Real Programmers Don't Use Pascal

Back in the Golden Era of computers, it was easy to separate the men from the boys (sometimes called "Real Men" and "Quiche Eaters," respectively). During this period, the Real Men were the ones who understood computer programming, and the Quiche Eaters were the ones who didn't. A real computer programmer said things like "DO 10 I=1,10" and "ABEND," and the rest of the world said things like "computers are too complicated for me" and "I can't relate to computers—they're so impersonal." A previous work, B. Feirstein's *Real Men Don't Eat Quiche*, a 1982 Pocket Books publication, points out that Real Men don't "relate" to anything and aren't afraid of being impersonal.

But, times change. Today, we are faced with a world in which little old ladies can get computerized microwave ovens, 12-year-old kids can blow Real Men out of the water playing Asteroids and Pac-Man, and anyone can buy and understand his very own personal computer. The Real Programmer is in danger of being replaced by high school students with TRASH-80s!

There are, however, differences between the typical high school junior Pac-Man player and a Real Programmer. Knowing these differences may give kids something to aspire to—a role model, a father figure. It will also help keep Real Programmers employed.

The easiest way to determine who the Real Programmers are is by the programming language they use. Real Programmers use FORTRAN. Quiche Eaters use Pascal. Niklaus Wirth, the designer of Pascal, was once asked, "How do you pronounce your name?" "You can either call me by name, pronouncing it 'Veert,' or call me by value, 'Worth,' " he replied. One can tell immediately from this comment that Niklaus Wirth is a Quiche Eater. The only parameter passing mechanism endorsed by Real Programmers is call-by-value-return, as implemented in the IBM/370 FORTRAN G and H compilers. Real Programmers don't need abstract concepts to get their jobs done; they are perfectly happy with a keypunch, a FORTRAN IV compiler, and a beer. Real Programmers do list processing, string manipulation, accounting (if they do it at all), and artificial intelligence programs in FORTRAN.

If you can't do it in FORTRAN, do it in assembly language. If you can't do it in assembly language, it isn't worth doing.

Computer science academicians have gotten into a structured programming rut during the past few years. They claim that programs are more easily understood if some special language constructs and techniques are used. They don't all agree on exactly which constructs, of course, and the examples they use to show their particular point of view invariably fit on a single page of some obscure journal. When I got out of school,

I thought I was the best programmer in the world. I could write an unbeatable tic-tac-toe program, use five different computer languages, and create 1,000-line programs that worked. Then I got out into the real world. My first task was to read and understand a 200,000-line FORTRAN program, then speed it up by a factor of two. Any Real Programmer will tell you that all the structured coding in the world won't help you solve a problem like that—it takes actual talent. Some observations on Real Programmers and structured programming:

- Real Programmers aren't afraid to use GOTOs.
- Real Programmers can write five-page-long DO loops without getting confused.
- Real Programmers like arithmetic IF statements because they make the code more interesting.
- Real Programmers write self-modifying code, especially if it saves them 20 nanoseconds in the middle of a tight loop.
- Real Programmers don't need comments; the code is obvious.
- Since FORTRAN doesn't have a structured IF, REPEAT . . . UNTIL, or CASE statement, Real Programmers don't have to worry about not using them. Besides, they can be simulated when necessary using assigned GOTOs.

Data structures have also been in the press lately. Abstract data types, structures, pointers, lists, and strings have become popular in certain circles. Wirth, the Quiche Eater, actually wrote an entire book (*Algorithms + Data Structures = Programs*, Prentice Hall, 1976) that said you could write a program based on data structures, instead of the other way around. As all Real Programmers know, the only useful data structure is the array. Strings, lists, structures, and sets are all special cases of arrays and can be treated as such without complicating your programming language. The worst thing about fancy data types is that you have to declare them, and real programming languages, as we all know, have implicit typing based on the first letter of the six-character variable name.

What kind of operating system is used by a Real Programmer? CP/M? God forbid. After all, it is basically a toy operating system. Even little old ladies and grade school students can use and understand CP/M.

