iSeries to UNIX: First Contact

In this three-part series, Thibault Dambrine describes the basics of the UNIX Operating System and first impressions, from the perspective of an iSeries programmer.

- *Part 1 describes the basics of UNIX, how it came to be and how the UNIX architecture differs from that of the iSeries.*
- *Part 2 deals with some of the basic UNIX commands, directories, security and the scheduling system.*
- *Part 3 is all about shell scripting, the UNIX equivalent of the iSeries' CL language.*

Thibault Dambrine

(Part III of III)

A Shell Scripting Minute

By Thibault Dambrine

Shell scripting in UNIX is the equivalent of CL programming on iSeries. It is part of the basic skill set necessary to effectively use a UNIX based system. The following examples are designed to give the reader a basic idea of three concepts that are key to the UNIX shell way of programming:

1) UNIX, unlike OS/400 or even NT is case-sensitive.
2) The UNIX shell scripting language uses a lot of pipes and directional operators to pass the output of one command to the input of the next. Think of it as having the *OUTFILE of your iSeries command piped directly into the command that follows without transiting via a physical file. (see **Example 1**)
3) In UNIX, even the simplest commands can have many optional flags that alter the command's default behavior. When unsure about which flags to use, you can get UNIX's equivalent of OS/400's F4 by typing the man command ("man" stands for "manual") followed by the command name. "man" pages are typically cryptic and provide minimal explanation. Many times, I find myself looking for other sources of information, either on manuals or on the Internet, which has a lot to offer for budding UNIX shell programmers.

Example 1: Piping the results of pf into the command grep:

```
ps -ef | grep costing
```

"ps" is the UNIX equivalent to WRKACTJOB. This script command produces a list of active jobs using "ps -ef", (-ef represents two directives for "ps". The "e" tells "ps" to display information on every process and the "f" to display full information about the process). The output of "ps -ef" is piped into the "grep costing" command which will filter every line outputted by "ps -ef" except lines which contain the word "cost".

Example 2: Pick an option!

```
ls -ltra *.*
```

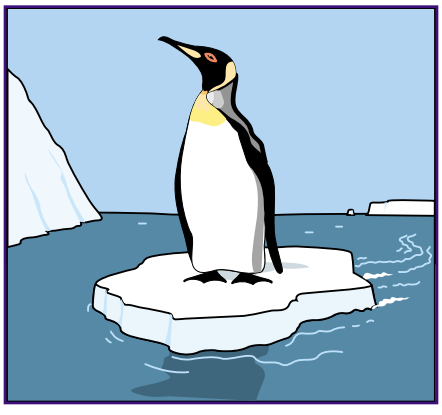
In this example, the meaning of the options attached to the command is:

`-l` lists files in long form, giving more details.

`-t` lists the contents in order of time saved, beginning with the most recent

`-r` lists the contents in reverse order

`-a` shows all files, including hidden files



The "ls" command in UNIX is similar to the "dir" command in DOS. It is both the simplest (nothing much to printing the contents of a directory) and most complicated command in UNIX. (There are a total of 23 options!) When used all by itself, it only yields very basic information on the content of a directory. What I mean is that by just typing "ls", you will not know from the results, which entries are directories and which entries are files. You will also not know which are executable and which are the ones you have access to.

A Shell and Oracle Exposure

This is a bit of a sidebar, but important enough that I think it should be mentioned. This point does also indicate how open UNIX really is. While most UNIX proponents tout the fact that it is an "open" operating system, this openness may be also misused.

As I said earlier on, UNIX does not have a database system of its own. In this circumstance, one of the most popular database solutions for UNIX is Oracle. In my short UNIX experience, I have noticed an interesting security point to be cautious of , which may not be found in most text books: Here is how it goes:

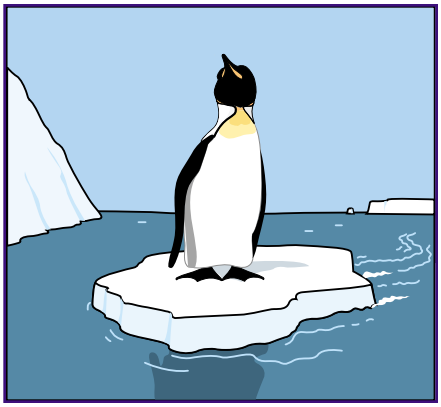
1) In a UNIX/Oracle situation, it is possible to submit a UNIX script which will then launch an Oracle PL/SQL stored procedure (like a program).

