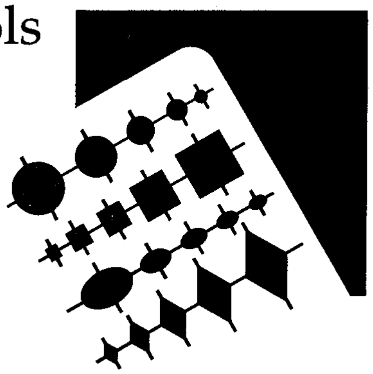
Software Tools



Hamilton C Shell Announcement

by Douglas A. Hamilton

Hamilton C shell is a brand new application for OS/2: not one line of code was ported from or developed on anything but OS/2. In response to the question often asked, "What can you do on OS/2 that you cannot do on DOS?," the Hamilton C shell is one answer. This is an application that simply could not be built on DOS.

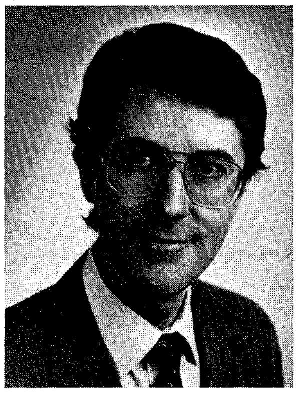
Hamilton C shell is an interactive programming language for OS/2. It is a language for talking about files and processes or threads and the connections between them. The entire Berkeley C shell language popular on engineering workstations has been faithfully recreated and carefully updated with modern compiler technology to produce a very fast, powerful tool.

Running in a Presentation Manager window or full-screen, Hamilton C shell is a quantum leap from the standard OS/2 command processor in performance, functionality, and ease of use. People who spend most of their day in front of OS/2 may well save an hour by using Hamilton C shell; the payback for purchasing this product can be measured in days.

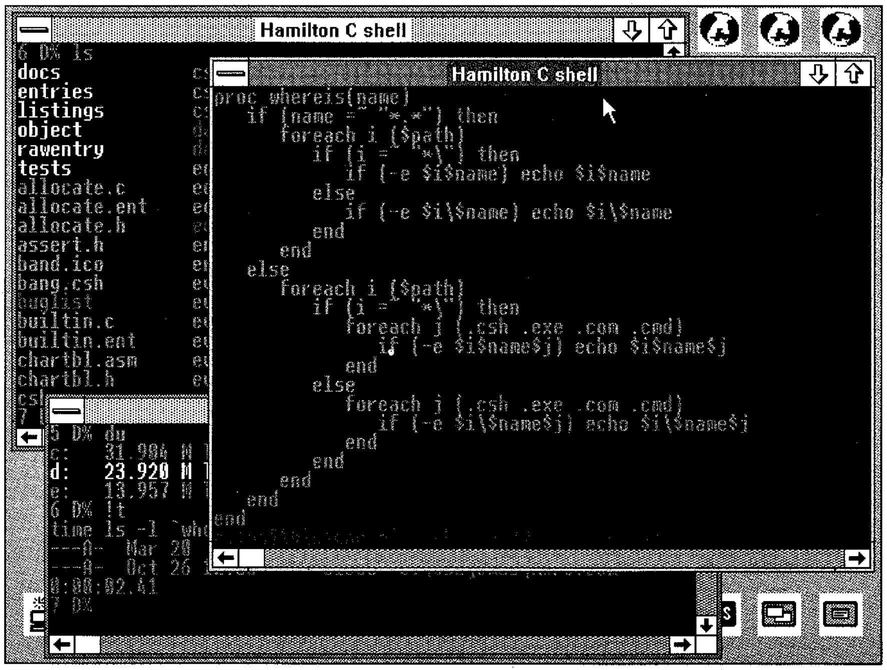
FEATURES

Shown in Figure 1, Hamilton C shell features a full range of programming constructs for iteration, condition testing, and expression evaluation. Statements can be nested—"for" loops inside "switch" statements inside "if" statements, etc.—to any desired depth. An individual statement can have up to 64 KB of command line arguments. Expressions can involve integers, strings, or floating point values and always result in sensible types, considering both the types and the values of the operands.

In addition to piping and I/O redirection, command substitution is provided, which allows the output of one command to be used as command-line arguments to another. Shell scripts, aliases, and procedures allow user-defined language extensions. Filename wildcarding is provided by a powerful recursive pattern-match algorithm; a wildcard can specify matches against several directory levels in a path and a sensible match is guaranteed no matter how complex the pattern. The history mechanism allows for recalling and editing past commands.



Douglas A. Hamilton



```
6 D% ls
docs
entries
listings
object
rawentry
tests
allocate.c
allocate.ent
allocate.h
assert.h
band.ico
bang.csh
buglist
builtin.c
builtin.ent
chartbl.asm
chartbl.h
```

```
proc whereis(name)
   if (name =~ "*.*") then
      foreach i ($path)
         if (i =~ "*\") then
            if (-e $i$name) echo $i$name
         else
            if (-e $i\$name) echo $i\$name
         end
      end
   else
      foreach i ($path)
         if (i =~ "*\") then
            foreach j (.csh .exe .com .cmd)
               if (-e $i$name$j) echo $i$name$j
            end
         else
            foreach j (.csh .exe .com .cmd)
               if (-e $i\$name$j) echo $i\$name$j
            end
         end
      end
   end
end
```

```
5 D% du
c:     31.984 M
d:     23.920 M
e:     13.957 M
6 D% !t
time ls -l `wh
---A-  Mar 20
---A-  Oct 26
0:00:02.41
7 D%
```

Figure 1. Hamilton C Shell

Hamilton C shell is a completely new implementation of the C shell language for OS/2 comprised of over 40,000 lines of C.

