REAL PROGRAMMERS Don't Use Pascal

By Ed Post

From DATAMATION, 1983

Back in the good old days - the "Golden Era" of computers, it was easy to separate the men from the boys (sometimes called "Real Men" and "Quiche Eaters" in the literature). During this period, the Real Men were the ones that understood computer programming, and the Quiche Eaters were the ones that didn't. A real computer programmer said things like "DO 10 I=1,10" and "ABEND" (they actually talked in capital letters, you understand), 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 [1] points out that Real MEn don't "relate" to anything, and aren't afraid of being impersonal.)

But, as usual, times change. We are faced today with a world in which little old ladies can get computers in their microwave ovens, 12-year -old kids can blow Real Men out of the water playing Asteroids and Pac-Man, and anyone can buy and even understand their very own Personal Computer. The REAL PROGRAMMER is in danger of becoming extinct, of being replaced by high-school students with TRASH-80's.

There is a clear need to point out the differences between the typical high-school junior Pac-Man player and a REAL PROGRAMMER. If this difference is made clear, it will give these kids something to aspire to - a role model, a Father Figure. It will also help explain to the employers of REAL PROGRAMMERs why it would be a mistake to replace the REAL PROGRAMMERs on their staff with 12-year-old Pac-Man players (at a considerable salary savings).

LANGUAGES

The easiest way to tell a REAL PROGRAMMER from the crowd is by the programming language he (or she) uses. REAL PROGRAMMERs use FORTRAN. Quiche Eaters use Pascal. Nicklaus Wirth, the designer of Pascal, gave a talk once at which he was asked "How do you pronounce your name?". He replied, "You can either call me by name, pronouncing it "Veert", or call me by value, 'Worth'." One can tell immediately from this comment that Nicklaus 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 all these 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 in FORTRAN**
- **REAL PROGRAMMERs do String Manipulation in FORTRAN**
- **REAL PROGRAMMERs do Accounting (if they do it at all) in FORTRAN**

- **REAL PROGRAMMERs do 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.

STRUCTURED PROGRAMMING

The academics in computer science have gotten into the "structured programming" rut over the past several years. They claim that programs are more easily understood if the programmer uses some special language constructs and techniques. They don't all agree on exactly what constructs, of course, and the examples they use to show their particular point of view invariably fit on a single page so some obscure journal or another - clearly not enough of an example to convince anyone. When I got out of school, I thought I was the best programmer in the world. I could write and unbeatable tic-tac-toe program, use five different computer languages, and create 1000-line programs that worked. (Really!)

Then I got out into the Real World. My first task in the Real World 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 quick observations on REAL PROGRAMMERs and Structured Programming:

- **REAL PROGRAMMERs aren't afraid to use GOTO's**
- **REAL PROGRAMMERs can write five-page-long DO loops without getting confused**
- **REAL PROGRAMMERs like arithmetic IF statements - they make the code more interesting**
- **Real programmers write self-modifying code, especially if they can save 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 GOTO's.**

Data Structures have also gotten a lot of press lately. Abstract data types, structures, pointers, lists, and strings have become popular in certain circles. Wirth (the above-mentioned quiche eater) actually wrote an entire book [2] contending that 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, sets - these are all special cases of arrays and can be treated that way just as easily without messing up your programming language with all sorts of complications. 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.

OPERATING SYSTEMS

What kind of operating system is used by a real programmer? CP/M? God forbid - CP/M, after all, is basically a toy operating system. Even little old ladies and grade school students can understand and use 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 UUCP-net and write adventure games and research papers.

No, your real programmer uses OS/370. A good programmer can find and understand that description of the IJK305I error he just got in his JCL manual. A great programmer can write JCL without referring to the manual at all. A truly outstanding programmer can find bugs buried in a 6 megabyte core dump without using a hex calculator. (I have actually seen this done.)

OS is a truly remarkable operating system. It’s possible to destroy days 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 time sharing system that runs on OS/370, but after careful study I have come to the conclusion that they were mistaken.

PROGRAMMING TOOLS

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. Back in the days when computers had front panels, this was actually done occasionally. 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’re better forgotten.) Legend has it that Seymore Cray, inventor of the Cray I supercomputer and most of control Data’s computers, actually toggled the first operating system for the CDC7600 in on the front panel from memory when it was first powered on. Seymore, needless to say 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 saving some important work. Jim was able to repair the damage over the phone, getting the user to toggle in disk I/O instructions at the front panel, repairing system tables in hex, 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 toolkit, he can get along with just a front panel and a telephone in emergencies.

In some companies, text editing no longer consists of ten engineers standing in line to use and 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 their Alto and Dorado computers [3]. Unfortunately, no real programmer would ever use a computer whose operating system is 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 - emacs and vi being two. The problem with these editors is that real programmers consider "what you see is what you get" to be just as bad a concept in text editors as it is in women. No the real programmer wants a "you asked for it, you got it" text editor - complicated, cryptic, powerful, unforgiving, dangerous. TECO, to be precise.

