
Embedded Muse 177 March 16, 2009

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor’s Notes

Did you know it IS possible to create accurate schedules? Or that most projects consume 50% of the development time in debug and test, and that it’s not hard to slash that number drastically? Or that we know how to manage the quantitative relationship between complexity and bugs? Learn this and far more at my Better Firmware Faster class, presented at your facility. See http://www.ganssle.com/classes.htm .

SILICON VALLEY UPDATE: I’m presenting a public version of the Better Firmware Faster class April 3 in San Jose, CA. Come to the Embedded Systems Conference that week AND join me for a day to learn my approach to crafting world-class systems in less time. See http://www.ganssle.com/classes.htm for more info. It’s right off the light rail line and there’s plenty of free parking.

The Embedded Systems Conference folks have offered a 25% discount to Muse readers. Use priority code CTPM15 to get the discount when you register.

Brazil! The country of samba, feijoada and caipirinha. And, of course, embedded systems. I’ll present a two-day version of my Better Firmware Faster seminar in Sao Paulo March 25th and 26th. See http://workshop.embarcados.com.br for more information.

I’ll be speaking at the Deep Agile 2009 event in Cambridge, MA. Here are the details:
Deep Agile 2009: Agile for Embedded Systems
Date: April 25 - 26, 2009 (Saturday & Sunday)

Registration at http://www.agilebazaar.org/

Max Maxfield, with whom I'm engaged in a walking competition (1000 miles in 2009; so far he's way ahead) is putting together a cool site about the way things were. Feel free to submit your own thoughts: http://www.thewaythingswere.com .

Quotes and Thoughts

Dave Wyland wrote: "I prefer the following definition, heard a long time ago, from I-can't-remember-where:

"Debugging is like alien abduction. Large blocks of time disappear, for which you have no explanation."

Multiprocessing in C

Walter Banks is trying to extend the C standard to handle multiprocessing/multicore systems. He wrote: "I am back at doing some work with WG14 the ISO committee responsible for standardizing C for all to use: "Several years ago I co-authored a document related to C for embedded systems. ISO/IEC TR 18037 "Programming languages - C - Extensions to support embedded processors"

"In the meeting in a few weeks I will be giving a survey paper on what if any C language support should developed to support the emerging multiprocessor systems. At this point my primary objective is that WG14 should either embrace multiprocessor language support or have a defensible reason why not.

"It occurred to me that many of your readers have a voice that rarely gets heard. I will incorporate any thoughts (pro and con) into the formal presentation and debate.

"The draft paper is here: http://www.bytecraft.com/downloads/N1351.pdf . Send any comments to walter@bytecraft.com ."

Computer Science Education

Pete Klammer wrote: "I am more concerned about "Software Engineering" than Computer Science education, because it is the engineering phase of software product development -- or the deficiency, or even lack of it -- that has the worst impacts on society and our lives. It matters little, as I negotiate life's traffic with automated aids, whether this or that algorithm is O(n) vs. O(n^2), as computer science may determine. On the extreme other hand, it matters entirely which algorithm was selected, and how implemented, as the remaining time in the path to my decision point reaches zero.

"I think that I know what good engineering is -- well-documented application of known principles to measurable problems -- and I know that I know that plenty of computerized electronic stuff is produced without much or any good engineering. Part of how I know is by how

many practitioners educated by liberal-arts, mathematics, and business-school departments are willing to let themselves be called, individually or by vendor certifications, "software engineers." I personally have a hard time accepting such nomenclature of a candidate who cannot pick out the correct definitions of Ohm's Law or hysteresis on a multiple-choice test. Thus, "software engineering" has been diluted to the point of oxymoronity.

“So I endorse the efforts to raise Software Engineering status by board testing and licensure as a Professional Engineering discipline. If nothing else (not that I only expect so little), the effort and trouble of obtaining P.E. licensure should indicate preparation, care, and experience in the professional field. But the more significant consequences should be broader, more thorough, and more standardized curricula and accreditation, as well better practitioners and higher quality results.

