The REXX SourceBook

Frequently Asked Questions About REXX

Last Revised: August 12, 1994

Eric Giguère
giguere@watcom.on.ca

The REXX SourceBook

Copyright Information

Introduction

This document is intended to serve as a useful reference for REXX-related information. It aims for *breadth* as opposed to *depth*, and references to other material are given where appropriate. Suggestions and updates should be sent to the author in an attempt to keep this document relevant and up-to-date.

Readers will notice the prevalence of OS/2-related materials in this document. Most of the REXX-related activity at this time is occurring on the OS/2 platform. This document is not intended to be OS/2-specific. The author is quite happy to include information on other platforms if you pass it on to him.

More information on REXX can also be had from the REXX Language Association. See below for details.

A. What Is REXX?

REXX is a programming language designed by Michael Cowlishaw of IBM UK Laboratories. In his own words: "REXX is a procedural language that allows programs and algorithms to be written in a clear and structured way."

REXX doesn't look that different from any other procedural language. Here's a simple REXX program:

```
/* Count some numbers */

say "Counting..."
do i = 1 to 10
    say "Number" i
end
```

What makes REXX different from most other languages is that it is also designed to be used as a **macro language** by arbitrary application programs. The idea is that application developers don't have to design their own macro languages and interpreters. Instead they use REXX as the macro language and support the REXX programming interface. If a REXX macro comes across an expression or function call that it cannot resolve, it can ask the application to handle it instead. The application only has to support the features that are specific to it, freeing the developer from handling the mundane (and time-consuming) task of writing a language interpreter. And if all applications use REXX as their macro language, the user only has to learn one language instead of a dozen.

B. REXX and the Internet

Networks connect computers in various ways for the exchange of data. The terminology is a bit confusing to the new user. Here are the definitions this document uses:

- **Usenet:** Not really a network, just the set of machines that exchange network news. Network news is really an extended form of electronic mail that groups messages from individuals into newsgroups that users can read using special newsreaders.
- **Internet:** The worldwide network based on TCP/IP protocols. Besides being able to receive mail and newsgroups, these machines can use programs like **ftp** and **telnet** to communicate with other machines in real time. Most Internet machines are Unix-based.
- **BITNET:** The worldwide network that connects many IBM mainframes. BITNET users can also transfer files using methods that are incompatible with those of the Internet.

Newsgroups

The Usenet group **comp.lang.rexx** exists for discussion of REXX in all its variations. Anything posted to this newsgroup also gets sent to the REXXLIST mailing list (see below) and vice-versa.

Other newsgroups of interest are machine-specific. Recommended groups are **comp.os.os2.programmer.misc** and **comp.sys.amiga.programmer**.

FTP Sites of Interest

FTP is a file transmission protocol used on the Internet to transfer files between machines. The transfers are done in real time and usually require that the user have an account on both machines. However, many machines on the Internet support what is known as **anonymous FTP**, which allows users on other machines access to a limited set of files without requiring an account. Some of the more interesting sites that offer this service are:

rexx.uwaterloo.ca	General repository for REXX-related information, including free REXX interpreters for Unix and DOS. An XEDIT clone for Unix and OS/2 may also be found here. Look under /pub/rexx.

ftp.pvv.unit.no	The official home of Regina, one of the free Unix interpreters. Archives of the messages in comp.lang.rexx as well as RexxLA messages are also maintained here. Check under /pub/rexx.
ftp-os2.cdrom.com	General OS/2 archives. Look under /pub/os2.
ftp.luth.se	
wuarchive.wustl.edu	General Amiga archive. Look under /pub/aminet.

