The Beauty of Programming

by Linus Torvalds

I don't know how to really explain my fascination with programming, but I'll try. To somebody who does it, it's the most interesting thing in the world. It's a game much more involved than chess, a game where you can make up your own rules and where the end result is whatever you can make of it.

And yet, to the outside, it looks like the most boring thing on Earth.

Part of the initial excitement in programming is easy to explain: just the fact that when you tell the computer to do something, it will do it. Unerringly. Forever. Without a complaint.

And that's interesting in itself.

But blind obedience on its own, while initially fascinating, obviously does not make for a very likeable companion. What makes programming so engaging is that, while you can make the computer do what you want, you have to figure out how.

I'm personally convinced that computer science has a lot in common with physics. Both are about how the world works at a rather fundamental level. The difference, of course, is that while in physics you're supposed to figure out how the world is made up, in computer science you create the world. Within the confines of the computer, you're the creator. You get to ultimately control everything that happens. If you're good enough, you can be God. On a small scale.

And I've probably offended roughly half the population on Earth by saying so.

But it's true. You get to create your own world, and the only thing that limits what you can do are the capabilities of the machine and, more and more often these days, your own abilities.

Think of a treehouse. You can build a treehouse that is functional and has a trapdoor and is stable. But everybody knows the difference between a treehouse that is simply solidly built and one that is beautiful, that takes creative advantage of the tree. It's a matter of combining art and engineering. This is one of the reasons programming can be so captivating and rewarding. The functionality often is second to being interesting, being pretty, or being shocking.

It is an exercise in creativity.

The thing that drew me into programming in the first place was the process of just figuring out how the computer worked. One of the biggest joys was learning that computers are like mathematics: You get to make up your own world with its own rules. In physics, you're constrained by existing rules. But in math, as in programming, anything goes as long as it's self-consistent. Mathematics doesn't have to be constrained by any external logic, but it must be logical in and of itself. As any mathematician knows, you literally can have a set of mathematical equations in which three plus three equals two. You can do anything you want to do, in fact, but as you add complexity, you have to be careful not to create something that is inconsistent within the world you've created. For that world to be beautiful, it can't contain any flaws. That's how programming works.

One of the reasons people have become so enamored with computers is that they enable you to experience new worlds you can create, and to learn what's possible. In mathematics you can engage in mental gymnastics about what might be. For example, when most people think of geometry, they think of Euclidian geometry. But the computer has helped people visualize different geometries, ones that are not at all Euclidian. With computers, you can take these made-up worlds and actually see what they look like. Remember the Mandelbrot set¾the fractal images based on Benoit Mandelbrot's equations? These were visual representations of a purely mathematical world that could never have been visualized before computers. Mandelbrot just made up these arbitrary rules about this world that doesn't exist, and that has no relevance to reality, but it turned out they created fascinating patterns. With computers and programming you can build new worlds and sometimes patterns are truly beautiful.

Most of the time you're not doing that. You're simply writing a program to do a certain task. In that case, you're not creating a new world but you are solving a problem within the world of the computer. The problem gets solved by thinking about it. And only a certain kind of person is able to sit and stare at a screen and just think things through. Only a dweeby, geeky person like me.

The operating system is the basis for everything else that will happen in the machine. And creating one is the ultimate challenge. When you create an operating system, you're creating the world in which all programs running the computer live¾basically, you're making up the rules of what's acceptable and can be done and what can't be done. Every program does that, but the operating system is the most basic. It's like creating the constitution of the land that you're creating, and all other programs running on the computer are just

common laws.

Sometimes the laws don't make sense. But sense is what you strive for. You want to be able to look at the solution and realize that you came to the right answer in the right way.

Remember the person in school who always got the right answer? That person did it much more quickly that everybody else, and did it because he or she didn't try to. That person didn't learn how the problem was supposed to be done but, instead, just thought about the problem the right way. And once you heard the answer, it made perfect sense.

The same is true in computers. You can do something the brute force way, the stupid, grind-the-problem-down-until-it's-not-a-problem-anymore way, or you can find the right approach and suddenly the problem just goes away. You look at the problem another way, and you have this epiphany: It was only a problem because you were looking at it the wrong way.

Probably the greatest example of this is not from computing but from mathematics. The story goes that the great German mathematician Carl Friedrich Gauss was in school and his teacher was bored, so to keep the students preoccupied he instructed them to add up all the numbers between 1 and 100. The teacher expected the young people to take all day doing that. But the budding mathematician came back five minutes later with the correct answer: 5,050. The solution is not to actually add up all the numbers, because that would be frustrating and stupid. What he discovered was that by adding 1 and 100 you get 101. Then by adding 2 and 99 you get 101. Then 3 and 98 is 101. So 50 and 51 is 101. In a matter of seconds he noticed that it's 50 pairs of 101, so the answer is 5,050.

Maybe the story is apocryphal, but the point is clear: A great mathematician doesn't solve a problem the long and boring way because he sees what the real pattern is behind the question, and applies that pattern to find the answer in a much better way. The same is definitely true in computer science, too. Sure, you can just write a program that calculates the sum. On today's computers that would be a snap. But a great programmer would know what the answer is simply by being clever. He would know to write a beautiful program that attacks the problem in a new way that, in the end, is the right way.

It's still hard to explain what can be so fascinating about beating your head against the wall for three days, not knowing how to solve something the better way, the beautiful way. But once you find that way, it's the greatest feeling in the world.