It has been observed that a TECO command sequence more closely resembles transmission line noise than readable text [4]. 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 talking 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 just 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 working programs on IBM systems bear no relation to the original FORTRAN code. In many 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 anything 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:

- **FORTRAN preprocessors like MORTRAN and RATFOR. The cuisinarts of programming - great for making quiche. See comments above on structured programming.**
- **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 up in a card file, because it implies that its owner cannot leave his important programs unguarded [5].**

THE REAL PROGRAMMER AT WORK

Where does the typical real programmer work? What kind of programs are worthy of the efforts of so talented an individual? 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 (literally!)

- **Real programmers work for Los Alamos National Laboratory, writing atomic bomb simulations to run on Cray I supercomputers.**
- **Real programmers 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 Russkies.**
- **Real programmers are at work for Boeing designing the operating systems for cruise missiles.**

Some of the most awesome real programmers of all 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 are able to do incredible feats of navigation and improvisation - hitting ten-kilometer wide windows at Saturn 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.

The current 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 a Pascal programmer) for navigation to these tolerances.

As you can tell, many of the world's real programmers work for the U.S. Government - mainly 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" (c, DoD). For a while, it seemed that ADA was destined to become a language that went against all the precepts of real programming - a language with structure, a language with data types, strong typing, and semicolons. In short, a language 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 Edsgar Dijkstra doesn't like it [6]. (Dijkstra, as I'm sure you know, was the author of "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 as we know it, providing there's enough money in it. There are several real programmers building video games at Atari, for example. (But not playing them - a real programmer knows how to beat the machine every time: no challenge in that.) Everyone working at LucasFilm is a real programmer. (It would be crazy to turn down the money of fifty million Star Trek fans.) The proportion of real programmers in computer graphics is somewhat lower than the norm, mostly because nobody has found a use for is done in FORTRAN, so there are a fair number of people doing graphics in order to avoid having to write COBOL programs.

THE REAL PROGRAMMER AT PLAY

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. 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 systems 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.**
- **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.**

THE REAL PROGRAMMERS NATURAL HABITAT

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 an environment where he can get his work done.

The typical real programmer lives in front of a computer terminal. Surrounding this terminal are:

- **Listings of all programs the real programmer has ever worked on, piled in roughly chronological order on every flat surface in the office.**
- **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.**
- **Unless, he is very good, there will be copies of the OS JCL manual and 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 are several wrappers for peanut butter filled cheese bars - the type that are made pre-stale at the bakery so they can't get any worse while waiting in the vending machine.**
- **Hiding in the top left-hand drawer of the desk is a stash of double-stuff Oreos for special occasions.**
- **Underneath the Oreos is a flowcharting template, left there by the previous occupant of the office. (Real programmers write programs, not documentation. Leave that 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 finishing the rest in the last week, in two or three 50-hour marathons. This not only impresses the hell out of his manager, who was despairing of ever getting the project done on time, but creates a convenient excuse for not doing the documentation. In general:

- **No real programmer works 9 to 5 (unless it's the ones at night)**
- **Real programmers don't wear neckties.**
- **Real programmers don't wear high-heeled shoes.**
- **Real programmers arrive at work in time for lunch [9]**
- **A real programmer might or might not know his wife'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 open at three in the morning. Real programmers survive on Twinkies and coffee.**

THE FUTURE

What of the future? It is a matter of some concern to real programmers that the latest generation of computer programmers are not being 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. College graduates these days 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 can only report that the future is bright for real programmers everywhere. Neither OS/370 nor FORTRAN show any signs of dying out, despite all the efforts of Pascal programmers the world over. Even more subtle tricks, like adding structured coding construct to FORTRAN have failed. Oh sure, some computer vendors have come out with the 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 like God meant them to be.

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 - two different and subtly incompatible user interfaces, an arcane and complicated teletype driver, 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 not type checking, variable names are seven (ten?, eight?) characters long, and the added bonus of the Pointer data type is thrown in - 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 ([7] and [8]) 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!

Acknowledgements

I would like to thank Jan E., Dave S., Rich G., Rich E., for their help in characterizing the real programmer, Heather B., for the illustration, Kathy E. for putting up with it, and atd!avsdS:mark for the initial inspiration.

References

[1] Feirstein, B., *Real Men Don't Eat Quiche*, Pocket Books, 1982.

[2] Wirth, N., *Algorithms + Data Structures = Programs*, Prenctice Hall, 1976.

[3] Ilson, R., "Recent Research in Text Processing", IEEE Trans. Prof. Comms, Vol PC-23, No., 4 Dec. 4, 1980.

[4]Finseth, C., "Theory and Practice of Text Editors - or - a Cookbook for an EMACS", B.S. Thesis, MIT, May 1980.

[5]Weinberg, G., *The Psychology of Computer Programming*, Van Nostrand Reinhold, 1971, p. 110.

[6] Dijkstra, E., "On the GREEN language submitted to the DoD", SIGPLAN Notices, Vol.3 NO.10, Oct. 1978.

[7] Rose, F., "Joy of Hacking", *Science* 82, Vol.3 No.9, 1982, p. 58-66.

[8] "The Hacker Papers", *Psychology Today*, Aug. 1980.

[9] sdcarl!lin, "REAL PROGRAMMERs", UUCP-net, Oct. 21, 1982.

Back to Net Humor, Best of DATAMATION, or the PlugIn DATAMATION Home Page