What I Learned This Month: My "Secret" Tips and Tools

Scott Chapman
American Electric Power

This month I want to take a break from what I've recently learned to share a few tools that I use to make my job easier.

First is a very common tool that I think is often under-utilized: Excel. If you want an open source alternative, Calc in either Libre Office or Open Office has many of the same features.

Pivot tables and charts are apparently not widely understood, but actually aren't very complicated and really make dynamic data analysis very easy. For example, if you want to analyze a dataset with a bunch of servers' utilizations over time, create a pivot table with the variables you want to analyze. The chart created from that pivot table will have buttons that will let you easily select subsets of data for analysis. You might start by looking at the utilization of just production servers but then refine your analysis by selecting just one data center. Then perhaps decide to look at all servers in that data center. With the correct pivot chart you can quickly switch between subsets of data with just a few clicks of the mouse. If you don't understand pivot charts and you regularly do such analysis, set aside an hour or two to learn how to use pivot charts. You'll be glad you did!

Did you know that Excel can import data directly from web URLs as well as text files stored locally? And that you can have the data refreshed automatically every time you open the spreadsheet? You can also set up pivot tables/charts to automatically refresh when the underlying data changes. I have several reports that are simply Excel spreadsheets that reference a CSV[1] file stored on the mainframe and served up by the web server. When the spreadsheet is opened it automatically reloads the data from the mainframe and automatically refreshes the pivot charts. While I could (and have) done such reporting with other tools, I find that generating interactive graphs with pivot charts in Excel is usually easier and faster to implement than alternatives such as SAS/Graph or coding up an interactive HTML page.

Another tool that I more recently started using is Apache Ant.[2] This tool is designed primarily as a tool to help automate the process of compiling, packaging, and deploying a software product or application but can be used for more generic tasks as well. It's written in Java so it runs across many platforms, including the mainframe. It can compile code, copy, move and delete files, update file contents, FTP files to servers, execute programs or scripts, and even script telnet sessions. I now have Ant scripts for a couple of different projects that make deploying code from my PC to the mainframe as simple as entering a single command. In some cases those scripts even telnet into the mainframe and run existing scripts on the mainframe to finish the deployment process. When I have to deploy across multiple systems, I can just kick off the process and let the Ant script update the code on all the systems.

I've also used Ant on the mainframe itself to script the execution of a Java application. Because Ant runs on multiple platforms, I can use the same Ant execution script on my PC as I do on the mainframe. I don't have to maintain parallel versions of a batch file on my PC and a shell script on the mainframe.

Ant scripts are stored in XML files and it does take a little time to learn how to design scripts to do more complicated tasks, but I've found the learning effort to have been well-spent.

[1] Comma Separated Values: simply a text file with data values separated by commas.

[2] Apparently named for the acronym "Another Neat Tool" and because ants are good at building things. See http://ant.apache.org

If you write code of any type you will eventually come to a problem that defies all of your attempts at figuring it out. Usually the problem behavior is perceived as "bizarre" or "impossible". In short you have no idea what the problem could be and so no idea how to fix it. This can be extremely frustrating and lead to a long debugging session. Recently I've tracked down such problems in an SQL statement, ACS routines[3], CSS rules, and even a Scratch[4] program.

In all cases the method for finding the problem was the same: delete code until the problem goes away. The concept is simple enough. First make sure you have a good backup of your code. Then start deleting segments of code and check to see if the confusing behavior has changed or disappeared. If it has, then the problem is likely somewhere in the block of code that was deleted, so put part of it back and repeat the process until you've narrowed the problem down to the line of code with the problem. At that point if the problem isn't obvious, you at least know exactly where it is, which greatly limits your debugging scope.

Even though it's very effective, "delete code until it works" sounds inelegant. Perhaps I should call it a "binary bug search". That sounds much more sophisticated. In the case of the ACS routines, there was a comma where a period should have been, resulting in syntactically correct, but completely broken, code. Three or four people overlooked those few extra pixels and it wasn't until we used the "binary bug search" that we found it.

I hope you found something interesting and useful in these tips. As always, if you have questions or comments, you can reach me via email at sachapman@aep.com.

[3] Automatic Class Selection routines are the code that assigns Systems Managed Storage attributes to datasets that are allocated on the mainframe.

[4] Scratch is a visual programming language that is aimed primarily at introducing children to programming concepts. See http://scratch.mit.edu