
Embedded Muse 183 Copyright 2009 TGG October 19, 2009
Permanent link to this issue: www.ganssle.com/tem/tem183.pdf

EDITOR: Jack Ganssle, jack@ganssle.com

CONTENTS:

Editor's Notes

I've been in this industry since the first 8 bit microprocessor appeared. In the intervening years the technology has changed in breathtaking ways.

The most radical change, though, is in how the business environment has been driving engineering. I'm not referring to the current economic woes; rather, communications technology has improved so that globally-distributed teams now work together to build a product. When I was a child a long-distance phone call (even one from an adjoining state) was so rare, and so expensive that the household came to a halt as dad took the call. It could take weeks to trade letters with a correspondent in Europe, and few of us knew anyone, other than soldiers, who had traveled outside the US and Canada.

Today it's all but free to call any place on the planet, and the Internet means all of our data is always connected.

The result: a global competitive environment heretofore unknown that drives management to demand more, better, faster. If you can design an ASIC with 100 million gates and a million lines of C in a month, well, they'll soon demand it in a week. And for good reason, as every company realizes that someone else will be looking for a way to get those kinds of productivity gains.

Sure, the economy is bad right now. You're buried in getting a project done and don't have time to find new ways to do things better. I'm reminded of the classic cartoon of a medieval battle being waged with swords and rocks; the general sniffs that he just doesn't have time to see the salesman, who product is a box of machine guns.

More, better, faster. I think that's the mantra that will drive this industry for the foreseeable future. How will you achieve those goals?

Take a day to learn how to meet that objective. Learn how at my Better Firmware Faster class, presented at your facility. See http://www.ganssle.com/classes.htm .

Kevin Fodor implemented some of my debouncing ideas (from http://www.ganssle.com/debouncing.pdf) at this site: https://sites.google.com/site/kfodorprojects/home/electronics-projects-1/hex-switch-debouncer-module .

Quotes and Thoughts

In “A View of the Parallel Computing Landscape” (Communications of the ACM, October 2009) the authors state that over the last 60 yrs the IT industry has improved the cost-performance of sequential computing by about 100 billion times overall. What an astonishing number!

Tools and Tips

Archis Bhave recommends Launchy: “www.launchy.net is a keystroke launcher with catalog facility. No more hassles to find where one stored the files, may they be images or data sheets. Runs on Linux (with a source build) and windows both. Can launch installed applications as well.”

He also likes KiCAD: “kicad.sourceforge.net is an open source and feature rich schematic editor and PCB lay-outing tool. Nice for the schematics, good for designing quick prototypes.”

Well-known author Jam Axelson suggested three tools: “OpenWide (http://lingo.atspace.com/openwide.html) customizes the File > Open dialog in applications (not just Windows Explorer) to default to the size, position, and view (List, Details, etc.) that you prefer.

“Clipmate Clipboard Extender (http://www.thornsoft.com) remembers and manages up to thousands of clipboard items. The PowerPaste feature will easily paste a series of clips in sequence.

“Ultramon (http://www.realtimesoft.com/ultramon) helps manage multiple monitors, including easy moving of maximized windows and optional custom taskbars for each monitor.”

Static Analysis

I’m fascinated by the various static analysis tools that find execution-time bugs without actually running the code. Static analysis can find problems like buffer overflows. The tools go beyond what Lint can find, but do not replace Lint.

But how do the tools stack up against each other? A newish paper attempts to answer this question. “A Comparative Study of Industrial Static Analysis Tools by Par Emanuelsson and Ulf Nilsson” (http://www.ep.liu.se/ea/trcis/2008/003/trcis08003.pdf) is a long (34

pages) but quite interesting and well-written comparison of static analysis tools from Coverity, Parasoft and Klocwork. Well worth reading for those looking into this technology. The authors reach a number of conclusions, but perhaps most interesting is that that each tool finds different bugs! None are cheap, and few can afford the cost to run all of the tools together.

Perhaps the ultimate solution is to nationalize these companies and unify the rule sets. Oops – that should be in the jokes section!

More on Debugging

Jef Mangelschots had a comment about my take on debugging from the last Muse: “In my experience with debugging. if you can't explain the bug, you will almost never be able to successfully fix it.

“Therefore, you need to (1) reproduce the problem (sometimes difficult), (2) prove the validity of your hypothesis with captured data and, after fixing the problem, (3) prove the validity of the fix with the same method as used to prove the hypothesis. This is usually done with extracting available information (e.g. log files, measurements, ...) or by instrumenting the software exhibiting the problem with extra logging (or any other means of capturing test data).

