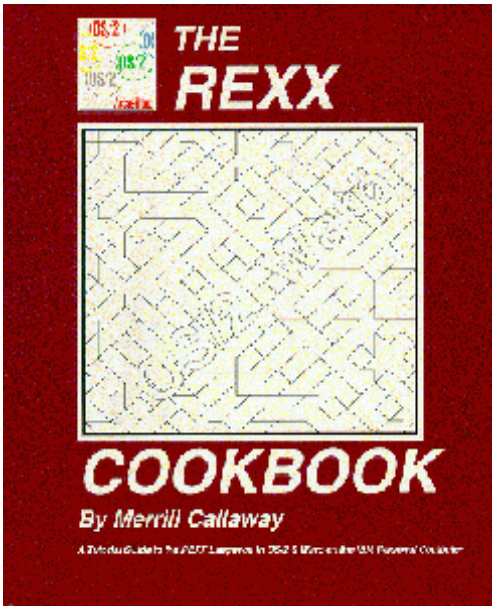
EDM/2

# The Codesmith's Library

Written by Carsten Whimster






## Introduction

In The Codesmith's Library, I focus on development books and materials. I have written this column from the point of view of an intermediate PM C programmer and intermediate REXX programmer. Pick up whichever book strikes your fancy, and join the growing group of people following our PM programming columns. I have already reviewed a number of beginner's books, and will try to concentrate a bit more on intermediate techniques and special topics from now on.

Please send me your comments and thoughts so that I can make this column what you want it to be. I read and respond to all mail.

The REXX Cookbook takes a different tack at teaching REXX, and uses lots of small and medium-sized applications, explained one at a time.

## Errata

As you will have noticed by now if you are a regular reader, I have changed the name of my column to something a little easier to pronounce and remember, and a little more in line with the rest of the columns around here. The reason is not that my development directory is now called Devel or that my BookReview directory is now called Review, but rather it is one of the last steps in unifying the image of the magazine a bit more, and also to have something a little more memorable. I hope you like it.

The OS/2 API Project is coming along nicely, with a fairly settled format, and more than 15 APIs on site, as well as over 75 #define's. Have a look, and if you have the time, chip in by writing up one or more APIs and sending them to me. The more people help out, the quicker this project will become very helpful. Currently we have an online version on the Web, a zipped HTML version, and an INF version. Check out the guidelines for submission at

http://www.undergrad.math.uwaterloo.ca/~bcrwhims/public_html/os2/api/index.html

In case you don't know what this is all about, it is the project of, so far, four people who decided to start documenting the entire OS/2 API and distribute it as a free document, to help out people who can't afford the IBM documentation, or have something better to spend their money on. I figure it can only help the number of OS/2 programs out there if more people have access to this information. We are also hoping to improve on the usefulness of the IBM documentation, and improve the structure. Eventually, if enough people help out, The OS/2 API Project may be

the documentation of choice.

## The REXX Cookbook

This book is refreshingly different in its approach to teaching a language. There are many who claim that there is no substitute for real code. This book is for them:

1. Introduction to the REXX Language
2. REXX Basics: Files, Strings, and Arrays
3. Parsing Made Easy
4. Numbers, Logic, and Recursion
5. Sorting and Working with Arrays and Lists
6. Debugging, Tracing, and Interrupting REXX
7. Extending and Enhancing OS/2 REXX
8. REXX Macros and Application Command Sets
9. REXX Queues and IPC Programs
10. VX-REXX and Event Driven REXX Programs
11. Using REXX and Postscript Together

Chapter one is a brief introduction to the book, its purpose, and its relationship to other books. The author stresses that you should have a reference book as a companion to The REXX Cookbook, and recommends either The REXX Language, Cowlishaw, Programming in REXX, Daney (see last month), or both. This chapter ends with a quick outline of the material in the rest of the book, and an introduction to REXX, both in general, and on OS/2.

The special typography of this book is explained, but in spite of what I am sure are good reasons, I had a measure of difficulty with the usage of bolding in this book. Technical terms are printed in bold, as well as other items, and reading a book which has several bolded words on each page is distracting for me. I find that starting several lines above where the bold term is, my eye starts straying down every time I pass over the bolded word, and this continues on even below the line with the word. The sheer number of boxes used around the code on some pages is also somewhat distracting. My final "package" complaint is that the book just will not lie open. It insists on closing itself unless I put something on it, so one of my Jugglebug juggling dice was quickly doing double duty.