Mailing Lists

Mailing lists are similar to newsgroups but use normal electronic mail to deliver the messages. The following mailing lists are mostly BITNET-based but are accessible from the Internet as well:

List name	BITNET Node	Internet Address	Discusses
REXXLIST	UCF1VM	ucf1vm.cc.ucf.edu	REXX in general
AREXX-L	UCF1VM	ucf1vm.cc.ucf.edu	Amiga REXX
PC-REXX	UCF1VM	ucf1vm.cc.ucf.edu	Personal REXX
REXXCOMP	UCF1VM	ucf1vm.cc.ucf.edu	IBM's REXX compiler
TSO-REXX	UCF1VM	ucf1vm.cc.ucf.edu	TSO REXX
VM-REXX	UCF1VM	ucf1vm.cc.ucf.edu	VM/SP REXX
UREXX-L	(none)	liverpool.ac.uk	Unix REXX

To subscribe to any of these lists, send a one-line message to the address LISTSERV@node, where node is the BITNET node or Internet address for the list you wish to join. In the body of your message should be the line

```
SUBSCRIBE list-name your--full-name
```

as in

```
SUBSCRIBE UREXX-L Eric Giguere
```

You will then be subscribed to the list and messages will start arriving in your mailbox. To send a message to the list, simply mail it to listname@node, as in UREXX-L@liverpool.ac.uk. Note the distinction between the LISTSERV address and the listname address. You can receive help by sending a HELP message to the LISTSERV address.

Note that some of these mailing lists may be available on Usenet in the form of newsgroups with names starting with "bit.listserv". Ask your system administrator if you're not sure.

Thanks to Scott Ophof for providing this summary.

Gopher Service

Gopher clients may find REXX-related information at the site gopher.pvv.unit.no (Europe) and index.almaden.ibm.com (North America).

C. Free REXX Products

Interpreters

There are at least three REXX interpreters available for free on the Internet. The first two are Unix based and are well-supported by their authors. The third is an MS-DOS interpreter.

Regina is Anders Christensen's REXX interpreter for various flavours of Unix and VMS. It is fairly complete and Anders even has an API for developers. It also apparently can be ported to OS/2. Anders can be reached at anders@pvv.unit.no. Regina's official home is ftp.pvv.unit.no.

REXX/imc is Ian Collier's REXX interpreter for SunOS, though it has also been ported to other Unix systems. Ian can be reached at imc@prg.ox.ac.uk.

BREXX is Bill Vlachoudis' REXX interpreter for MS-DOS. The interpreter is not complete but is quite small. Bill can be reached at bill@donoussa.physics.auth.gr.

All three interpreters are available for anonymous FTP on rexx.uwaterloo.ca in the /pub/freerexx directory, each interpreter in its own subdirectory. Regina and REXX/imc are in source form, BREXX is only available as binary.

REXX-Aware Text Editors

Also on rexx.uwaterloo.ca in the /pub/editors directory is the text editor THE by Mark Hessling (mark@snark.itc.gu.edu.au). THE is a full-featured XEDIT/KEDIT clone (by XEDIT here we mean the IBM mainframe text editor, not the X Windows editor xedit) with REXX support. THE is available in versions for OS/2 and Unix. THE's official home is on ftp.gu.edu.au in /src/THE.

D. Commercial REXX Products

Interpreters

REXX interpreters are available commercially for a wide variety of systems and come standard on some operating platforms such as the Amiga, OS/2 and the IBM AS/400 and mainframes (VM, TSO, VSE). The following vendors sell REXX interpreters:

The Workstation Group [Various UNIX platforms, also VMS]
6300 River Road
Rosemont, IL 60018
(800) 228-0255 (US only)
sales@wrkgrp.com

Quercus Systems [DOS, Windows, Windows NT, OS/2]
P.O. Box 2157
Saratoga, CA 95070
(408) 867-7399
(800) 440-5944 (US & Canada)
75300.2450@compuserve.com

Simware [Novell Netware]
2 Gurdwara Road
Ottawa, Ontario, Canada K2E 1A2
(613) 727-1779

IBM also sells REXX interpreters for AIX and Netware.

