Essential Forth

In most Operating Systems and programming languages a file system is below any database. In Forth a database is below any file system.

Over the years I have worked with lots of databases, databases on mainframes, minis, micros, and embedded computers. I've seen them evolve from flat files with fixed records to relational and distributed databases. I've seen the evolution of user interfaces and development tools. I've built them in lots of languages on different machines under different OS and using different tools. I've done the whole range and found 4GL and 'C' to be a nice mix when I used it. But I can see how Forth has always had an advantage in this arena.

When I first encountered Forth it was being widely used as both a development language and Operating System. One of the things that I liked about it was that as an OS and Integrated Development Environment it was smaller than many alternative Operating Systems without any application code or data. This left more room in memory and mass storage for application code and data. So programs that did the same thing as their non-Forth counterparts were not only more compact but there was more room to allow programs to be larger and more powerful.

The inventor of Forth, Charles Moore, said that Forth was invented to avoid the unsolvable problems in computer science, like the problems related to files. **Files are handled differently in different Operating Systems or even on different versions of the same Operating System. So are directories, so are filenames, so are permissions, so are low level access mechanisms, and so is file caching, even character formats.** All create compilications, problems, and require a great deal of effort for those spend their careers dealing with these same unsolvable problems again and again over the years. You solve the problems it creates for one application today, but tomorrow you have to try to solve it again in a different context. [Moore, 1999]

Early Forth was designed to free the programmer from the black boxes and restrictions that foreign Operating Systems and other languages imposed on programmers. They were designed to provide the highest performance for difficult problems and provide high quality, reliability, and maintainability. The nature of Forth is that it resembles the fourth generation database languages that we learned in the 80s. When it is time to define the scheme of our database data we just extend our language. We build our databases up from the dictionary and the memory and virtual memory managment words. They are as simple and early bound as possible but no more. [Moore, 1970] [Moore, 2001]

What is a file system? It is a very lazy form of database. Any database built on top of it will have a lazy foundation unless the OS has a file system fine tuned for streaming files and maximum performance on file and record access and caching in all cases. In reality most users have highly fragmented non-linear files and don't have the file system optimized for their important applications.

This is probably one of the greatest differences between Forth and C. C was designed to write an OS in which the concept is that everything is a file. Databases are built using the file system and C structures as the containers. In Forth we have the virtual memory system and dictionary

instead, the concept of extensible database below file system level.

Now it is often argued that one can do the same thing as Forth databases using memory mapped files on conventional Operating Systems etc. And if those systems are as small and flexible as Forth systems, and provide fast boot times and provide direct mapped access to the mass storage and an extensible dictionary and code that if factored to pass arguments on a stack then well maybe, sort of, but that's Forth!

Those who rely on extenal OS to provide the database containers for them often end up not only with files that take orders of magnitude longer to access, but fragmented files that take further orders of magnitude to process, and sometimes complications like the poor file interfaces in some famous Forths that combine with things like virus protection software to slow things down and complicate them by futher orders of magnitude. I have seen that in the workplace.

We have the examples of people just pulling code from libraries and using it as most C programmers will tend to do by reflex. The examples of Forth database code are attrocious. They bypass all the traditional things that Forth always did to beat other products on database software. They are pale copies of primitive file based flat databases using C style structures with runtime address resolution, terrible examples!

Chuck is too kind to say that the people doing this will do 100 times as much work as he and will write 100 times more code to do roughtly the same thing. But it will only be roughly the same thing. At the heart of Chuck early programs for the Smithsonian Astrophysical Observatory or the National Radio Astronomy Observatory at Kit Peak, or many of the early Forth system were databases. [Moore, 2001]

Forth Inc. got hundreds of users updating database records on the same computer in the old days on machines that were considered too small to host a modern Operating System. Because of nature of Forth they could host 300 users and a high performance database on machines when other people could only run toy Operating Systems designed for one user. Forth from those days demonstrates some important concepts such as:

A file system is really a database application.
A metacompiler is really a database application.
CAD is really a database application.
A high-performance database is not limited by a file system.

When recently asked in the comp.lang.forth newsgroup on usenet **"What did Forth fix?"** I half-jokingly said that other programming languages and Operating Systems were too big and too slow to write decent database systems. That appears to be the legacy of Forth. Other people said that other systems were black boxes. Chuck Moore, the inventor of Forth, replied that, **"Forth solved the mega-software problem."** That's the short version. Of course when asked, **"How would you implement C style structures which are essential to my work?"** Chuck's reply was his famous, **"I wouldn't."** [Moore, 2002]

That's because that's not how Forth works. Chuck would manually optimize everything to the maximum while more typically a C programmer would abstract the problem and hope that the

compiler will optimize the code. If it uses files it cannot optimize the way Chuck would.

Chuck and I have gone over individual details before, how designing the database scheme for the application takes most of the time because it is all about making it optimal for the most frequently executed code. When done right most of the C style structure code optimizes away completely in the most common cases and allmost all the file code optimizes away. As Chuck said when talking about file code in somone else's program, **"This is an opportunity to remove 100% of the code from this section of your program."** The way Chuck does these things today the only cost is the inclusion of auto-increment instructions in the instruction set at the hardware level.

The Forth style is to build the highest performance foundation as close to the metal as possible and then put the Forth database right on that.

There is a whole generation who have a hard time separating the ideas pervasive in the C/UNIX world, where one of the most basic ideas is that **'everything is a file'**, from the essence of many computing problems. Many people cannot separate the Microsoft software from the Intel hardware from the essence of many computing problems. A danger of this trend is that this will result in not only file abuse, but total dependence on standard file systems or worse yet on proprietary file or OS standards. It also carries the danger that they may miss the entire idea behind alternative languages, Operating Systems, and hardware that may make better use of programmer and computer time.

GNU stands for GNU isn't UNIX. People sometimes use the term GNU as Gnu is New UNIX. But Forth, when it provides OS services in Forth, is definately not UNIX. **As an Operating System it is just there to support Forth and Forth is just there to support a solution. It is a language and Operating System based on words, stacks, and database blocks. In a picture of a problem, a computer, and solution, Forth is part of the solution, not part of the problem. It imposes no formal requirements or standards on the solution, it exists as part of a well factored solution.** [Moore, 1999]

Jeff Fox
7/16/02

References:

[Moore, 1999] Moore, Charles H., 1x Forth, Charles Moore, Interview 4/13/99

[Moore, 1970] Moore, Charles H. and Leach, Geoffrey C., FORTH - A Language for Interactive Computing (pdf) (html), Amsterdam NY: Mohasco Industries, Inc. (internal pub.) 1970.

[Moore, 2001] Moore, Charles H., colorforth.com, 2001

[Moore, 2002] Moore, Charles H., Internet Chat, May 5, 2002

[Moore, 1999] Moore, Charles H., Dispelling the User Illusion, SVFIG, 5/22/99