Chapter two covers some basic REXX concepts, and the author suggests that programmers familiar with REXX can skim the first half of the chapter. I/O is the first language feature covered, oddly enough. One wouldn't expect a cookbook to introduce French cuisine this quickly, and I/O seems a slightly difficult way to start off a book for beginners. However, the sample programs are very well documented, so this doesn't really ever pose a problem, as long as you are not a newcomer to programming in general, but rather, a REXX novice. In addition to the comments, pseudo-code is used extensively to develop the programs. One nice feature about this book that sets it a little apart from other similar books is that you will find little non-REXX helpful hints here and there. In chapter 2 you will find a hint on how to turn REXX syntax-expansion on in EPM. Another nice touch is that for some of the programs, the author actually takes you through an improvement cycle, to add features to a program, or perhaps to clean up the code, or make it a bit more efficient. It is nice to learn this way, seeing what makes one program better than another. Occasionally, there is an example which isn't as clear as it could be, such as the woodworker and the template example. I had to read that one through a few times to see what was meant, even though I knew the material he was presenting.

Chapter three covers parsing, which of course is where much of the strength of REXX lies. A program called PARSETST is used to demonstrate the templates and pattern matching which is done with PARSE. This program is unfortunately quite complex for a beginner, and probably will even turn a few intermediate REXX programmers' heads, so I feel that perhaps the source for this program should have been left as a black box until later in the book. Callaway does warn the reader, however, that he or she may want to skip the source for now, and come back to it much later, and just learn how to use the program meanwhile. In some ways I wish that I was still a novice REXX programmer, so that I could evaluate the quality of the learning that takes place with this program's aid. It is hard for me to look at it, and say "yes, this is effective" or "no, this doesn't work". I feel a little skeptical about it, but I will give this method of teaching the PARSE instruction the benefit of the doubt. I personally lean towards simpler, easier to understand examples as my preferred way of learning, but there are probably others who learn fine with a test program like this. What the program does is to emulate the operation of the PARSE instruction with the use of the INTERPRET instruction. You basically type in the parsing source, and the template, and the program types out the result in an easy to examine manner. It works a bit like REXXTRY in this respect. It sounds simple enough, but there is a little bit of

black magic needed to make it work, and this is where the complexity comes in.

Chapter four covers the areas of arithmetic, boolean logic, recursion, and concatenation. These concepts are demonstrated with the help of some examples. One particularly interesting one goes like this: you are on a game show on TV, and you have to try to guess which one of three doors hides the sportscar that you can win. The probability of choosing the correct door is 1/3 for you, obviously. Then a wrinkle is introduced. AFTER you have chosen a door, the game show host opens one of the two remaining doors. The door he opens never has the car behind it, so now you know that the car is behind one of two doors. The one you chose, or the third door. Then the host lets you choose between staying with the door you chose or switching to the remaining closed door. What should you do? I have put the answer in the summary so that you don't have go crazy, but try to figure it out yourself first, and no it isn't the obvious answer. I have also included a small VX-REXX program called TheDoor.exe which demonstrates the solution so that you can watch it graphically. You will need a VROBJ.DLL, 2.0c or later. This can be had from ftp.watcom.on.ca somewhere. I have not included the source since it is fairly simple, but if you are interested, mail me. If you want an explanation of the answer, buy the book. There is also a couple of proofs in the book, and believe me, it will take two proofs and an explanation to convince you that the given answer is correct. There are also some other neat examples, among which is a classic math puzzle.

The next chapter covers sorting, and how to use arrays and lists. Again, the sample programs are larger, more complex, more real, and ultimately more helpful than those of most other books. If you like reading real code, you'll love this book. A full explanation is given for both bubblesort, and the shellsort. I vaguely disagree with his prognosis of shellsort as "very fast and efficient", but then again covering mergesort and quicksort and variants thereof in a book for beginning REXX programmers is probably not warranted, and shell sort isn't all that bad. It is certainly more efficient than the types of things novices come up with . I tutor some 1st and 2nd year CS courses from time to time, and after handing over a large pile of marked and sorted assignments to a mob of eager students, the resulting smaller return pile is invariably sorted with what I call "the student messy sort". Callaway goes to great lengths to explain exactly how bubble sort and shellsort work, listing output for whole runs and bolding dozens of words in the process, and eventually I could say that if I had not already known how these two sorts worked, I certainly would now. After the sorts are explained, a neat program is built which uses the shell sort and some other code to analyze word counts and redundancies in a given input file. Callaway's solution is quite interesting and uses a neat feature of REXX's arrays which enables using a single line of code to check whether a word has already been encountered without searching through any kind of list first. In the end I wonder if sorting the list first, and then removed duplicates (which at this point are consecutive) would not have been faster than removing duplicates, albeit in an efficient manner, and then sorting, but then that is probably just the UNIX in me (cat file | sort | uniq or something similar).

Chapter six is simply the most comprehensive treatment of the debugging features in REXX I have ever seen. It has lots of screen shots, lots of examples, lots of explanations and lots of... lots of... well, I am sure that there is lots of something else in there as well, such as, perhaps, e's or something. In any case, it is thorough. Tracing, condition handling, error traps, options, interactive and numeric tracing, the RC variable, saving traces, REXX interrupts, default condition handlers, and special variables all get their time in the limelight, but because this is a fairly straightforward topic, that is all I am going to say about that.

