The Hamilton C shell

A command-line interface for OS/2 that borrows the power of Unix

BY KENNETH G. GOUTAL

The Hamilton C shell is an implementation under OS/2 version 1.1 of the Unix C shell. The C shell is a command-line interpreter with commands for running programs, spawning threads and processes, and manipulating directories and files. It also has high-level programming constructs that allow you to execute those commands in other than simple sequential order.

Users who are used to working with Unix but are totally unfamiliar with OS/2 will find that this tool eases the transition greatly. With a couple of minor exceptions, the Hamilton C shell is an exact functional duplicate of the C shell found on Unix systems. The few exceptions are clearly documented and are mostly a matter of cleaning up ugly irregularities in the traditional C shell or adapting to the OS/2 environment itself.

For example, the character used by OS/2 to punctuate path names is the backslash rather than the forward slash used by Unix. This precludes the use of the backslash to mean "take the next character literally, and ignore any special meaning it usually has in the Unix C shell." OS/2 normally uses the caret for this purpose, so the Hamilton C shell follows suit. As a result, the caret cannot be used for short-form history substitution as it is on the Unix C shell, and the percent sign is used instead. However, these choices are controlled by environment variables, so if you really must use the characters you're used to, you can.

OS/2 users who have never seen Unix will benefit from the Hamilton C shell's interface, which is much more powerful than the CMD.EXE supplied with OS/2. Control constructs include `switch`, `until`, and `while`. Contrary to what the product name suggests, the syntax of these constructs is not much like that of the C programming language. It is, however, faithful to the Unix C shell syntax.

You can store a sequence of commands, called a shell script, in a file and execute the sequence with the `source` command. You can also define shell procedures, which are stored within the shell itself until you exit from it. A procedure may be a simple series of commands that you perform frequently. On the other hand, it may be complex, its function and behavior based partly on command-line arguments, the environment, and user interaction.

A shell procedure may even be recursive. For instance, I wrote a recursive procedure called "dirt" that pretty-prints a directory tree using indentation to indicate common parentage. When called with no arguments, it prints the directory tree from your current working directory on down. When called with one argument, it prints the directory tree from the specified directory on down. When it calls itself recursively, it passes a subdirectory and the current indentation as arguments.

Several commands can be "stacked" on a single command line. For instance,

```
cat *.txt >alltxt; wc -w
 alltxt
```

concatenates all the files with the extension ".txt" into a file called "alltxt." It then counts the words in that file.

Commands can be "piped" together with a single command. For example,

```
cat *.txt | wc -w
```

does the same thing as the previous example, but without the need for an intermediate file. The output of the first command is fed directly into the next.

A "history" mechanism makes short work of retyping commands either to correct an error in the command just typed or to reuse a complicated command issued several minutes or even hours before. For example, `!cat` reexecutes the entire command line in the previous example; `!!:s/txt/log` does this again, except that it counts the words in all the files with the extension ".log" instead of ".txt." `!23` reexecutes the 23rd command you had typed since starting the current copy of the Hamilton C shell.

The Hamilton C shell provides several useful utility programs as well as several built-in commands. The most common Unix commands are provided either as built-in commands or as external utilities: `rotd`, `sleep`, `source`, `cat`, `chmod`, `cp`, `date`, `du`, `ls`, `mkdir`, `more`, `mv`, `pwd`, `rm`, `rmdir`, `tee`, `touch`, and `wc`. Others, such as `awk`, `grep`, and `sed`, are on the way. Some of these do the same things as commands in CMD.EXE, some do the same things as other utilities supplied with OS/2, while others have no OS/2 equivalent. For convenience, many common CMD.EXE commands and OS/2 utility names have been defined as aliases of the built-in commands and of the utilities included with the shell.

PRODUCT INFORMATION

Hamilton C shell 1.03
Hamilton Laboratories
13 Old Farm Rd.
Wayland, MA 01778-3117
(508) 358-5715
$350; requires IBM AT, PS/2, or compatible; OS/2 1.1 with Presentation Manager
Inquiry 71

This product appears to work perfectly and exactly as described in the manual. It also works blindingly fast, for two reasons. First, it compiles commands and procedures before executing them, rather than slavishly interpreting them. Second, it makes use of the OS/2 threads whenever the Unix C shell would use processes.

I did encounter some installation difficulties. This was due partly to my unfamiliarity with OS/2, partly to some lack of clarity in the manual, and partly to the lack of a program to supervise the installation. It was not due to any bugs in the software or errors in the manual.

Hamilton Laboratories reports that, to date, it has a 24-hour turnaround time for fixing serious bugs, with updates sent out the next day. Based on the company's responsiveness when I called with some questions about installation, I'm inclined to believe this. The fixes to less serious bugs are included in subsequent releases. The company can be reached by phone, mail, or electronic mail.

In conclusion, it looks like we have a winner. I expected to be put off by OS/2, but after I got the Hamilton C shell installed and tailored to my tastes, it was just like being back home. I had some difficulties at first adapting to the minor incompatibilities with the Unix C shell, but after that it was smooth sailing. I expect that someone who is not used to using the Unix C shell would actually have an easier time with the Hamilton C shell than with the Unix C shell. Naturally, it will take a while to learn how to exploit all of its power, but its features can be used independently and learned only when necessary. There are some improvements that could be made in the area of installation and in the manual, but these are not prohibitive and may have already been made by the time you read this. All in all, the Hamilton C shell is a much-needed and well-done product. ■

Kenn Goutal is a technical support engineer at Interbase Software Corp. of Bedford, MA. He can be reached via uucp as kenn@rr.MV.COM.