Command to external environment indicates error
FAILURE Command to external environment failed
HALT External interrupt to the script (such as a Ctrl+C)
NOTREADY Unready I/O device
SYNTAX Syntax or runtime error in the script
LOSTDIGITS Arithmetic operation lost digit(s)

Flow of Control

IF-THEN IF-THEN-ELSE SIGNAL
SELECT CALL LEAVE
RETURN EXIT ITERATE
DO-END group DO-WHILE
DO n times DO-UNTIL
DO initialize-loop-counter TO limit BY increment

Help

Forum: https://groups.io/g/rexxla-members
Instructions

= (assignment) symbol = expression ;

ADDRESS | environment [ command ] [ redirection ] |
| [ VALUE ] expression [ redirection ] |

redirection is: WITH INPUT input_redirection
and/or: WITH OUTPUT output_redirection
and/or: WITH ERROR output_redirection

input_redirection is: [ NORMAL | STREAM | STEM ] symbol

output_redirection is: [ APPEND | REPLACE ]
plus a destination: [ NORMAL | STREAM | STEM ] symbol

ARG [ template ] → see HINTS for [template]
| name [ expression ] [ , [ expression ] ] . . . |

CALL | ON condition [ NAME trapname ] |
| OFF condition |

DO [ repetitor ] [ condition ]
| [ instruction_list ] |

END [ symbol ]

repetitor is either:
| symbol = expression_i [ TO expression_t ] |
| BY expression_b [ FOR expression_f ] |
or: expression_r
or: FOREVER

condition is either:
| WHILE expression_w |
| UNTIL expression_u |

DROP symbol [ symbol ... ]

EXIT [ expression ]

IF expression [] THEN [] instruction [ ELSE [] instruction ]

INTERPRET expression

ITERATE [ symbol ]

LEAVE [ symbol ]

NOP

NUMERIC DIGITS [ expression ]
| FORM [ SCIENTIFIC | ENGINEERING ] |
| [ VALUE ] expression |
| FUZZ [ expression ] |

OPTIONS expression

PARSE [UPPER] type [template] → see HINTS for [template]

type is: [ ARG | LINEIN | PULL | SOURCE | VERSION ]
or: VALUE [ expression ] WITH
or: VAR symbol

PROCEDURE [ EXPOSE variable_list ]

PUSH [ expression ]

QUEUE [ expression ]

RETURN [ expression ]

SAY [ expression ]

SELECT ; when_part [ when_part ... ] [ OTHERWISE [] ]
| [ instruction_list ... ] ] END ;

when_part is: WHEN expression [;] THEN [;] instruction
| label_name |

SIGNAL [ VALUE ] expression
| ON condition [ NAME trapname ] |
| OFF condition |

TRACE trace_setting | [ VALUE ] expression
| A – All |
| C-- Commands |
| E-- Errors |
| F-- Failure |
| I-- Intermediates |
| L-- Labels |
| N-- Normal |
| O-- Off |
| R-- Results |
| ?-- Toggles interactive trace On or Off |
| ( ? can be followed by a letter from the above list) |
| +n -- Skips number of pauses specified by whole number |
| -n –  Inhibits trace for number of clauses specified |

Hints

Character I/O –
| CHARS( [name] [,option] ) |
| CHARIN( [name] [,start] [,length] ) |
| CHAROUT( [name] [,string] [,start] ) |

Line I/O –
| LINES( [name] [,option] ) |
| LINEIN( [name] [,line] [,count] ) |
| LINEOUT( [name] [,string] [,line] ) |

Conversational PULL [template]
| I/O – PARSE PULL [template] |
| SAY [expression] |

Concatenation –
| apple='Apple' |
| say 'Candy'apple outputs: Candy-Apple |
| say 'Candy' apple Candy-Apple |
| say 'Candy' || apple Candy-Apple |
| say ‘Candy’ apple Candy-Apple |

Parsing for the ARG, PARSE, and PULL [template] –
| by Words (strings separated by blanks or spaces) |
| by Delimiters (delimiters identify input data elements) |
| by Numbers (for starting or relative column positions) |
| → Periods can be used to skip or exclude data elements |

Examples:
| phone = '212-855-1212'; sep = '‐'; |
| parse value phone with area_code (sep) prefix (sep) number |
| parse value phone with area_code 4 5 prefix 8 9 number |
| parse value phone with area_code +3 +1 prefix +3 +1 number |
| parse value phone with 4 sep +1 . 1 area (sep) prefix (sep) numb |
| parse value phone with . (sep) prefix (sep). → returns prefix |
| parse value phone with +8 number returns number |
| parse value phone with =9 number returns number |
| parse value phone with =9 number returns number |
Functions

ABBREV(information, info [,length])
ABS(number)
ADDRESS()
ARG([argnum [,option]])
BITAND(string1 [,string2 [,pad]])
BITOR(string1 [,string2 [,pad]])
BITXOR(string1 [,string2 [,pad]])
B2X(binary_string)
CENTER(string, length [,pad])
CENTRE(string, length [,pad])
CHANGESTR(needle, haystack, newneedle)
CHARIN([name] [,start] [,length])
CHAROUT([name] [,string] [,start])
CHARS([name] [,option])
COMPARE(string1, string2 [,pad])
CONDITION([option])
COPIES(string, times)
COUNTSTR(needle, haystack)
C2D(string [,length])
C2X(string)
DATATYPE(string [,type])
DATE([option_out [date [,option_in]]])
DELSTR(string, start [,length])
DELWORD(string, start [,length])
DIGITS()
D2C(integer [,length])
D2X(integer [,length])
ERRORTEXT(error_no)
FORM()
FORMAT(number [,before] [,after])
FORMAT(number [,before] [,after] [,expp] [,expt]])
FUZZ()
INSERT(string, target [,position] [,length] [,pad])
LASTPOS(needle, haystack [,start])
LEFT(string, length [,pad])
LENGTH(string)
LINEIN([name] [,line] [,count])
LINEOUT([name] [,string] [,line])
LINES([name] [,option])
MAX(number1 [,number2]...)
MIN(number1 [,number2]...)
OVERLAY(string1, string2 [,start] [,length] [,pad]])
POS(needle, haystack [,start])
QUALIFY([streamid])
QUEUED()
RANDOM(max)
REVERSE(string)
RIGHT(string, length [,pad])
SIGN(number)
SOURCeline([line_number])
SPACE(string [,length] [,pad])
STREAM(name [,option [,command]])
STRIP(string [,option] [,char])
SUBSTR(string, start [,length] [,pad])
SUBWORD(string, start [,length])
SYMBOL(name)
TIME([option_out [time [,option_in]]])
TRACE([setting])
TRANSLATE(string [,tableout] [,tablein] [,pad]])
TRUNC(number [,length])
VALUE(symbol [,newvalue] [,pool])
VERIFY(string, reference [,option] [,start])
WORD(string, wordno)
WORDINDEX(string, wordno)
WORDLENGTH(string, wordno)
WORDS(string)
X2B(hexstring)
X2C(hexstring)
X2D(hexstring [,length])

Options for Functions

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Datatype</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>C</td>
<td>A</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>B</td>
</tr>
<tr>
<td>E</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>M</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>N</td>
<td>M</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>N</td>
<td>S</td>
</tr>
<tr>
<td>S</td>
<td>R</td>
<td>U</td>
</tr>
<tr>
<td>U</td>
<td>S</td>
<td>W</td>
</tr>
<tr>
<td>W</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Special Variables

RC Return code from a command, or a SYNTAX error code
SIGL Line number of last instruction that caused a jump to a label
RESULT Set by a RETURN instruction in a subroutine