Unix is a lot more complicated of course—the typical Unix hacker never can remember what the PRINT command is called this week—but when it gets right down to it, Unix is a glorified video game. People don't do serious work on Unix systems; they send jokes around the world on USENET or write adventure games and research papers.

No, the Real Programmer uses OS/370. A good programmer can find and understand the description of the IJK305I error he just got in his JCL manual. The great programmer can write JCL without referring to the manual at all. A truly outstanding programmer can find bugs buried in a six-megabyte core dump without using a hex calculator.

OS/370 is a truly remarkable operating system. It's possible to destroy several days' worth of work with a single misplaced space, so alertness in the programming staff is encouraged. The best way to approach the system is through a keypunch. Some people claim there is a timesharing system that runs on OS/370, but after careful study I have come to the conclusion that they are mistaken.

What kind of tools does a Real Programmer use? In theory, a Real Programmer could run his programs by keying them into the front panel of the computer. In the early days, when computers had front panels, this was occasionally done. Your typical Real Programmer knew the entire bootstrap loader by memory in hex, and toggled it in whenever it got destroyed by his program. Back then, memory was memory—it didn't go away when the power went off. Today, memory either forgets things when you don't want it to, or remembers things long after they should be forgotten. Legend has it that Seymour Cray, inventor of the CRAY I supercomputer and most of Control Data's computers, toggled in the first operating system for the CDC 7600 on the front panel from memory when it was first powered on. Cray, of course, is a Real Programmer.

One of my favorite Real Programmers was a systems programmer for Texas Instruments. One day, he got a long distance call from a user whose system had crashed in the middle of some important work. Jim repaired the damage over the phone, getting the user to toggle in disk I/O instructions at the front panel, repairing system tables in hex, and reading register contents back over the phone. The moral of this story: while a Real Programmer usually includes a keypunch and line printer in his tool kit, he can get along with just a front panel and a telephone in emergencies.

In some companies, text editing no longer consists of IO engineers standing in line to use an 029 keypunch. In fact, the building I work in doesn't contain a single keypunch. The Real Programmer in this situation has to do his work with a text editor program. Most systems supply several text editors to select from, and the Real Programmer must be careful to pick one that reflects his personal style. Many people believe that the best text editors in the world were written at Xerox Palo Alto Research Center for use on Alto and Dorado computers. Unfortunately, no Real Programmer would ever use a computer with an operating system called Smalltalk, and would certainly not talk to the computer with a mouse.

Some of the concepts in these Xerox editors have been incorporated into editors running on more reasonably named operating systems, editors such as EMACS and VI. The problem with these editors is that Real Programmers consider "what you see is what you get" a bad concept in text editors. The Real Programmer wants a "you asked for it, you got it" text editor; one that is complicated, cryptic, powerful, unforgiving, and dangerous. TECO, to be precise.

It has been observed that a TECO command sequence more closely resembles transmission line noise than readable text. One of the more entertaining games to play with TECO is to type your name in as a command line and try to guess what it does. Just

about any possible typing error while communicating with TECO will probably destroy your program, or even worse, introduce subtle and mysterious bugs in a once-working subroutine.

For this reason, Real Programmers are reluctant to actually edit a program that is close to working. They find it much easier to patch the binary object code directly, using a wonderful program called SUPERZAP (or its equivalent on non-IBM machines). This works so well that many programs running on IBM systems bear no relation to the original FORTRAN code. In a number of cases, the original source code is no longer available. When it comes time to fix a program like this, no manager would even think of sending anyone less than a Real Programmer to do the job—no quiche-eating structured programmer would even know where to start. This is called job security.

Some programming tools not used by Real Programmers include:

- FORTRAN preprocessors like MORTRAN and RATFOR. These Cuisinarts of programming are great for making quiche.
- Source language debuggers. Real Programmers can read core dumps.
- Compilers with array bounds checking. They stifle creativity, destroy most of the interesting uses for EQUIVALENCE, and make it impossible to modify the operating system code with negative subscripts. Worst of all, bounds checking is inefficient.
- Source code maintenance systems. A Real Programmer keeps his code locked in a card file, because it implies that the owner cannot leave his important programs unguarded.