To make this combination of UNIX Shell script and Oracle PL/SQL stored procedure work, the UNIX Shell script needs to pass a user profile and a password to the Oracle PL/SQL procedure, in order for it to be able to access the database and execute the job. One way to do this is to create a script which can accept the Oracle user profile and password as a parameter. It would then use this user profile/password in combination with the name of the PL/SQL procedure to tell Oracle to execute the commands within. The shell command, written that way, would look like shellscript_to_execute_oracle_job OracleUser/OraclePassword

2) The UNIX equivalent of the iSeries SBMJOB is "nohup". The format of this command is nohup command_arguements &, where the "&" is the end of the command to submit.

3) If you wanted to submit the job described in point (1) in UNIX, you would say nohup shellscript_to_execute_oracle_job Oracleuser/Oraclepassword &

4) The UNIX equivalent of iSeries WRKACTJOB is the command "ps" In a situation where a command such as the one described in point (3) is submitted in UNIX, anyone with enough UNIX authority to run a "ps -ef" command can see plain and clear the content of the submitted job (including the password!). In this case, "shellscript_to_execute_oracle_job Oracleuser/Oraclepassword". Fortunately, this is NOT the only way to design such a script, but it is something to keep in mind.



ISeries and QSHELL

Over the course of this article, I have mentioned QSHELL several times. This article would not be complete without a word about this remarkable iSeries development.

To be fully Java compliant, IBM needed to supply a JDK (Java Development Kit) on its iSeries eServer that could run standard Java commands such as: **`java, javac or javadoc,`** the same way other systems could. Remember, Java was first developed by SUN, who also manufactures UNIX systems. OS/400 CL commands just would not do. The OS/400 Team resolved the problem by supplying a new "shell" on the iSeries, appropriately named QSHELL. In addition to being able to run UNIX style commands, QSHELL is also the access door to the iSeries IFS (Integrated File System).

If you have not yet explored the IFS on your system, I strongly encourage you to do so. It is easily accessible (use the iSeries command **`STRQSH`**) and you are there.

`STRQSH` will start a UNIX shell type of command session on your iSeries. You can think of it as a green-screen QCMD session, with a different set of commands. These commands are mostly UNIX style commands. To see what commands are available, you have to do a directory listing (use **`ls /usr/bin`**) .

I have tried it myself, and have found it works fine. My experience is that QSHELL works in a standard UNIX fashion for most (but not all) commands. While there is no help or access to man pages in QSHELL, if you have access to the Internet, there is no shortage of documentation for UNIX commands (look for "UNIX man pages" in any search engine). Two other UNIX commands that are noted for their absence in QSHELL: "more" and "vi". There are thus no convenient file viewers (except "cat" which is not practical for larger files) or editors on the initial version of QSHELL. On its FAQ, IBM says:

- The Source Entry Utility (SEU) allows you to edit file members in the QSYS.LIB file system.
- The Edit Files (EDTF) CL command allows you to natively edit both stream files and QSYS.LIB file members.
- A mapped network drive to AS/400 allows you to edit files using your favorite PC editor.

For my initial shell scripts, I have typed my text file on a PC and FTPed it over to the iSeries to run.

Although QSHELL is not completely new, I have to admit it is new to me. It seems that quietly, UNIX has slipped-in through the back door of OS/400. If you are curious about UNIX, if you have no access to a "pure" UNIX machine, if you are reluctant to partition your PC to install a copy of Linux at home, this may be a good spot to get familiar with UNIX commands and the UNIX type of environment.

