Advice to a Newbie

Do you remember the first time? The first time when, after struggling with a piece of code, you felt not only “I can do this!” but also “I love doing this!” OK, perhaps the latter feeling has yet to arise, but if you’re reading this, chances are you have at least some appreciation for the art of coding.

As any experienced developer knows, however, pure coding is but one piece of the software development landscape. This month Kode Vicious shares some of this hard-won perspective with an inspired newbie. Even if you’re a seasoned engineer, it can be helpful to be reminded of the things you did—or should have done—when you were just starting out.

Dear KV,

I am new to programming and just started reading some books about programming, particularly C++ and Visual Basic. I truly enjoy programming a lot—to the extent that for the past couple of months I have never missed a day without writing some code. My main concern now is what the world holds for programmers. If someone is called a programmer (i.e., professionally), what will he or she really be programming? As in, will you always be inventing new software or what, really? This is mainly in the case of someone who will not be working for someone else. My other question is, can you really build a career as a freelance programmer, and what will you basically be programming?

Programming Newbie

Dear Newbie,

“What does the world hold for programmers?” This is such an interesting question that I’m not going to give you any abuse for describing yourself as a newbie nor for

Got a question for Kode Vicious? E-mail him at kv@acmqueue.com—if you dare! And if your letter appears in print, he may even send you a Queue coffee mug, if he’s in the mood. And oh yeah, we edit letters for content, style, and for your own good!

A koder with attitude, KV ANSWERS

YOUR QUESTIONS.

MISS MANNERS HE AIN’T.

mentioning Visual Basic in polite company. Not that I’m polite company, but there might be children around who could hear

you. We must consider the children!

Although picking up a few books on programming languages, which is what you have already done, will teach you about those languages, they will likely not teach you about programming but only about the syntax of the language you are studying. I’m going to go out on a limb here and assume that what you really want to be is not so much a *programmer*, a term that really ought to go out of vogue, but an *engineer*.

In the old days when I had hair and teeth, or, actually closer to when I was born and therefore had no hair and teeth, a programmer was the person who input the program into the computer. Other people designed the software and systems, and then the programmers—there were often legions of them—translated these designs into software code that the computer could understand.

With the advent of interactive programming, as opposed to batch programming, the line between programmer and engineer blurred. Now a single engineer can be the designer, programmer, documenter, quality assurance person, etc. Not that this is always the right choice, but it is how the industry has evolved. The number of people who are strictly programmers, therefore, is actually quite small.

If you want to become an independent—as you say, freelance—engineer, then you must put down those books on random programming languages and pick up a few books on how to build software. I have given my recommended reading list in a previous piece (“Kode Vicious Bugs Out,” April 2006) and so won’t repeat it here.

What I will say is that you may have one of the skills you need right off, which is that you practice every day. Computer science is both a science and an art. I don’t know why people feel the need to make that distinction so often, because most sciences are art, in the sense that to remain good at doing science you need to practice, just as with an art. The fact that you say, “...for the past

couple of months I have never missed a day without writing some code," leads me to believe that you have a life of drudgery and depression to look forward to! Oh, huh, that's not quite what I meant. What I meant is that practicing every day is a very important component of being a good engineer, or programmer if you like.

It's not enough just to spew out code, though, as anyone who has heard the self-taught neighborhood trombone player can tell you. After years of playing the same songs, he never gets them right. You need to take additional steps to really advance in your art. The first is to *read* as much code as you *write*, probably more. Software is a form of expression, like writing prose, and as with any type of expression there are idioms, phrases, and modes of thought that lead to pleasing—or in our case, working—designs and those that lead to disaster. Literature teachers are correct in their assertion that "reading the classics" can help with your writing (though it's too bad that they are so often forced to teach the most boring versions for fear of offending someone).

In software there are plenty of classics to read, and there are books about these classics. For networking, one of my main areas of interest, there is the FreeBSD operating system and its TCP/IP networking stack. In the database world there is MySQL, and for Web servers, Apache. These code bases are classics, and they are open source so they make excellent systems to study.

Having the full development history, in the form of a revisioned source code repository, is another reason to work with these systems, as you can see how they evolve over time, what changes took place, and for what reason. None is perfect, by any means, but each is well documented and an example of a living, breathing system. If you were an architect—of buildings, as architects design buildings and not software, though that conceit is currently popular—you would want to travel the world and study various pieces of architecture. Learning software is much cheaper, because you don't need a plane ticket—you can just stay home and study the great works of our industry with a simple Internet connection.

The next step is to get others to look at your code. Unless you are that one-tenth-of-one-percent genius who can do all this right the first time—and I think that .1 percent is being generous—you need people to mentor you and review your work. Again, open source can help here.

Submit patches to your favorite programs, and learn to work with others, as that is one of the great skills not taught in any computer science department. Note well that a mentor is just a guide and you must not grow too dependent on him or her, or you will actually learn nothing.

A mentor is the person to turn to for advice when you have come to the end of your wits, not when you have a question you can answer but are too lazy to work it out on your own. Laziness is a poor way to become a good engineer, and it pisses KV off no end.

As to what you can expect in the big wide world of software, well, like the rest of the world, 90 percent of it is crap. Whether you're a freelance engineer or working for a large company, you are rarely going to write huge amounts of pristine code on your own. You will definitely be working more often on other people's code than your original designs. There are two ways to handle this: the first is to kill every other engineer on earth, a task that could take a while and result in you losing your Internet privileges; the second is always to make sure you have some system that you are leading or working on. In reality, this is how I keep the remaining shreds of my sanity.

It amazes many of my non-computer-geek friends that after a day spent looking at and working on software, I would in the evenings and on weekends go and do what to them seems like the same thing, but it's not. When I work on my own designs and code, I am the master, I call the tune, and that has more value than most people can imagine.

To your last question the answer is, yes, freelance programmers can do quite well. Often the freelance folks get some of the least interesting work, though, at least in my experience, because they get the drudgery that the full-timers don't want to do. That doesn't mean you shouldn't follow this path, as it can be quite lucrative and if balanced well can actually leave you more time for your own projects. But you shouldn't fool yourself into thinking that you'll come in as a freelance person and design or build a large system for a group or company; that is a very rare event. What you'll be building is completely up to the people you work with. It could be just about anything, which is half the fun of working in software.

KV

CODE VICIOUS, known to mere mortals as George V. Neville-Neil, works on networking and operating system code for fun and profit. He also teaches courses on various subjects related to programming. His areas of interest are code spelunking, operating systems, and rewriting your bad code (OK, maybe not that last one). He earned his bachelor's degree in computer science at Northeastern University in Boston, Massachusetts, and is a member of ACM, the Usenix Association, and IEEE. He is an avid bicyclist and traveler who has made San Francisco his home since 1990.

© 2007 ACM 1542-7730/07/0400 \$5.00