Not all it's cracked up to be

Fed up with trying to develop own Windows programs? Let Unix help / **Simson L. Garfinkel**

I DISCOVERED THE thrill of programming when I was 8 years old. I was taking courses at the Franklin Institute in Philadelphia when I discovered a book called "Basic Instruction, a Programmed Course" in the library. I read the book from cover to cover, then started writing computer programs on paper, tracing execution with my pencil and eraser.

It was three years before I had the chance to type one of the programs in to a computer to see if it would work. It didn't, of course, but with a little trial and error I eventually got the hang of it. I didn't know it, but I was well on my way to becoming a hacker – a person who writes code and explores computers for the sheer joy.

That was more than 20 years ago, but the thrill of programming has never worn off. There's just something intoxicating about being able to type a few commands into a computer, click a button, and then sit back and watch the computer do exactly what you told it to do.

It's the word "exactly" that is responsible for much of the thrill. As anyone who has ever wiped out an important word processor file knows, computers have this habit of doing exactly what you tell them. They can't help it, it's how they are built. Computers are deterministic devices. If you give them the same input, they will always produce the same output.

Programming, then, becomes an exercise in will and control.

You sit down with a blank screen and a few hours later (or a few months later) you have a completed program. If the program does something you like, it's because you put that particular feature there. If the program does something wrong, it's your fault. Any bug that's in your program is there because you put it there. And the worst part is knowing that if you work hard enough, and smart enough, and diligently enough, you can find all the bugs and eradicate them.

Gosh, how I wish the programmers at Microsoft felt that way!

I've been thinking about programming a lot in recent months. A few months ago I took the plunge and decided to learn how to write programs for Microsoft Windows. It's something that I had been putting off doing for a long, long time. The reason was elitism, really. I really honed my programming skills in the 1980s and early 1990s, working on a variety of Unix computers at the Massachusetts Institute of Technology. Like many of my geek friends, I've always looked down on PCs – and with good reason!

It's so much easier to write a program on a Unix workstation than it is to try to cobble together an application that runs on Windows 95.

The main difference between programming on Unix and programming in Windows is stability. Beginning programmers make a lot of mistakes. The problem is, when you run buggy programs on a PC, you risk crashing the entire operating system – or worse. A month ago, for example, a buggy program that I was working on wiped out my computer's hard drive. Fortunately, I had a recent backup. Microsoft's Windows NT operating system is a little better, but it's still possible to crash the entire computer with one bug.

Unix, on the other hand, is a very safe environment for learning to program. No matter how buggy your program, it's all but impossible for it to crash the rest of the computer. This makes it easier for beginners and experts alike, because you are not constantly waiting for your computer to reboot.

Unix is also a hacker's environment, because it is so easy to modify the tools with which you are working.

There are other differences as well. Most PC programmers use integrated development environments (IDEs) like Microsoft's Visual C, Visual Basic, or Symantec's Visual Cafe. An IDE is like an overgrown word processor with a built-in editor for creating your program and then a compiler for turning the program into running code. The problem with these systems is that the editors are based on a system created for editing English text, not programs. Frequently a lot of keystrokes or mouse-clicks are required to do fairly simple operations.

Most Unix programmers, on the other hand, use a separate editor and compiler. The advantage here is that it's easier to figure out exactly what the computer is doing when it compiles your program – knowledge that is critical to figuring out if your program is running correctly. For an editor, most programmers I know use EMACS, a powerful system that was developed by Richard Stallman at the MIT Artificial Intelligence Laboratory in 1974 – one of the greatest hackers ever.

After nearly 25 years of development, people have written extensions for EMACS to let people compile their programs, read their mail, browse the Web, and even partake in a computer-mediated psychotherapy session, all from within the program. In many ways, EMACS offer all of the advantages of those PC-integrated development environments without many of the problems

The gnu is the symbol of GNU Project, started by Richard Stallman at MIT to create free software. GNU stands for "GNU's Not Unix."

One program that tries to bridge the gap between EMACS and the PC world is Visual Slick Edit, by MicroEdge. For $295, this program has many of those standard EMACS features that I've come to love. For example, SlickEdit will automatically beautify your program – indenting and re-indenting it as necessary. SlickEdit also has some of the better PC IDE features, such as colorizing your program so that keywords and comments stand out, and giving instant access to the PC documentation. Find out more at www.slickedit.com

On the other hand, if you are stuck using Windows but would prefer programming the Unix way, you might go to the Internet and download the Unix programming tools – most of which have now been ported to Windows 95 and NT. You can download EMACS from ftp://ftp.cs.washington.edu/pub/ntemacs. To compile your programs you'll want the GNU-Win32 files from http://www.cygnus.com/misc/gnu-win32/.

Have fun, and happy hacking.

Technology writer Simson L. Garfinkel can be reached at plugged-in@simson.net.