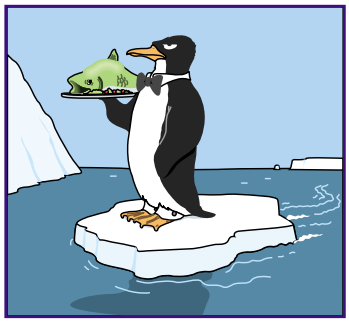
Conclusion

At some point, while writing this article, I had the following idea for a conclusion.

In the field of Biology, the first person to express a systematic presentation of evolution was **Jean Baptiste de Lamarck** (1774-1829) in 1809. Lamarck described his vision of evolution as "the inheritance of acquired characteristics." No kidding! It nearly sounds like "Object Oriented Programming"! In 1859, Charles Darwin (1809-1882) published "The Origin of Species" in which he coined the term "natural selection" to describe the process by which organisms with favorable variations survive

If one could think of operating systems as actual living and evolving organisms, who would use innovation, research and marketing as food to grow, one could actually apply Lamarck's and Darwin's theories to the UNIX's and the OS/400's of the world.

If you think about it, you could say that Digital's CPM or Microsoft's Xenix for example have not evolved, or not evolved fast enough.



From lack of support and/or lack of market, they eventually died. By contrast, similar to the dog-like creature that a few million years ago evolved into today's horse; DOS, fueled by Microsoft, mutated into Windows. The horse, user friendly, rather pretty and widely used, is also fragile. If it breaks a single leg, you have to put it down and start training a brand new horse.

The S/34 OS, fueled by IBM, became S/38 CPF and then OS/400. I would compare it to the mammoth, which evolved into the elephant. Famous for its quiet stability, it did evolve with time, but while doing so, kept its general shape. The elephant is famous for its long memory of past events – a feature which compares well with OS/400's advanced system logging features.

UNIX, fueled by the rich intellectual grounds of AT&T's Bell Labs and Berkeley, hit the market running right from the start. A living fossil, I would compare it to the crocodile. Long time survivor, it has kept a lot of its old proven concepts. Not pretty, not friendly, but it has stood the test of time. It will be there for a while yet. One more thing about the crocodile – like UNIX, it comes in several brands that look similar but have subtle differences, like the alligator and the caiman.

Looking back to May 17, 1994, I now suspect Mr. Carthra was referring to PASE (Portable Application Solutions Environment) – an integrated solution to allow UNIX applications to run on AS/400. As far as I have seen, PASE was a good idea but it did not change anybody's life. Close but no cigar! Will the iSeries's ability to run Linux on a logical partition be different?

This all begs the question, which one of these operating systems will be the next survivor? Which one will be the next to go extinct in the process of evolution?
Eventually, I decided that all this Darwin stuff would be a bit too deep for such a light-hearted piece. Here is my real conclusion to this article. UNIX is not particularly hard to learn. It does however require a good road map if one needs to understand and/or investigate anything in this operating system. There is a large perceived cultural gap between it and OS/400. But perhaps it is caused more by what we do not know than by what we do know. Case in point: OS/400 and QSHELL now coexist quite comfortably on the same box.

The primary aim of this article was to share with fellow RPG/CL colleagues some of my own experiences as a new UNIX user. QSHELL is now part of the iSeries Operating System and iSeries now also supports Linux. It might be a matter of time before you are exposed to some form of UNIX influence, without having to even leave the iSeries world.

Through my descriptions and examples in this article, I hope to have given you, iSeries programmers, a feel for what UNIX is like. Overall, it has a different look but in practical terms, it does many of the same things as OS/400. Of course, reading someone's impressions is a start, but it is no substitute for trying it yourself. I encourage every iSeries programmer who reads these words to fire up QSHELL, see what you can do with it, explore your system and see the possibilities. If you are interested in a more in-depth look at UNIX, buy a copy of Linux and install it on a spare PC. With this you can have your own risk-free UNIX training ground. TUG

Thibault Dambrine *works as an independent contractor in Calgary, Alberta. You can visit the iSeries / AS/400 related website he manages at http://www.tylogix.com or e-mail him directly at thibault.dambrine@tylogix.com.*