Where does the typical Real Programmer work? What kind of programs are worthy of such talented individuals? You can be sure that no Real Programmer would be caught dead writing accounts-receivable programs in COBOL, or sorting mailing lists for *People* magazine. A Real Programmer wants tasks of earth-shaking importance.

Real Programmers work for Los Alamos National Laboratory, writing atomic bomb simulations to run on CRAY I supercomputers. They also work for the National Security Agency, decoding Russian transmissions.

It was largely due to the efforts of thousands of Real Programmers working for NASA that our boys got to the moon and back before the cosmonauts. Computers in the Space Shuttle were programmed by Real Programmers, and these true professionals are at work for Boeing, designing operating systems for cruise missiles.

Some of the most awesome Real Programmers work at the Jet Propulsion Laboratory in California. Many of them know the entire operating system of the Pioneer and Voyager spacecraft by heart. With a combination of large ground-based FORTRAN programs and small spacecraft-based assembly language programs, they can do incredible

feats of navigation and improvisation—such as hitting 10-kilometer-wide windows at Saturn after six years in space, and repairing or bypassing damaged sensor platforms, radios, and batteries. Allegedly, one Real Programmer managed to tuck a pattern-matching program into a few hundred bytes of unused memory in a Voyager spacecraft that searched for, located, and photographed a new moon of Jupiter.

One plan for the Galileo spacecraft is to use a gravity-assist trajectory past Mars on the way to Jupiter. This trajectory passes within 80 ± 3 kilometers of the surface of Mars. Nobody is going to trust a Pascal program or programmer for this kind of navigation.

Many of the world's Real Programmers work for the U. S. Government, mainly in the Defense Department. This is as it should be. Recently, however, a black cloud has formed on the Real Programmer horizon. It seems that some highly placed Quiche Eaters at the Defense Department decided that all Defense programs should be written in some grand unified language called Ada. For a while, it seemed that Ada was destined to become a language that went against all the precepts of Real Programming. It is a language with structure, data types, strong typing, and semicolons. In short, it's designed to cripple the creativity of the typical Real Programmer. Fortunately, the language adopted by DoD has enough interesting features to make it approachable—it's incredibly complex, includes methods for messing with the operating system and rearranging memory, and Edsger Dijkstra doesn't like it. Dijkstra, as you should know, authored "GOTOs Considered Harmful," a landmark work in programming methodology applauded by Pascal programmers and Quiche Eaters alike. Besides, the determined Real Programmer can write FORTRAN programs in any language.

The Real Programmer might compromise his principles and work on something slightly more trivial than the destruction of life, providing there's enough money in it. There are several Real Programmers building video games at Atari, for example. But they don't play the games. A Real Programmer knows how to beat the machine every time and there's no challenge in that. Everyone working at LucasFilm is a Real Programmer because it would be crazy to turn down the money of 50 million *Star Wars* fans. The proportion of Real Programmers in computer graphics is somewhat lower than the norm, mostly because nobody has found a use for computer graphics yet. On the other hand, all computer graphics is done in FORTRAN, so there are some people doing graphics to avoid writing COBOL programs.

Generally, the Real Programmer plays the same way he works—with computers. He is constantly amazed that his employer actually pays him to do what he would be doing for fun anyway, although he is careful not to express this opinion out loud. Occasionally, the Real Programmer does step out of the office for a breath of fresh air and a beer or two. Here are some tips on recognizing Real Programmers away from the computer room:

- At a party, the Real Programmers are the ones in the corner talking about operating system security and how to get around it.

- At a football game, the Real Programmer is the one comparing the plays against his simulations printed on 11-by-14 fanfold paper.
- At the beach, the Real Programmer is the one drawing flowcharts in the sand.
- A Real Programmer goes to a disco to watch the light show.
- At a funeral, the Real Programmer is the one saying "Poor George. And he almost had the sort routine working before the coronary."
- In a grocery store, the Real Programmer is the one who insists on running the cans past the laser checkout scanner himself, because he never could trust keypunch operators to get it right the first time.

