Kode Vicious Bugs Out

This month Kode Vicious serves up a mixed bag, including tackling the uncertainties of heisenbugs—a nasty type of bug that’s been known to drive coders certifiably insane. He also gives us his list of must-reads. Are any of your favorites on the list? Read on to find out!

Dear KV,

I’m on a small team that is building a custom, embedded, consumer device that is due out by Christmas. Of course the schedule is tight and there are make-or-break dates that if we miss basically mean the product will never make it to market. Not the most fun environment in which to have problems.

The software was carefully specified and laid out and then simulated while the hardware was being manufactured. Now we have real hardware, and real problems as well. Aside from the timing issues we found when we were no longer running the software in a simulator, several bugs remain that show up only under very special circumstances and that disappear when I use the debugger or turn on the logging code built into the system.

When tools fail you like this, what do you do next?

Stuck in the Nitty-Gritty

Dear Gritty,

No problem in debugging is more annoying than the heisenbug. Named after the physicist Werner Heisenberg, who observed that the observer of an experiment can influence its outcome, these bugs can be seen only when you’re not looking. A heisenbug can happen at any time and at any place in your programs, but, of course they happen most often under deadline pressure and in a file that no one has touched in a long time, which is just Murphy’s law making your life even more interesting.

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.

Often no amount of logging will help you find a heisenbug. You have to sneak up on it, or take a different approach. Which

approach you take depends on what kind of system you’re building and what kind of tools you have. I’ve sat with another engineer and an oscilloscope trying to find one of these beasts in a system. It took a week, but after we were done we had fixed the bug and were able to ship the product. You’ll note that almost all heisenbugs are showstoppers—no shipping products with those in there.

One technique for tracking down such bugs is to use a debugger’s watchpoint support, in particular if the watchpoints have hardware support. A watchpoint allows the person debugging the program to tell the debugger to stop the program whenever a variable is written, read, or either of the two happens. Since most of the heisenbugs I’ve hunted are related to inappropriate data access or small amounts of pointer smashing, this has been an invaluable tool in finding such problems. Of course, this won’t work if compiling the code with debugging turned on hides the heisenbug; in that case you’ll have to get craftier, but the watchpoint is a good first step when the normal use of a debugger or logging doesn’t help.

A nontechnical aid in these situations is rest. If you have been staring at the same bug, slowly sipping very black coffee from your favorite mug for 8 or 12 hours, you’ve probably gotten (a) a bad case of the jitters and (b) nowhere near figuring out the problem. Take a walk, take a nap, take a shower—the first two will help your mind to relax and perhaps find another route to a solution, and the latter will make anyone still in the office more likely to sit near you and help you out.

It is said that 90 percent of the effort goes into the last 10 percent of a project, and I can tell you that 90 percent of that last 10 percent is solving these hard problems such as heisenbugs. Oh, you can say all you want, “If the system had only been designed correctly...” but by the time you’re saying that, it’s too late. You’ll just have to fix that bug, or make another cup of coffee.

KV

Dear KV,

I just finished my degree and started a new job at a big IT company in Silicon Valley. The work is OK, if a bit boring (how much can one do with Web pages, and who cares about blogs anyway?). I took this job because the company develops all its code on open source systems, and that means I get to look around at the code while fixing menial bugs, which is what they pay me for. Most of the people in my engineering group are recent grads, and a few of them seem interested in more than just taking home a big paycheck, counting their stock options, and planning on which car they'll buy when their first stock chunk becomes available. A few of us are passing around our favorite tech books, and we occasionally pass around your column. We have a little lunch bet going on which books you would recommend for your readers to have on their bookshelves. The person in the group whose list most matches yours gets lunch bought by the rest of us.

So, what's on your list? I want that lunch!

Hungry Reader

Dear Hungry,

Every programmer and engineer I know has a small cluster of books always near their work area. I think it's that set of books—the ones you can't, or shouldn't, live without—that you're after. The problem with book lists is that they're highly subjective and in a space as diverse as IT and computer science, where there is a new fad every... well, there goes one now! Uh, what was I saying? Oh, right, the lists are always subjective and pompous, but then so am I, so, I guess I can give you my list.

I would like to point out that not only are these books useful, but they are also well written and easy to read, which is very important when you have 400 or more pages of complex ideas. There is never any reason to read a book, no matter how important someone says it is, if it is not a well-crafted piece of writing.

***The Art of Computer Programming* by Donald Knuth (Addison-Wesley Professional).** Perhaps the best-known masterworks on computer science, these books are both for reference and relaxation. I received my first set as a Christmas present my freshman year of college, and yes, I requested them, as Santa rarely peruses the computer science section of bookstores. When you have a question about an algorithm or you're even thinking of optimizing some piece of code, these are the books to spend the day with. You will find out either that Dr. Knuth already knows the answer or that no one does and you're on your own. The books have been being written for almost 40

years now and are worth having near at hand.

***The Art of Computer Systems Performance Analysis* by Raj Jain (Wiley, 1991).** This is a book that seems to be much less well known than it should be. First published in 1991, it reads a bit dated now; the hardware used in its examples will either bring a nostalgic tear to your eye or just make you ask, "Who is DEC?" Dr. Jain is heavily involved in the networking side of computing, which shows in this book, but it is much more than a book about networking: It's a great book on applying the scientific method to solving problems in computer science. The book covers such useful topics as proper experiment design, workload selection, and all the other things you need to approach performance problems in your systems.

Anything written by W. Richard Stevens including, but not limited to, *TCP/IP Illustrated, Volumes 1 and 2* (Addison-Wesley Professional). Stevens loved to write, and that's obvious when you read his books. Most were about networking, and TCP/IP in particular, but some were broader, covering subjects such as programming in the Unix environment. Each book is interesting to read, has plenty of relevant examples, and teaches you something on every page.

***The Practice of Programming* by Brian W. Kernighan and Rob Pike (Addison-Wesley Professional, 1999).** This great book runs fewer than 300 pages, yet is filled with interesting stories about programming and with practical advice. It is one of those must-reads, and must-read-agains.

And, finally, a noncomputer book, *The Elements of Style* by William Strunk and E. B. White (fourth edition, Allyn and Bacon, 1999). No, it's not a book on how to dress for those of us who can't figure out if orange and green really do clash, but a very short book on the proper use of written English. Why would I suggest such a book? Science is, after all, the pursuit of knowledge via the scientific method, and one of the important components of the scientific method is that you are able to tell other people what you did and how you did it, so that they can verify your work. I don't care how clever your code is—if you can't explain it to someone else, it's useless. (I'm sure my editors wish I referred to this little book more often.)

Now, did you get the lunch, and how do I get my cut?

KV

KODE 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. He earned his bachelor's degree in computer science at Northeastern University and is a member of ACM, the Usenix Association, and IEEE.

© 2006 ACM 1542-7730/06/0400 \$5.00