The Embedded Muse 82

Editor: Jack Ganssle (jack@ganssle.com) March 7, 2003

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor's Notes

The next Better Firmware Faster seminars will be in Chicago on April 15 and Baltimore/Washington on April 17 – see http://www.ganssle.com/classes.htm for more information.

I often do this seminar on-site, for companies with a dozen or more embedded folks who'd like to learn more efficient ways to build firmware. See http://www.ganssle.com/onsite.htm.

A lot of email lately has been asking about ways to make C safer. Though C is a powerful language, it should come with an "adults-only" disclaimer, as tiny errors can yield disastrous consequences.

The classic work on using the C language safely is "Safer C: Developing Software for High-Integrity and Safety-Critical Systems" by Les Hatton. It's out of print now, and is somewhat dated. But if you can find a copy there's a lot of good ideas to be learned.

A more recent work is the MISRA (Motor Industry Software Reliability Association) standard (www.misra.org.uk), which defines C constructs to be avoided. A very interesting work, indeed, and you may disagree with some of their recommendations. I reviewed this a couple of years ago; the review is at http://www.ganssle.com/tem/tem52.pdf.

Comments on Commenting

Arrghhh! I can't stand it anymore! I just read the source code of a small PIC-based system, and was appalled by the commenting conventions. Actually, "conventions" is much too generous of a term. The developers documented the obvious, neglected the obtuse, used capital letters randomly, rarely worried about sentence structure and abandoned lucidity. No substantial truth was to be found in the comments. In my opinion, extracting meaning from assembly is difficult and a horrible waste of time. This product's code was as much of a mess as its comments, not an uncommon occurrence.

Here's a few rules about commenting:

1. If the comments don't agree with the code, they're not worth much. Neither is the code.
2. If the code is not absolutely self-commenting, rewrite the code. Good code needs fewer comments than lousy code.
3. Comments should provide extra info not readily attainable from the code
4. When you turn an ordinary page of code into just a handful of instructions for speed, expand the comments to keep the number of source lines constant.
5. If the code and the comments disagree, both are probably wrong.

I wrote about commenting in Embedded Systems Programming recently. A copy of that article is on-line at www.ganssle.com.

Reader Feedback

In the last Muse I reviewed Karl Wiegers new book "Peer Reviews in Software", and wrote: "In Maryland we're required to have our cars tested for emissions every two years. I dutifully took mine in last week. Another fellow in the waiting room went on a loud rant about the experience. His complaint (minus the four letter words) was that, since practically every car passes, it's all a waste of time."

Well, it turns out there's no process that can't be perverted, and no technological solution to bad governance. Mark Blough had much to say about his experiences:

I too recently went to the Maryland MVA VEIP to have my emissions tested. I was out of my car for less than two minutes when the technician told me I was done. They never hooked up anything to my tail pipe, didn't put me on the treadmill, as far as I could tell they hadn't done anything but take my $14.00. So I asked... Since August of last year, if your car is a 1996 or newer, they just plug in a diagnostic probe under the dash and ask your car's computer if the emissions are OK. If the computer says everything is all right,

you pass. Its called the ODB (onboard diagnostic) test. Of course, if your cars computer thinks there is something wrong with your emissions, its not going to keep that to itself, that's what the dreaded "Check Engine" light on the dash is for. In fact, buried in the paper work from the MVA is a little note that says to the effect don't bring in a car with the check engine light on, because it will fail.

So for your waste of time and $14.00 you get someone to confirm that the light bulb for your check engine light is not burned out (just not on). No need to actually check if the emissions are good and the on board computer isn't fooled by some sensor somewhere.

I suppose there are people who drive around with the check engine light on and to force these idiots to get their cars repaired, we all could drive to the VEIP station every two years, hand the guy $14 bucks to have him stick his head in the window and confirm that the check engine light isn't on. No actual testing needed to weed out that burnt out check engine bulb...how many millions of dollars are wasted to find that 1 in a million bulb?

Practically every car passes the current test, as more cars "retire" and more are 1996 and newer, EVERY car will pass unless its check engine light is on. Since we are now trusting the on board computer and sensors, maybe we could have it disable the car after 500 miles (or 1000 miles) of driving with the check engine light on and force the owner to have it fixed. Save us all the expense of this elaborate testing program, and catch the polluters (or at least the confused on board computers) in short order and make the world a better place.