“I don't advocate prohibition of programming by unlicensed developers. But I do want something like truth in labeling: a manufacturer or public agency should be able to tell (and consumers should be able to find out) whether or not its software or firmware was developed by, or under the responsible supervision of, a licensed Software Professional Engineer, and with that information we can shop and negotiate and pay accordingly. Perhaps some critical (life-critical? Mission-critical? Safety-critical) products may even require such development, but I'm not going that far myself: just give me the choice at least, and I for one will pay a little more for the airplane, car, X-ray machine, etc., that came that way.

“See PE Magazine Jan/Feb'09 page 17, "Alliance Builds Support for Software Engineering PE Exam" for some thoughts on this.”

Responses to Naming Conventions

A number of people responded to my thoughts on naming conventions. Some agreed, others objected. What fun! Here are John Carter’s thoughts: “Ooo! My Hobby Horse! Let me ride it!

“Developers with a hardware background are poor misshapen, dreadfully abused, broken folk.

“And alas, embedded software is often written by such hag ridden refugees from the harsh cruel world of hardware.

“They have lived their lives with components labeled things like "N3D7R819". What does it do? Nobody knows, look it up in the datasheet.

“Does one thing label "N3D7R819" do the same thing as another labeled "N3D7R819"? Not quite, there was a major hardware revision that came in devices with serial numbers greater than 200289938. But the N3D7R8l9 is the same, it's just a different packaging.

“Alas, sometimes they mistake convention, lack of label space and good old stupidity for Best Practice and inflict these horrors on the Software Domain. We must forgive them, they suffered through a truly dreadful upbringing. :-)

“More than anything, we _are_ the "Department of Notational Engineering". Our living _is_ making up names. More than anything, it is What We Do, so we better be good at it.

“Get the name Right, and odds on it will grow to do The Right Thing.

“Keep the name Right, and odds on nobody will abuse it later.

“If you can't name a thing Right, it's a strong indicator that, not even you, knows what it does.

“If it isn't named Right, don't be surprised when maintainers treat it according to its name, rather than according to what it actually is.

“So what's a Right name? Well, it comes down to ye olde Single Responsibility Principle. If you can't state very simply and clearly what the Single Responsibility of this thing is... you need to be thinking harder, a lot harder.

“Perhaps break it into two (or more) things.

“Pick another similar thing. What is it called? Can you tell (obviously) from the names what the difference in responsibility between the two is?

“Perhaps you need to rename both.

“Imagine telling a new hire about this thing, would he guess from the name what it's about, or would you start to feel embarrassed and awkward and mumble that it was done in a hurry?

“Perhaps call it FRED for a minute...and write down in a bit more detail how you're going to implement it, especially what state it holds, and what constraints must be enforced on that state. Then think of the name. But only leave it called FRED for at most about 5 minutes.

“What are Bad names? System, manager, process, data, interface, computer. The SystemDataManagerComputerProcessingInterface is down. Eh, What!? Sounds smooth, rolls off the tongue, says nothing as every project you ever worked on is a system with several computers that manage and process data and they all have an interface.

“If it ain't broke, don't fix it? How many times have you heard that?

“If the names are broke, it's broken and will grow more so, so fix it now.”

Mark Borgerson, and many others, took me to task: “When discussing naming conventions, you had:

Street addresses make no sense. Why do we label an envelope:
2000 Lakeshore Drive
New Orleans, LA

That's exactly the opposite of how the post office sorts the mail. A

better form is: LA, New Orleans, Lakeshore Drive, 2000

That's written from the big to the small. Start with the general - the State - and work to the specific - the street number.

“Actually, the Street address convention does make a lot of sense.

“Automatic sorting machines handle the zip code, state, and city stuff---and probably some of the binning for the carriers. Those machines have no problem reading from the bottom up. However, it is the human mail carriers who have to make that final decision based on the name and street address. Better to have the information that they need to make that final decision in the easiest place for them to read. They already know that they're in New Orleans and on Lakeshore Drive. Why should they have to read through all that other data to get to the '2000'?

