How REXX Helped Me Hit the Ground Running in UNIX

Lois White
Stanford Linear Accelerator Center

Pages 360-362

How REXX Helped Me Hit the Ground Running in UNIX

or

How I Stopped Worrying & Learned to Love Typing in Significant Mixed Case

Lois White
SLAC Computing Services
Stanford Linear Accelerator Center, Stanford, California

6th International REXX Symposium
Stanford Linear Accelerator Center, Stanford, California
May 3, 1995

Since the early 80's, REXX has been my language of choice in VM/CMS for a system of execs and SAS programs which manipulate and store data and produce daily, month-to-date, and month-end reports on resource utilization and performance. Actually, I created and still maintain three VM service machines which keep track of utilization and some performance statistics for VM/CMS, a VAX cluster, and SLAC's telephone system.

About three years ago, it was announced that UNIX would be the future direction for physics computing at SLAC. In order to prepare for the coming of UNIX, I took an introductory course and was appalled...shell scripts??...no REXX??? I looked at bourne, korn, and c shell scripting and said to myself "I thought we had progressed beyond EXEC and EXEC2". Then someone mentioned that **perl** was the scripting language to use in UNIX. When I looked at **perl** I found it to be powerful and concise, but nearly impossible to decipher without comments on each line! I was looking at a steep learning curve here!

As the new RS6000 machines began arriving, it quickly became evident that I would need to start accounting for resource utilization on them and get an idea of how much they were being used and by whom. The accounting software included with the RS6000 Base Operating System was minimal, but it produced most of the information we needed. However, it didn't store data in a form that is readily used for reporting purposes. It also didn't clean up after itself very well, allowing directories to grow indefinitely. A great deal of manual intervention was required in order to save data and produce reports on a regular basis and to keep directories from filling up. It was clear that I needed to have more than just crontab entries and SAS software and I needed it soon!

Writing **perl** or shell scripts would have accomplished the task but I estimated that it would take some time, perhaps months, to become proficient enough to do what was needed in a lot less time. In early 1993, I learned that The Workstation Group's uni-REXX had arrived at SLAC and I felt as if I'd been saved. All subsequent references to REXX in UNIX in this paper refer specifically to the use of The Workstation Group's uni-REXX.

In the VM/CMS world I had already developed techniques for data storage, data manipulation,

and report production which I could now use in the UNIX world with the arrival of REXX. Close to ten years of experience using VM/CMS REXX meant that I wouldn't have to spend three to six months achieving the skills I needed before I could start managing data and producing reports for the rapidly multiplying UNIX machines. Furthermore, the uni-REXX manuals were written for users making the transition from VM/CMS to UNIX. With all this encouragement, I jumped in and wrote my first REXX script in UNIX which copies AIX disk accounting data from the file where it gets replaced each time disk accounting runs to another directory where it is saved and used later to analyze disk usage on a long term basis.

Since then I have developed a system of crontab entries and REXX and SAS programs which store and manipulate UNIX accounting and performance data and produce daily, month-to-date, and monthly UNIX resource utilization and performance reports. On each of the (now) sixty-four RS6000s where we collect accounting data, a REXX program executes daily which copies the locally stored data to a generally accessible directory where accounting data for all sixty-four machines is stored. After that, another crontab entry on one machine executes a REXX program which initiates the daily data processing and subsequent analysis reports by executing SAS and additional REXX programs. This daily program's decisions on which processes to start are based on the current date, day of the week, and other criteria. There are several other cron-initiated REXX programs which take care of data copying, directory cleanup, and daily checking of all automatic processes. In addition, there are several REXX programs which are executed manually to rerun processes which failed and to perform tasks such as large scale data backup.

Outside of the accounting and performance area, I have used the same techniques to develop a system of crontab entries and REXX and SAS programs to produce regular reports and graphs analyzing network performance data for our UNIX systems.

In conclusion, REXX enabled me to "hit the ground running in UNIX" because it has:

Familiarity:
I had many years of experience writing REXX code.

Portability:
I was able to transfer several large pieces of REXX code from VM/CMS to UNIX and use them, often without modifications. These included useful algorithms for cleaning up old files, finding dates, creating file names, etc. to use as parameters for execs and SAS programs.

Communication with UNIX:
It is possible to issue UNIX commands from REXX programs. By using the POPEN instruction or function, it is possible to read output from UNIX commands and to test return codes. Sometimes I've found that using a UNIX command is more efficient than doing the same task with REXX code, e.g. file editing using the sed utility instead of REXX's linein/linout functions.

Readability:
REXX code is relatively easy to understand and usually doesn't require adding comments on every line in order to remember "*Why* did I do *that*?!"