What sort of environment does the Real Programmer function best in? This is an important question for the managers of Real Programmers. Considering the amount of money it costs to keep one on the staff, it's best to put him or her in optimal environment.

The typical Real Programmer lives in front of a computer terminal. Surrounding this terminal are the listings of every program he has ever worked on. These are piled in roughly chronological order on every flat surface in the office. You will also find some half-dozen or so partly filled cups of cold coffee. Occasionally, there will be cigarette butts floating in the coffee. In some cases, the cups will contain Orange Crush. And, unless he is very good, there will be copies of the OS JCL manual and the Principles of Operation open to some particularly interesting pages. Taped to the wall is a line-printer Snoopy calendar for the year 1969. Strewn about the floor there will be several wrappers for peanut butter-filled cheese bars (the type that are made stale at the bakery so they can't get any worse while waiting in the vending machine). Finally, in the top left-hand desk drawer, underneath the box of Oreos, is a flowcharting template, left there by the previous occupant. Real Programmers write programs, not documentation, which is left to the maintenance people.

The Real Programmer is capable of working 30, 40, even 50 hours at a stretch, under intense pressure. In fact, he prefers it that way. Bad response time doesn't bother the Real Programmer; it gives him a chance to catch a little sleep between compiles. If there is not enough schedule pressure on the Real Programmer, he tends to make things more challenging by working on some small but interesting part of the problem for the first nine weeks. Then he finishes the task in the last week, in two or three 50-hour marathons. This not only impresses his manager, but creates a convenient excuse for not doing the documentation. In general: no Real Programmer works 9 to 5, except those on the night shift. Real Programmers don't wear neckties. Real Programmers don't wear high-heeled shoes. Real Programmers arrive at work in time for lunch. A Real Programmer may or may not know his spouse's name. He does, however, know the entire ASCII (or EBCDIC) code table. Real Programmers don't know how to cook. Grocery stores aren't often open at 3 a. m., so they must survive on Twinkies and coffee.

Looking to the future, some Real Programmers are concerned that the latest generation of programmers are not brought up with the same outlook on life as their elders. Many of them have never seen a computer with a front panel. Hardly anyone graduating from school these days can do hex arithmetic without a calculator. Today's college graduates are soft—protected from the realities of programming by source level debuggers, text editors that count parentheses, and user-friendly operating systems. Worst of all, some of these alleged computer scientists manage to get degrees without ever learning FORTRAN! Are we destined to become an industry of Unix hackers and Pascal programmers?

From my experience, I think it's safe to report that the future is bright for Real Programmers. Neither OS/370 nor FORTRAN shows any signs of dying out, despite the efforts of Pascal programmers. Even more subtle tricks, like adding structured coding constructs to FORTRAN, have failed. Oh sure, some computer vendors have come out with FORTRAN 77 compilers, but every one of them has a way of converting itself back into a FORTRAN 66 compiler at the drop of an option card—to compile DO loops as God intended.

Even Unix might not be as bad on Real Programmers as it once was. The latest release of Unix has the potential of an operating system worthy of any Real Programmer. It has two different and subtly incompatible user interfaces, an arcane and complicated teletype driver, and virtual memory. If you ignore the fact that it's structured, even C programming can be appreciated by the Real Programmer. After all, there's no type checking, variable names are seven (10? Eight?) characters long, and the added bonus of the Pointer data type is thrown in. That's like having the best parts of FORTRAN and assembly language in one place, not to mention some of the more creative uses for #define.

No, the future isn't all that bad. Why, in the past few years, the popular press has even commented on the bright new crop of computer nerds and hackers leaving places like Stanford and MIT for the real world. From all evidence, the spirit of Real Programming lives on in these young men and women. As long as there are ill-defined goals, bizarre bugs, and unrealistic schedules, there will be Real Programmers willing to jump in and solve the problem, saving the documentation for later. Long live FORTRAN!

—Ed Post
Wilsonville, Oregon

(in: *Datamation*, July 1983, pp. 263–265)