Sometimes it takes an outsider

Craig Hackerd - March 29, 2018

In the 1990s, I was working as an embedded software engineer at a telecom company. They had decided to branch out in a new direction, and had farmed out the design and programming of a new device. This device required several configuration items unique to each customer. The IT department designed a database program to configure and maintain the information. Each unit came enclosed in a shipping box with a small cutout that allowed power and a serial connection to the unit, so that it could be configured before shipping.

The manufacturer provided a windows DLL so that the database program could communicate with the device, retrieve the serial number of the unit, and download the configuration data. As production started, the shipping department ran into a problem. The PCs they were using to configure the units were crashing randomly after just one to three configurations, requiring the PC to be rebooted. After two weeks of this, the shipping manager was pulling his hair out because the IT department hadn’t been able to solve the problem. So he came to my boss, who picked me to go help.

First I walked over to the shipping department to observe their procedures and see if they were doing anything wrong. They had no problems connecting to the devices, and the first configuration always worked. It was only after a configuration was completed that the PC would crash and need to reboot. I then went to the IT department, and the manager greeted me with open arms (NOT!). After convincing him that it wouldn’t hurt for me to take a look, I was escorted over to the programmer. The programmer welcomed the help, and he gave me copies of the DLL source and his program, and showed me how his program worked. He was using a high level database programming language that automatically took care of many of the lower level details.

I had never done any device level programming on a PC before, but I was very good with C (the DLL language), and had done some limited database programming previously. I started with the DLL code, specifically looking at the interface it presented to the database program. After about a half hour, I thought I knew what the problem was. The DLL required a pointer to a memory location where it would store the unit serial number. It would then receive the configuration data and pass it on to the device for storage. The high level database program was passing the pointer to the DLL,
but because it didn’t know the length of the serial number variable, it didn’t reserve any space for it. When the DLL returned the serial number, it was being written to a location on the stack, and depending on the values would sometimes recover, but otherwise just crash.

Taking my suspicions back to the programmer, he reserved space for the serial number first, then passed the pointer to the DLL. Problem solved. We shipped many thousands of these units, and the shipping manager praised my name all the way to the president of the company. Sometimes it takes an outsider.

—Craig Hackerd has been designing and programming embedded computer systems since the 1970s. He currently designs and programs many small projects to play with new microcontrollers.

Related articles:

- It's the firmware, of course
- Five more top causes of nasty embedded software bugs
- 11 steps to successful hardware troubleshooting
- The mythical software engineer