Finally, a large set of utilities is provided to round out the vocabulary of the language. There are tools for manipulating files or directories as objects, displaying disk usage, browsing forward and backward through a file or through the data coming through a pipe, etc.

Is it a better batch language? Yes, but it is much more than that. Hamilton C shell can be used to build script files to do many things. Scripts are also easier to get working and their performance is dramatically better. More importantly, the regularity of the language, the lifting of restrictions on command line length, the ability to edit and rerun a command, the powerful built-in constructs, etc., make it possible to do many more things interactively. It is much easier to get things right the first time. A problem has to be far more complex before the edit-debug-edit cycle we associate with getting a batch file to work is needed.

A good shell provides interaction but tackles a different problem than icons and windows. Instead of the point-and-shoot immediacy of "do this single thing now," a shell offers language and the ability to describe more customized or repetitive actions (for example, identify a set of files, perform some action against them, and filter the results in some interesting way).

BACKGROUND

The earliest command processors did little more than let the user type the name of the program to be run. In the late 1960s, things began to change, reflecting new developments in operating system and language design. The introduction of UNIX brought with it notions of filename wildcarding, I/O redirection, piping, and background execution. The notation we use today for these constructs comes from the earliest UNIX shell. The author of the first shell, Steven Bourne, claims to have borrowed the term "shell" from the earlier MULTICS system; certainly, it captured the idea of a user interface layer wrapped around the kernel of the operating system.

Overall, the Bourne shell reflected the times: structured programming was a new concept and in scholarly journals of the time it was fashionable to reverse the letters of a keyword opening a control structure to indicate the end of it. In the Bourne shell, an "if" statement ended with "fi," "case" with "esac," etc. Also, there were relatively few "creature comforts." For example, if you mistyped a command, you typed it over—there was nothing else you could do.

By the end of the 1970s, considerably more experience had been collected with high level languages in general and with the shell in particular. Seeing the need for a shell with a regular, modern style and grammar, and a history mechanism for recalling and editing past commands, Bill Joy (then at University of California at Berkeley) undertook the design of the Berkeley C shell. He called it the C shell to point out its similarity in control structures and arithmetic expression operators to the popular C language; realistically, though, its structure will be familiar to users of any modern block-structured language including Pascal or PL/I.

DESIGNING FOR OS/2

Hamilton C shell is a completely new implementation of the C shell language for OS/2 comprised of over 40,000 lines of C. This is significantly larger than the roughly 10,000 lines in the original Berkeley C shell. We estimate that about half of the growth was due to our choice of more modern and more powerful, albeit more complex, compiler technology.

The rest of the growth was due to the special engineering considerations of OS/2. There were a few cases where functions that UNIX programmers take for granted just did not exist and had to be recreated at the application level. For example, UNIX systems are commonly supplied with a process status facility that reads tables inside the kernel to produce a list of active processes. Lacking a similar interface in OS/2, Hamilton C shell maintains a list of child activities that it creates; doing this requires the creation of dedicated "clean-up" threads whose sole purpose is to sleep in the background waiting for child activities to finish.

Generally speaking, we did not find OS/2 to be a particularly difficult system to program. What we did find is that OS/2 provides unique facilities that are too good to pass up even though putting them to best advantage does take more code. Multi-threading, which offers an extremely low-cost, high performance mechanism for spawning concurrent activities, was probably the most exciting. In contrast to a process, which has its own address space and resources (current directories, file descriptors, etc.), a thread owns only a stack and a register set; a large number of threads, each separately scheduled, can share the memory space and resources of a single process. Because of their extremely low overhead, threads can be spawned very quickly. Since they share the same memory space, they also can share information with greater facility.

In the Berkeley C shell, if the user requested an activity be performed in the background or concurrent with other activities, separate processes were needed. If a background activity performed a calculation, it was awkward to retrieve the result in the foreground. In Hamilton C shell, threads are generally used instead, although obviously if an external application is invoked, that must be done with a separate process or screen group. The result is far better performance. If a background thread assigns a value to a variable or creates a new procedure definition, the results are available instantaneously to any other thread.

Of course, threads are not free: anywhere a thread needs to "own" a resource normally shared with other threads, the designer is left to his own devices to build an appropriate mechanism. For example, critical sections that examine or modify shared objects must be protected with semaphores. In a shell, it would hardly be acceptable if a script running quietly in the background could suddenly, without warning, change the foreground current directory! Building a high-performance mechanism to recreate a current directory notion for each thread turned out to be a fairly challenging project.

On the whole, OS/2's superior kernel services were a big win; we expect that on similar hardware platforms, Hamilton C shell should easily outperform any UNIX shell.

SUMMARY

Hamilton C shell has been shipping since December 12, 1988. It is available for immediate delivery for $350 and comes with an unconditional satisfaction guarantee. Anyone with a need to quickly increase their productivity on OS/2 will likely find this to be a very worthwhile acquisition.

Douglas A. Hamilton, *Hamilton Laboratories, 13 Old Farm Road, Wayland, MA 01778-3117. (508) 358-5715. Mr. Hamilton is the author of the Hamilton C shell. Prior to founding Hamilton Laboratories in September, 1987, he was the software development manager for Prime Computer's RISC-based engineering workstations, jointly developed with MIPS Computer Systems and Silicon Graphics. He holds bachelor's and master's degrees in Electrical Engineering from Stanford University and an MBA from Boston University. His background includes eight years at IBM and six years at Prime Computer in various engineering design and management positions.*

Hamilton C shell has been shipping since December 12, 1988. It is available for immediate delivery for $350.