“Having the name of the person at the top makes sense when I fetch the mail from the mailbox also. At that point, I only need to read the top line to decide whether the letter is for me or for a family member.

“Purely hierarchical addresses make sense for machines. They make less sense for the human operators at the end of the mail delivery system.”

Naming Conventions

So, in the interest of continuing to stir up debate, let me make a few more comments about naming conventions.

Spelling matters. Misspelled words are a sign of sloppy work. Our crummy tools, though, don’t do any sort of spell-checking, even though programmers have given the rest of the world fabulous tools that immediately flag a misspelled word. Invariably a spelling error will creep in from time to time. When discovered, fix it. It’s nearly impossible to maintain code littered with these sorts of mistakes, as now the developer has to remember the oxymoronic “correct misspelling” to use.

Long names are a great way to convey meaning, but C99 requires that only the first 31 and 63 identifiers to be significant for external and internal names, respectively. Restrict all names to 31 characters or less. (And define them as “static” if their scope is local. There are some good thoughts about the efficiency of statics in the Dec 13 posting here: http://www.embeddedgurus.net/stack-overflow/2008_12_01_archive.html).

Don’t redefine a name using C’s scoping rules. Though legal, having two names with different meanings is confusing. Similarly, don’t use names that differ only in case.

On the subject of case, it’s pretty traditional to define macros and constants in upper case while using a mix of cases for functions and variable names. That seems reasonable to me. But what about camel case? Or should I write that CamelCase? Or is it camelCase? Everyone has a different opinion. But camel case is merely an awkward way to simulate

a space between words, which gets even more cryptic when using acronyms: UARTRead. Some advocate only capitalizing the first letter of an acronym, but that word-izes the acronym, torturing the language even more.

We really, really want to use a space, but the language recognizes the space character as an end-of-token identifier. The closest typographical character to space is underscore, so why not use that? This_is_a_word is, in my opinion, easier to grok while furiously-scanning hundreds of pages of code than ThisIsAWord. Underscore's one downside is that it eats away at the 31 character name size limit.

Developers have argued passionately both for and against Hungarian notation since it was first invented in the 70s by space tourist Charles Simonyi. At first blush the idea is appealing: prefix variables with a couple of letters indicating the type, increasing the name's information density. Smitten by the idea years ago I drank the Hungarian cool-aid.

In practice Hungarian makes the code ugly. Clean names get mangled. szString means "String" is zero-terminated. uiData flags an unsigned int. Then I found that when changing the code (after all, everything changes all of the time) sometimes an int had to morph to a long, which meant editing every invocation of the name. One team I know avoids this problem by typedefing a name like iName to long, which means not only is the code ugly, but the Hungarian nomenclature lies to the unwary.

C types are problematic. Is an int 16 bits? 32? Don't define variables using C's int and long keywords; follow the MISRA standard and use the following typedefs to remove all ambiguity, and to make porting much simpler:

int8_t – 8 bit signed integer
int16_t – 16 bit signed integer
int32_t – 32 bit signed integer
uint8_t – 8 bit unsigned integer
uint16_t – 16 bit unsigned integer
uint32_t – 32 bit unsigned integer

See
http://www.opengroup.org/onlinepubs/009695399/basedefs/stdint.h.html
for some interesting extensions to these typedefs for use where performance issues mean we want the compiler to make the smartest decisions possible.

Jobs!

Let me know if you're hiring firmware or embedded designers. No recruiters please, and I reserve the right to edit ads to fit the format and intents of this newsletter. Please keep it to 100 words.

Again, none this week, though a lot of people wrote about losing their jobs.

Joke for the Week

Mark Hudson sent in:

There is no 'I' in 'TEAM' but there is 'ME' in 'BLAME'

About The Embedded Muse

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