

## Re: Disobeying jet engines – why?

---

*Source:* <http://sci.tech-archive.net/Archive/sci.electronics.design/2008-01/msg04997.html>

---

- *From:* Martin Brown <|||newspam|||@nezumi.demon.co.uk>
  - *Date:* Thu, 31 Jan 2008 11:13:47 +0000
- 

In message <fnq85j\$7et\$1@xxxxxxxx>, Jan Panteltje <pNaonStpealmtje@xxxxxxxx> writes

On a sunny day (Wed, 30 Jan 2008 07:08:01 –0800) it happened John Larkin <jjlarkin@xx> wrote in <4f41q3hq25ra51d18dfa5jv1cnkge8klnc@xxxxxxxx>:

Sometimes the right thing to do is to buy the correct development tool for the job. Anyone who attempts to write a database from scratch for a PC wants their head examining (and that was true almost from the early days of CPM). Same with anyone who attempts to debug embedded code in a hard RT environment without using the right tools.

Umm, I should have my head examined. Twice.

John

Hey, John, I agree.

Did not want to comment at first, and that is especially about debug.

If you know your code, then you need no debugger! HIGH LEVEL LANGUAGES LIKE HEX or ASM need no debugger (I ain't kidding).

And what if it is someone else's code (which is frequently the case on large projects) and almost always the case when contracting.

Most I will do in huge programs in C (C++ is not really a language but a disability, Stroustrup did not know how to program, and that is why he created C++), is add some printf() statements.

What a dinosaur. You might at least use the ASSERT, VERIFY and TRACE macros so thoughtfully provided

Re: Disobeying jet engines – why?

in modern implementations.

Honestly last time I used a debugger was in 1983? They tried to sell me all sorts of stuff, ICE, hell, you should be able to understand what is going on from what happens.

So you come back to a piece of kit that isn't working and have to binary chop to find the failing line of code each time carefully reproducing whatever situation caused it to fail. Or alternatively waiting hours or days for the failure situation to happen again by chance.

The relatively simple post mortem debug tools we had in the mid 80's would log the failing address, failure code, stack top and registers. From that and the link map you could go to the exact failing line of code and knowing how it failed usually work out why.

This method worked to allow the last remaining bugs in the shipped systems to be practically eliminated as the number deployed increased and failures became rarer and rarer. You can even get MS compilers to do this trick now.

I am inclined to agree that there is too much mindlessly watching the screen in a runtime debugger with brain in neutral these days. But that is not the fault of the debugging tools it is the fault of the users.

Regards,

—

Martin Brown

—

Posted via a free Usenet account from <http://www.teranews.com>

.