Thought for the Week

Creators Admit UNIX, C Hoax
In an announcement that has stunned the computer industry, Ken Thompson, Dennis Ritchie and Brian Kernighan admitted that the Unix operating system and C programming language created by them is an elaborate prank kept alive for over 20 years. Speaking at the recent UnixWorld Software Development Forum, Thompson revealed the following:

"In 1969, AT&T had just terminated their work with the GE/Honeywell/AT&T Multics project. Brian and I had started work with an early release of Pascal from Professor Niklaus Wirth's ETH labs in Switzerland and we were impressed with its elegant simplicity and power. Dennis had just finished reading 'Bored of the Rings', a National Lampoon parody of the Tolkien's 'Lord of the Rings' trilogy. As a lark, we decided to do parodies of the Multics environment and Pascal. Dennis and I were responsible for the operating environment. We looked at Multics and designed the new OS to be as complex and cryptic as possible to maximize casual users' frustration levels, calling it Unix as a

parody of Multics, as well as other more risque allusions. We sold the terse command language to novitiates by telling them that it saved them typing.

Then Dennis and Brian worked on a warped version of Pascal, called 'A'. 'A' looked a lot like Pascal, but elevated the notion of the direct memory address (which Wirth had banished) to the central concept of the language. This was Dennis's contribution, and he in fact coined the term "pointer" as an innocuous sounding name for a truly malevolent construct.

Brian must be credited with the idea of having absolutely no standard I/O specification: this ensured that at least 50% of the typical commercial program would have to be re-coded when changing hardware platforms. Brian was also responsible for pitching this lack of I/O as a feature: it allowed us to describe the language as "truly portable".

When we found others were actually creating real programs with A, we removed compulsory type-checking on function arguments. Later, we added a notion we called "casting": this allowed the programmer to treat an integer as though it were a 50kb user-defined structure. When we found that some programmers were simply not using pointers, we eliminated the ability to pass structures to functions, enforcing their use in even the simplest applications. We sold this, and many other features, as enhancements to the efficiency of the language. In this way, our prank evolved into B, BCPL, and finally C.

We stopped when we got a clean compile on the following syntax:

```
for(;P("\n"),R-;P("|"))for(e=3DC;e-;P("_"+(*u++/8)%2))P("| "+(*u/4)%2);
```

At one time, we joked about selling this to the Soviets to set their computer science progress back 20 or more years.

Unfortunately, AT&T and other US corporations actually began using Unix and C. We decided we'd better keep mum, assuming it was just a passing phase.

In fact, it's taken US companies over 20 years to develop enough expertise to generate useful applications using this 1960's technological parody. We are impressed with the tenacity of the general Unix and C programmer. In fact, Brian, Dennis and I have never ourselves attempted to write a commercial application in this environment.

We feel really guilty about the chaos, confusion and truly awesome programming projects that have resulted from our silly prank so long ago."

Dennis Ritchie said: "What really tore it (just when Ada was catching on), was that Bjarne Stroustrup caught onto our joke. He extended it to further parody, Smalltalk. Like

us, he was caught by surprise when nobody laughed. So he added multiple inheritance, virtual base classes, and later ... templates. All to no avail. So we now have compilers that can compile 100,000 lines per second, but need to process header files for 25 minutes before they get to the meat of "Hello, World".
Major Unix and C vendors and customers, including AT&T, Microsoft, Hewlett-Packard, GTE, NCR, and DEC have refused comment at this time.

Borland International, a leading vendor of object-oriented tools, including the popular Turbo Pascal and Borland C++, stated they had suspected this for a couple of years. In fact, the notoriously late Quattro Pro for Windows was originally written in C++. Philippe Kahn said: "After two and a half years programming, and massive programmer burn-outs, we re-coded the whole thing in Turbo Pascal in three months. I think it's fair to say that Turbo Pascal saved our bacon". Another Borland spokesman said that they would continue to enhance their Pascal products and halt further efforts to develop C/C++.

Professor Wirth of the ETH institute and father of the Pascal, Modula 2 and Oberon structured languages, cryptically said "P.T. Barnum was right." He had no further comments.

About The Embedded Muse

The Embedded Muse is an occasional newsletter sent via email by Jack Ganssle. Send complaints, comments, and contributions to him at jack@ganssle.com.

To subscribe, send a message to majordomo@ganssle.com, with the words “subscribe embedded *your-email-address*” in the body. To unsubscribe, change the message to “unsubscribe embedded *your-email-address*”.

The Embedded Muse is supported by The Ganssle Group, whose mission is to help embedded folks get better products to market faster. We offer seminars at your site offering hard-hitting ideas - and action - you can take now to ***improve firmware quality and decrease development time***. Contact us at info@ganssle.com for more information.