Compilers

Although REXX is usually thought of as an interpreted language, it can also be compiled. The following vendors all sell REXX compilers:

Dineen Edwards Group [Amiga]
19785 West 12 Mile Road, Suite 305
Southfield, MI 48076-2553
(313) 352-4288

IBM [VSE, MVS/TSO and VM/CMS]
Contact your local representative

Systems Center [VM/CMS]
1800 Alexander Bell Drive
Reston, VA 22091

Visual Development Environments

There are three REXX-based visual development environments available for OS/2:

VX-REXX **Watcom International**
415 Phillip Street
Waterloo, Ontario
Canada N2L 3X2
Phone: (519) 886-3700
Fax: (519) 747-4971

VisPro/REXX **HockWare**
315 N. Academy St., Ste. 100
Cary, NC 27513
Phone: (919) 380-0616
Fax: (919) 380-0757

GpfRexx **Gpf Systems**
10 Falls Road
Moodus, Conn. 06469
Phone: (203) 873-3300
Fax: (203) 873-3302

REXX-Aware Text Editors

Clones of the popular XEDIT editor are available for Unix from the **Workstation Group** (see address above) and for DOS and OS/2 from **Mansfield Software**. **Tritus** sells an ISPF/PDF text editor with REXX support for OS/2. **One Up** sells SourceLink, an integrated development environment for OS/2 with REXX macro capabilities. **Command Technology** sells the SPF/PC editor.

Mansfield Software
P.O. Box 532
Storrs, CT 06268
Phone: (203) 429-8402
Fax: (203) 487-1185

Tritus
3300 Bee Caves Road, Suite 650
Austin, Texas 78746
Phone: (512) 794-5800
Fax: (512) 794-3833

One Up
1603 LBJ Freeway, Suite 200
Dallas, Texas 75243
Phone: (800) 678-0187

Command Technology
1040 Marina Village Parkway
Alameda, CA 94501
Phone: (800) 336-3320

The OS/2 Enhanced Editor (EPM.EXE), which is bundled with OS/2, also has REXX support. Use its online help and search for the 'rx' command.

REXX Extensions

A number of vendors sell extensions to REXX:

- **Quercus Systems** (address above) sells **REXXLIB** (a collection of over 150 REXX extension functions), **REXXCOMM** (a function package for accessing serial ports from REXX) and **REXXTERM** (a full-featured asynchronous communications program).
- **SofTouch Systems** (address below) sells the **GammaTech REXX SuperSet/2**, a collection of over 300 REXX extension functions for OS/2.
- **dSoft Development** (address below) sells the **dbfREXX** function library that lets you read and write dBASE files from OS/2 REXX.

SofTouch Systems
1300 S. Meridian, Suite 600
Oklahoma City, Okla. 73108-1751
Phone: (405) 947-8080
Fax: (405) 632-6537

dSoft Development
4710 Innsbruk Drive
Houston, Texas 77066
Phone: (405) 360-3045
Fax: (713) 537-0318

E. REXX and ANSI

The American National Standards Institute (ANSI) sets national standards for various things in the United States, including programming languages. The X3J18 REXX Standards Committee is currently defining a formal standard for the REXX language, using Mike Cowlishaw's book as its base document. The Committee meets 3 or 4 times a year and holds ongoing discussions throughout the year by electronic mail. Members of X3J18 are mostly REXX implementors, but anyone can participate. The Committee intends to release a draft standard next year. More information can be had from the vice-chair, Neil Milsted at nfnm@wrkgrp.com.

Note that public ANSI documents relating to X3J18 can be had using the LISTSERV service at PSUVM on BITNET or by Gopher to gopher.pvv.unit.no on the Internet.

F. The REXX Language Association

The REXX Language Association is an independent organization dedicated to promoting the use of the REXX programming language. Activities of RexxLA include:

- Maintaining an electronic mailing list server where members share information.
- Distributing a quarterly newsletter.
- Providing electronic resources for access to language expertise, hints and tips, example programs, product sources, and other valuable information.
- Developing resource guides, both printed and electronic, for publications, products, training and language experts.
- Developing educational, guest speaker, and publicity programs to promote the use of REXX.
- Participating in the work of standards bodies.
- Promoting integration of REXX into all operating systems and as the common scripting language for a wide array of software.
- Cooperating with the REXX Symposium in providing an annual conference forum.

