

Re: Computer programmers' habits in electronics

Re: Computer programmers' habits in electronics

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2005-12/msg03822.html>

- *From:* Tim Wescott <tim@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 21 Dec 2005 14:39:20 -0800
-

Ignoramus32515 wrote:

On Wed, 21 Dec 2005 12:28:17 -0800, Tim Wescott <tim@xxxxxxxxxxxxxxxxxxx> wrote:

Ignoramus32515 wrote:

On Wed, 21 Dec 2005 19:31:08 GMT, Rich Grise, but drunk <yahright@xxxxxxxxxxxxx>

On Tue, 20 Dec 2005 19:49:09 +0000, Ignoramus10397 wrote:

On Tue, 20 Dec 2005 19:29:21 GMT, Rich Grise, but drunk <yahright@xxxxxxxxxxxxx>

Got any
overflow
work?

I am sorry, what is
overflow work.

Re: Computer programmers' habits in electronics

When you have so many customers that you can't fill the need, so you turn on outside help temporarily to take care of the overflow. You know, like "My cup runneth over", but of work?

I'm kinda looking for some stuff where I could telecommute; I know just enough C and perl to get myself in trouble, and can do hobbyist-level electronics - I used to be able to slap together uC circuits, but I don't really have a proper lab these days.

I'm wondering if I should look around for proofreader work, or does anybody bother to have anything proofread these days?

Sorry, nothing that I can think of, of the sort.

Somewhat tangentially...

We are interviewing people for computer programmer positions. We are looking for those who can actually "do stuff" without too much babysitting.

Lots of people come in with impressive resumes. When I talk to them, I know that some people are very good at bullshitting, so I give them a couple of actual tasks to do. Very small simple things. One is to write a nice function that reverses a string in place. For example, "Rich" would become "hciR".

Almost no one can actually do this without making mistakes, many people give up completely.

Very frustrating. I consider it the most basic capability of a programmer.

i

Re: Computer programmers' habits in electronics

Someone who likes to ask that question told me of an interviewee who got as far as saying "I think it involves recursion...".

So I wrote a version that did it using recursive function calls and sent it to her. I don't know if I would have gotten the job -- they had already made the mistake of hiring me.

That's interesting, although as you can well see, not the most efficient.

Good thing the PC has a lot of stack space.

yep

For interviewing embedded SW engineers we finally settled on a fairly basic scaling problem. We started with a little story problem that required the interviewee to find the ratio of a couple of numbers and multiply it to a third, then we said "oh, by the way, our floating point library is too slow -- do it with integers". The question usually took about 40 minutes to explore fully, with some folks never getting it and some just glancing at the board and writing down the correct answer.

That's interesting. How is your actual problem phrased? Just curious.

It's amazing how you can separate the desktop programmers from the embedded engineers with that one.

Re: Computer programmers' habits in electronics

That's true... Much better than just chatting with people.

i

It was a display problem, where we had a display of a certain known size, and we wanted to plot a line on it that reflected an even-numbered (10, 20, 50, 100, etc.) length. We told them that we wanted a procedure that took the screen size in real-world coordinates, the desired size in real-world coordinates, and the number of pixels in real-world coordinates, and that coughed up the line size in pixels. It actually came from a real product (sometimes we had system pictures in the conference rooms & we could point to the box & say "it's in that there box, as a matter of fact").

If they didn't start that way themselves we'd encourage them to "just do the math" first, and look for how well they did their 8th grade algebra problems.

Once the math was done, and sorted out into something that worked well, we'd say "Oh, that's good, now write the procedure that calculates it". Sometimes the interviewee would have to be encouraged to just do it in floating point with no error handling -- if it looked like they were concerned with side issues because of good programming habits we'd give them extra credit.

Once the function was written then we'd drop the bombshell: "This particular code base doesn't have a functioning floating point library -- if you try to do any floating point operations the processor will lock up. Please do this with integer arithmetic". Then we'd spend the rest of the time allotted for the question working through issues of truncation, overflow and (for extra credit) rounding.

We probably only saw 10% of the candidates make it through the question 100% unscathed, so we didn't grade on results. Instead we looked for how quickly folks picked up on the concepts being thrown at them and figured out solutions.

Interestingly enough the folks that did the worst at this were the telecom guys. In spite of doing "embedded" software they never had to do math and they always worked with 32-bit machines -- to the point where some of them required convincing that 'int' in C could refer to a 16-bit number, and many had no concept of the notion that a 'long int' is, well, longer than an 'int'.

Probably the most amusing anecdote that came out of this was a guy we

Re: Computer programmers' habits in electronics

hired after he did quite well on The Question. His first task on the job was to embellish a control loop, which he did using floating point. I was called in to figure out why it didn't work, and was able to point out that the code was just too damn slow. When I pointed out that he did so well at The Question he replied that he thought it was just a question -- he didn't realize it was a warning, too.

--

Tim Wescott
Wescott Design Services
<http://www.wescottdesign.com>

.