Chapter seven discusses something that maybe of some interest to lurking C programmers out there, namely extending and enhancing REXX through DLLs. Then again, there is probably not enough depth here to warrant buying the book for this purpose only. RexxUtil is given a cursory treatment, and a program is given which saves some important system files, such as the autoexec.bat, config.sys, OS2*.INI, and so on. Pretty rudimentary stuff, but if you don't have a handle on this yet, potentially very useful. This program uses some exception handling, but otherwise doesn't seem to belong with DLLs. There is no C code in here by the way.

Chapter eight is next. In this chapter, and in chapter nine to a lesser extent, macros are introduced. Many OS/2 applications (well, some...) include a REXX API for customizing or programming the application. In this chapter we are introduced briefly to programming and customizing TSPF and EPM. If you are interested in EPM programming, however, you really want to use E, not REXX, but that is another story. REXX will let you solve simple problems in EPM. The ADDRESS command is fairly well explained, and some real-life examples are given here. The REXX interfaces in TSPF and EPM are very different and serve to highlight the variety of methods which software companies can use to REXX- enable their applications. In chapter nine we are treated to a much more complicated example involving queues, REXX APIs, DB2/2 (now DB/2, I believe), and Ami Pro (soon to be Word Pro). Using some REXX glue,

a queue is set up which enables the transfer of some data from DB2/2 to an Ami Pro document. Unfortunately, there wasn't enough depth in the queue treatment to tell me why my own REXX queue project doesn't work yet, but then again it could be a VX-REXX problem rather than a REXX problem (i.e.. my problem). This large example involves both SQL and the APIs of both products, and is probably an excellent problem for programmers who need to do this type of thing day-to-day.

Callaway apparently has a great deal of respect for VX-REXX, which incidentally I share with him, but I feel that he ought to at least mention VisPro-REXX which is also supposed to be an excellent package. It is possible that the book was published before VisPro-REXX came out, but since it was published in 1995, I doubt it. I can't afford to run both, so I haven't seen VisPro-REXX at all, but people who have used both say that they are more similar than different, although they both have their strengths and weaknesses. One of VisPro REXX's strengths is its close ties to VisPro C and VisPro C++ which saves heaps of time if you want to move a visual REXX project to C or C++ later. I have in fact needed this very feature recently myself, but VX-REXX unfortunately doesn't offer a similar feature. In any case, chapter ten introduces VX-REXX. After the introduction, the sample program from chapter nine is expanded with a graphical interface done with VX-REXX. This interface enables graphically constructing an SQL query, which you can then fire off to DB2/2. The data is then formatted correctly and sent back to Ami Pro. Quite an impressive example for a introductory REXX book, I must say.

The final chapter of the book, chapter eleven, covers a way of interfacing REXX with a PostScript printer. Because REXX runs on the computer and PostScript runs on the printer, the combination is a very powerful way of controlling the output of an application. After running quickly through how PostScript uses its LIFO stack (redundant, I know) to print output, and certain basics of the PostScript language, a program is presented which prints envelopes via PostScript. And if that isn't enough, Callaway then briefly outlines how ANY form you use or want to use or need to use can be generated fairly easily with REXX and PostScript.

Finally, a good bibliography is listed, and a funny index. This index is **very** thorough, covering virtually every page that a certain word occurs on, rather like a computer-based search program, but unfortunately, these long lists of page numbers are not indented, so the numbers and the index entries run together, making it harder to read. Still, a good index is nothing to shrug at.

## Summary

Take it or leave it, this book is different! It has some really excellent aspects, but also some minor drawbacks. On the one hand the typography can be very distracting, at least for me, and it really needs an O'Reilly style binding to lay open properly. When a book advocates that it should be used in conjunction with a computer and the REXX capabilities of OS/2, it really should be able to come through on that. On the other hand, it jumps very quickly into fairly complicated material (for a novice programmer). On the gripping hand it gives a very unusual, interesting and thorough treatment to the material it covers, using sample programs that are far more interesting and useful than those presented in most books. These examples cover virtually every aspect of REXX that you are likely to come across, even if the text doesn't always describe the algorithms and paradigms in detail. When it comes down to a final analysis, I feel that some of the drawbacks are personal opinion, others aren't, and overall it is a very good to excellent book, so therefore I give it an A-.

The answer to the door problem is that you should switch. If you stay, you have one third chance of winning, whereas if you switch, you have two thirds! Now go buy the book to find out why.

The REXX Cookbook, Callaway

- Whitestone.
- The REXX Cookbook (ISBN 0-9632773-4-0) MSRP $27.95US
- The REXX Files Disk (ISBN 0-9632773-5-9) MSRP $14.95US
- (Both of the above) REXX Cookbook Deluxe Set (ISBN 0-9632773-6-7) MSRP $42.90US
- Beginning and Intermediate REXX programmers
- Mark: A-

## Reviewed Books

Our Bookstore has reviews and links to Amazon for all books we have reviewed in the past, as well as several books waiting to be reviewed, and others which we recommend.

FEEDBACK SEARCH TOP BACKWARD FORWARD