Join today and start reaping the benefits available from an international consortium of individuals, corporations, vendors, authors, and experts.

For more information, contact the REXX Language Association by mail or fax:

RexxLA Membership
6300 North River Road, Suite 501
Rosemont, Illinois 60018
FAX: (708) 696-2277

Or by electronic mail at rexxla@wrkgrp.com.

G. The REXX Symposium

The REXX Symposium is an annual conference devoted to REXX, attended both by users and vendors, held at the beginning of May. It is sponsored by the Stanford Linear Accelerator, with the cooperation of the RexxLA. The 1995 conference is still being planned.

H. REXX Bibliography

Magazines

OS/2 magazines such as *OS/2 Magazine*, *OS/2 Professional* and *OS/2 Developer* typically have columns or articles featuring REXX. The publishers of *OS/2 Developer* (Miller Freeman) recently printed a special issue devoted to REXX on OS/2 called the *REXX Report*.

Books

Mike Cowlishaw and Linda Green have kindly provided the following partial bibliography of REXX books:

The REXX Language -- M.F. Cowlishaw

English:	ISBN 0-13-780735-X	Prentice-Hall, 1985
	ISBN 0-13-780651-5	2nd edition, 1990
German:	ISBN 3-446-15195-8	Carl Hanser Verlag, 1988
	ISBN 0-13-780784-8	P-H International, 1988
Japanese:	ISBN 4-7649-0136-6	Kindai-kagaku-sha, 1988

The REXX Reference Summary Handbook -- Dick Goran
ISBN 0-9639854-1-8, CFS Nevada Inc., 1994.

Modern Programming Using REXX -- Robert P. O'Hara and David R. Gomberg

English:	ISBN 0-13-597311-2	Prentice-Hall, 1985
	ISBN 0-13-579329-5	2nd edition, 1988

REXX in the TSO Environment -- Gabriel F. Gargiulo
ISBN 0-89435-354-3, QED Information Systems Inc.
320 pages, 1990.

Using OS/2 REXX -- Gabriel F. Gargiulo
ISBN 0-894-35449-3, QED Publishing Group

Practical Usage of REXX -- Anthony S. Rudd
ISBN 0-13-682790-X, Ellis Horwood (Simon & Schuster), 1990

Using ARexx on the Amiga -- Chris Zamara and Nick Sullivan
ISBN 1-55755-114-6, Abacus Books, 1991

The REXX Handbook -- Edited by Gabe Goldberg and Phil Smith III
ISBN 0-07-023682-8, McGraw-Hill, 1991

Programming in REXX -- Charles Daney
ISBN 0-07-015305-1, McGraw-Hill, 1992

Command Language Cookbook -- Hallett German
ISBN 0-442-00801-5, Van Nostrand Reinhold, 1992

OS/2 2.1 REXX Handbook -- Hallett German
ISBN 0-442-01734-0, Van Nostrand Reinhold, 1994

OS/2 REXX: From Bark to Byte -- International Technical Support Organization (IBM)
IBM Document Number GG24-4199-00, 1993

REXX: Advanced Techniques for Programmers -- Peter Kiesel
ISBN 0-07-034600-3, McGraw Hill, 1992

REXX Tools and Techniques -- Barry Nirmal
ISBN 0-89435-417-5, QED Publishing Group, 1993

The ARexx Cookbook -- Merrill Callaway
ISBN 0-96-327730-8, Whitestone, 1992.

Writing OS/2 REXX Programs -- Ronny Richardson
ISBN 0-07-052372, McGraw Hill

I. Common REXX Coding Errors

The following list of common REXX coding errors is derived from a list included in the online documentation for Watcom VX-REXX.

