Forth versus C Howerd Oakford

http://www.inventio.co.uk/forthvsc.htm

Some IAQs (Infrequently Asked Questions) about Forth and C.

Q. What is Forth?

A. Forth is a computer programming language which has a return stack, an explicit parameter stack and a dictionary. It features an incremental compiler, an interpreter and a very fast edit-compile-test cycle. Forth is extensible.

Q. What is C?

A. C is a computer language which has a single, implicit stack. It features a “tool set” that includes a batch-mode pre-processor, compiler, linker and makefile program. It also features a binary (pre-compiled) distribution system using libraries and header files. C is not extensible.

Q. Why is Forth not as widely used as C?

A. C is much better suited to most business environments.

Q. So why use Forth?

A. Because it is possible to write better programs in Forth than in C.

Q. Could you define “better” in this context?

A. Easier to read and maintain, fewer bugs, smaller, faster development.

Q. So why doesn’t everyone use Forth?

A. Because it is also possible to write worse programs in Forth than in C.

Q. Perhaps you should also define “worse” …

A. Impossible to read and maintain, more bugs, bigger, longer development.

Q. Why is this?

A. Because Forth does not impose any restrictions on what you do or how you do it. However large your C program gets the source for it will still be C. Since Forth is extensible, you can extend the language so that part of your program can be written in a superset of Forth, specifically tailored to the application.

Q. Doesn’t this mean that you have to learn what is essentially a new language for every application?

A. Yes.

Q. And this doesn’t worry you?

A. If the "new language" maps exactly onto the problem domain, what you are learning about is the problem, and then describing the solution becomes trivial.

Q. What about if the "new language" does not map to the problem domain?

A. You are in big trouble! The worst programs are those where an entirely new language is created independently of the problem, and then the problem is solved using this new language. You then really do have to learn a new language <u>and</u> the problem domain. This is actually much worse than programming in C, where at least you already know the language.

Q. How do you make sure that your Forth programs map to the problem domain?

A. Feedback. Ask the people who understand the problem domain if the program is right. Non-programmers can spot deviations from reality better than programmers, who usually immerse themselves in too much detail.

Q. But how can a non-programmer even read a program, let alone tell whether it is right?

A. Forth syntax can be learnt in an hour or two. If the top level of the program cannot be read and understood by someone who understands the problem domain, it is wrong and you should change it.

Q. How much faster can you develop programs in Forth compared to C?

A. This depends on the type of application and what is already available. Almost all programs are not so much written as "ported" from similar programs. It is very unusual to see the same application written in both Forth and C, so comparison of development times must be anecdotal.

For embedded, non-multitasking applications I would put the ratio of Forth:C development times at around 1:2 to 1:3. For embedded multitasking applications at between 1:4 to 1:6, and for PC based Windows applications about 1:4. Others have suggested figures of 1:10 or even higher. I think 1:pi is about right.

Q. You are claiming that you can finish a program three times more quickly in Forth than C. How can this be true?

A. The difference is not only in the language syntax - you can convert Forth code into C and back relatively easily. It is possible to develop code in a C-like way in Forth, but it is <u>not</u> possible to develop code in a Forth-like way using the C tool set.

Q. What is it about the C tool set that stops you programming in a Forth-like way?

A1. The C language and tool set operate in a batch mode. The time taken to edit source, build and download is not usually less than a number of seconds.

Because Forth compiles incrementally and has an interpreter and explicit stack, changes can be tested immediately.

The turn-round speed is actually more important than most people realise.

A2. The lack of an interpreter in C means that code must be tested using "printf", a source level ICE/debugger or a special test harness or Command Line Interpreter. All of these techniques add to the complexity of the system.

Forth has an interpreter, and this combined with the explicit stack means that you can enter arguments and test any function at any time.

A3. Forth is extensible, so you can create a superset of Forth which describes your problem domain exactly.

Q. I have seen versions of Forth written in C. Doesn't this mean that you can get all of the advantages of Forth, but written in C?

A. No. The real power of Forth lies in the synergy of the entire system. You can certainly gain an interpreter, but at the expense of a duplicate set of tools, two file formats, and two languages to learn before you even start on the application.

Q. Does Forth have any known weaknesses?

A. Yes. There is no support within the Forth language for version control. C provides pre-compiled libraries and header files which allow vendors to supply their code in a form that cannot be modified by the customer. This is a simple but effective version control system. Many Forths are supplied with a similar binary module system but this is an extension to the language.

Q. Does Forth have any perceived weaknesses?

A1. Yes. Because Forth is so simple, many people have written their own.

In fact it is almost mandatory that computer science students spend an afternoon writing a Forth, just to see how easy it is. Not surprisingly, not all Forths are created equal.

The finely honed products available from Forth, Inc., MPE Ltd and other commercial vendors are vastly better than most of the free Forths available on the WWW. Specifically, with advances in native compiler technology, Forth is no longer slow.

A2. Forth is sometimes accused of being cryptic or "read only" code. This is because there is a lot of cryptic, "read only" code written in Forth. If you are going to use Forth, you must control its use, and ensure that your code is easy to read. This is a project management issue.

A3. Forth uses "Reverse Polish" notation. This is because the problem domain that the Forth system is addressing is "how to tell the computer what to do". The simplest way to solve this problem is to give the computer its instructions in the order that it needs them. The code "1 2 +" tells the computer to put a 1 on the stack, put a 2 on the stack and add them.

The fact that the rest of the world (HP calculators excepted) write "1 + 2" is because this shows the operator and its arguments in a graphical way, as written on a chalkboard. It is easy to write a Forth parser that converts "1 + 2" into "1 2 +", but this would put a boundary onto the Forth system that would make further extensions to the language much more difficult.

Reverse Polish is seen as antiquated, and the fact that the computer has not even been taught to understand common usage is interpreted as a major weakness. In fact, it is just that Forth is not attempting to solve an entirely different problem.

A4. Forth is perceived as a "dead" language. This is because it is not being championed by big companies, nor is it taught as the de-facto standard in most computer science courses.

In fact, Forth is alive and well in many low-profile niche markets, usually where resources are limited, software must be reliable or there is a need for an interpreted “scripting language”.

One limited resource is often money - short development times mean reduced costs. Smart cards and space environments are examples of restricted hardware where Forth flourishes.

Q. Why has Forth been described as “a polarising language”?

A1. People either love Forth or hate it. It is easy not to like Forth. Programmers don’t like Forth because it is not fashionable (at the time of writing) and does not look good on a CV. Project managers don’t like Forth because they can’t get Forth programmers easily, and the quality of code depends critically on the programmer.

University lecturers don’t like it because there is not enough that can be taught about Forth that can be examined. Software vendors don’t like Forth because it removes the complexity they need to survive. Users don’t like Forth because they have never heard of it.

A2. People either love Forth or hate it. It is easy to like Forth. Programmers like Forth because its fun – you remove the tedious 66% of the job.

Project managers like Forth because they can get the project finished on time, with fewer, easier-to-find bugs.

University lecturers like Forth because they can teach their students about the art of programming and be involved in the cutting edge of computer science.

Software vendors like Forth because they get involved with their customers to help them solve their problems and refine their products at the same time.

Users like Forth because it makes programs that work.

Q. So you like Forth then?

A. Yes.

Q. What about C?

A. I have nothing against C – it is a language specifically tailored to the problem domain of “writing C based operating systems”. Unix/Linux and DOS/Windows are written in C and are immensely successful.

“If your only tool is a hammer, all problems become nails”. For those applications that are not operating systems, Forth is probably a better language to use than C.

Q. Is that it?

A. Yes!

Howerd Oakford