1. Blank space where it does not belong

In REXX expressions, blank space is interpreted as an implicit concatenation operator -- the terms are concatenated with a blank in between. As a result, REXX will interpret many mistyped statements as an expression involving the blank concatenation operator.

For example, inserting a blank after a function name in a function call changes the meaning of the expression from:

```
text_upper = translate( text )
```

to:

```
text_upper = "TRANSLATE" || " " || text
```

Blank space also plays a special role in the PARSE instruction. Compare the following:

```
parse arg a b c
parse arg a, b, c
```

The first line parses the first argument passed to the routine into three parts, while the second line sets the three variables to the value of the first three arguments passed to the routine.

2. Function calls versus the CALL statement

When you call a routine that returns a result, you must enclose the parameters in parentheses:

```
text = VRGet( "EF_1", "Value" )
```

Always assign the value of a function to a variable, or use the CALL statement as described below. Otherwise REXX will pass the return value to the default host environment, leading to strange and possibly damaging behaviour on some systems.

If you are calling a routine that does not return a value, or you wish to ignore the return value, you should use the CALL instruction:

```
call VRSet "EF_1", "BackColor", "Blue"
```

Note that there is no comma between the name of the routine and the first parameter. Note also that parentheses are not used when using the CALL instruction.

3. Line continuation

The comma is used in REXX to split clauses across two or more lines. For example:

```
call foo a, b, c
```

can also be written as:

```
call foo a, ,
         b, ,
         c
```

It's easy to forget the second comma when breaking a line in the middle of a function parameter list.

4. Omitted arguments

REXX allows arguments to be omitted. Be careful not to omit arguments by accident, such as including an unnecessary comma:

```
call foo , a, b, c
```

5. Undefined variables

It is not a syntax error to use undefined variables in REXX. Undefined variables are defined to have their own name translated to uppercase as their value. As a result it is often difficult to find programming errors that are a result of using undefined variables. Some tips:

- Add a SIGNAL ON NOVALUE statement to the main section of your programs. This will cause the system to issue a syntax error if you use an undefined variable.
- Be careful to include the period when referring to stems. The variables "A" and "A." are unrelated.
- Misspelled commands will often be interpreted as undefined variables. The line:

  ```
  sy 'hello'
  ```

  (which should be "say 'hello'") will be interpreted as:

  ```
  "SY" || " " || 'hello'
  ```

  and will be send to the default command host for execution.

6. Using expressions in the tail of a compound symbol

The tail of a compound symbol can only be a simple variable, as in:

```
ok = A.I
```

Literals are not allowed. For example, the following is interpreted as a concatenation between "A." and "name":

```
bad = A."name"
```

Expressions are also not allowed. You must first assign the value of the expression to a symbol and then use that symbol:

```
J = I - 1
ok = A.J
```

J. Frequently Asked Questions

1. Is REXX better than <some other language>?

Short answer: Yes. No. Maybe. Does it matter?

Long answer: This question wastes a lot of bandwidth in comp.lang.rexx and other newsgroups. Every language has its good points and its bad points. Some people love REXX, some people hate it. Use a language that suits your needs.

2. Why does my OS/2 REXX program run more quickly the second time?

When you run a REXX CMD file for the first time, a tokenized version will be stored on disk using the OS/2 extended file attributes. (You can see how big the tokenized version is by using the /N option on the DIR command.) If a tokenized version exists AND the file has not been modified, CMD.EXE will use the tokenized version instead of parsing the source.

Note that there is a 64K limit on the size of an extended attribute entry, so very large REXX programs do not benefit from this automatic tokenization.

3. How can I return multiple values from a function?

REXX does not provide any support for returning more than a single value from a function. If you wish to return multiple values, you must devise an alternate scheme. A simple solution is to concatenate the values together into a single string and on return from the function use the PARSE instruction or the various string functions to split the string back into its elements. Don't forget that you can use non-printable characters (such as '00'x) to separate the data -- REXX will correctly handle such strings.

There may also be other alternatives available to you if you are using an external function library that lets you store data in separate memory pools or in disk files.

4. Why does linein, lineout, charin or charout fail?

Most versions of REXX (ARexx is an exception) use implicit file opening. That is, each time you reference a file in a LINEIN, LINEOUT, CHARIN or CHAROUT function, REXX will open the file for reading or writing if the file is not already open. However, some operating systems like DOS and OS/2 impose limits on the number of files that can be open simultaneously, usually around 20 or so. After the limit has been reached, any further attempts to open another file will fail. That is why it is always good

practice to close a file when you're done with it. In OS/2 this is done using the STREAM function, as follows:

```
call stream "c:\foo.out", "command", "close"
```

The STREAM function can also be used to open files, query their sizes and seek into the file. Consult your REXX documentation for specific instructions for your interpreter.

5. How do I iterate over all the tails in a stem variable?

One of the features REXX lacks is a function to return a list of defined tails. There are external libraries that provide functions to do so, but if that is not an option then the only solution is to maintain your own list of tails in a string and use the PARSE instruction or the WORDS function to traverse the list.

6. How do I REXX-enable my application?

REXX-enabling an application means being able to run REXX macros within an application. This information is very system-specific, so the best place to start is with the documentation provided with the REXX interpreter.

For OS/2, there are several sources of information. The most basic information is found in the OS/2 Toolkit, which includes the REXXSAA.H header file and the *REXX Reference* online document. The REXX Report (see above) includes a couple of articles on the subject. Sample source code comes with the OS/2 Toolkit and is also available on rexx.uwaterloo.ca in the directory /pub/os2/vxrexx as VX-REXX Tech Notes #1 and #7 (vxtech01.zip, vxtech07.zip -- neither tech note requires that you own VX-REXX). OS/2 technical conferences such as ColoradOS/2 or the IBM Technical Interchanges often includes sessions on this topic.

For ARexx, a book was available from Commodore, but with the latter's demise it is unclear whether the book is still available.

7. How do I do inter-process communication in REXX?

Again, this is system-specific. The ARexx interpreter is built on a messaging model, making it very simple to do inter-process communication, but the OS/2 REXX interpreter has no such features, though in some cases queues can be used to achieve the desired effect.

8. How do I use global variables in my REXX programs?

The scope of variables is controlled by the PROCEDURE instruction. If a routine is declared with the PROCEDURE instruction, only those variables exposed using the EXPOSE instruction are available to the routine. If no PROCEDURE instruction is used, all of the caller's variables are available to the callee. Here is a simple example:

```
        a = 10
        b = 20
        call first
        call second
        call third
        exit

    first:
        say "first -- a is" a "b is" b
        return
```

```
second: procedure
    say "second -- a is" a "b is" b
    return

third: procedure expose a
    say "third -- a is" a "b is" b
    b = 30
    call first
    return
```

Running this program yields the following output:

```
first -- a is 10 b is 20
second -- a is A b is B
third -- a is 10 b is B
first -- a is 10 b is 30
```

Use the PROCEDURE instruction to keep variables local to a procedure, using EXPOSE to explicitly expose any "global" variables. The only catch is that you have to make sure you expose the variables inside every procedure.

One way to define and use global variables is to use a stem called "Globals." and define all your procedures like this:

```
Foo: procedure expose Globals.
```

Then at the top of you program initialize the Globals stem and assign appropriate values to your global variables:

```
Globals. = ''
Globals.!NeedToSave = 0
Globals.!TmpDir = "D:\TMP"
```

The tail names in this example are all prefixed with '!', though you could also use an underscore ('_'). This is just a convention used to avoid this kind of problem:

```
Globals.TmpDir = "D:\TMP"
call Foo
say Globals.TmpDir
exit

Foo: procedure expose Globals.
    tmpdir = "foo"
    Globals.TmpDir = tmpdir
    return
```

It's a subtle bug that has to do with how REXX interprets stem tails.