“Also, the most important advice I can give is: a problem is half fixed when you can visualize the problem. Instead of pouring endlessly over raw test data, write a simple filter (I like Python but you can do it in Perl or any other high-level language) that processes the incoming test stream and extracts the information you are interested in and then visualize the conclusions to support your hypothesis. The simplest form is a text output but you can also use CSV output to import into Excel (sky is the limit). Try to show obtained and expected values side-by-side and a general PASS/FAIL outcome. Spend a couple of minutes on the layout of presenting the results. This layout influences the understanding of the problem.”

Jef’s last point is important. Edward Tufte’s excellent book “The Visual Display of Quantitative Information” shows how important it is to think carefully about how one presents data.

Naming Conventions

A scrambled paragraph in the last Muse showed how good people are at extracting meaning from massively-misspelled words. Don Peterson wrote a program that does this sort of scrambling – you can get it from http://www.gdssw.com/tools/ . For fun he ran the last issue of this newsletter through it, which is posted here: http://ganssle.com/misc/scrambled-muse.pdf .

Too Much Optimism

In the last issue I wrote: His testosterone came across in the same way exhibited by the crab. Whenever memory was needed, he issued a malloc(), and by God, that was a demand from this programmer to get

some memory! Gimme some memory, NOW! A gentler and more robust approach might be to recognize that malloc() has a return code. It might fail. Check the return code and take action if the system is low on resources. Demand memory? No, request it, with a sort of virtual "please", and recognize that it's possible the system will reply, "sorry, there's none available now".

John Carter had some objections and a quite interesting take: “NO! NO! NO! NO!

“By the time malloc fails, the system is just plain fubared. I hate deeply convoluted error handling code that checks the return code of malloc....

“* Then goes through horrible contortions to attempt to recover sanity.
“* And has _never_ been tested.
“* And God help it if it either directly, or anywhere else on it's call graph, invokes malloc again! Which it will if it tries to printf anything!

“So what is the correct solution?

“Wrap malloc in a function that checks the return code, and if it fails uses statically pre-allocated space to log a stack trace, then system error and reboot!

“But that sounds just like what you are ranting against! Hear me out to find the correct solution.

“What has gone wrong is you have exceeded the design load for that system. Suppose you were told to design a ferry to carry cars. But you couldn't get management to decide how many cars it should carry. So you designed it so you could load any number of cars.

“When the ferry starts sinking and taking on water, on every gunwale there is a "water coming in" detector. Attached to each of those detectors is an amazingly complex one-off uniquely-designed gadget which you can't test without sinking a loaded ferry (for many highly customised loads), that flings the last few cars into the water!

“One solution is to tell management to get their butts into gear and actually _decide_ on the designed for carrying capacity of the ferry. And then design mechanisms to only allow that many on.

“Your customers are OK with knowing that their system can only handle a finite load.

“The other solution is to have a "Plimsoll Line", a safety margin and to keep loading until you have reached that level of resource usage and then stop loading. See http://en.wikipedia.org/wiki/Waterline .

“ Wire into your malloc wrapper a counter of how much space you have used. Once you approach your safety margin, stop loading. Shed load at the periphery. And then LOAD TEST your system. Throw way more load at it than it can possibly handle. It should gracefully refuse to accept more load, or throttle the load to what it can handle.

“Resource depletion is _never_ an unexpected condition. But once you have hit resource depletion you

“* lack resources to handle it gracefully. Doh!
“* you are deep deep into the nested logic of the system.

“Resource depletion needs to be handled at a _much_ higher level and at the peripheries of the system.

“Shedding load once you are halfway through processing is always complex, processing intensive, error prone, and often down right defective. (You need to carefully back out of every resource acquisition, and reassemble every invariant on every possible error path or you have a resource leak or a defect.)

“Hmm. Perhaps we should listen a bit more to the Environmentalists. After all, resource depletion is _never_ an unexpected condition...but once you are in it, you lack resources to handle it gracefully.”

Joke for the Week

NEWS BULLETIN: Saying it will improve the education of children who have grown up immersed in computer lingo, the school board in San Jose, Calif., has officially designated computer English, or "Geekonics," as a second language. "This entirely reconfigures our parameters," Milton "Floppy" Macintosh, chairman of Geekonics Unlimited, said after the school board became the first in the nation to recognize Geekonics.

"No longer are we preformatted for failure," Macintosh said during a celebration that saw many Geekonics backers come dangerously close to smiling. "Today, we are rebooting, implementing a program to process the data we need to interface with all units of humanity." Controversial and widely misunderstood, the Geekonics movement was spawned in California's Silicon Valley, where many children have grown up in households headed by computer technicians, programmers, engineers and scientists who have lost the ability to speak plain English and have inadvertently passed on their high-tech vernacular to their children.

While schools will not teach the language, increased teacher awareness of Geekonics, proponents say, will help children make the transition to standard English. Those students, in turn, could possibly help their parents learn to speak in a manner that would lead listeners to believe they have actual blood coursing through their veins. "Bit by bit, byte by byte, with the proper system development, with non-preemptive multitasking, I see no reason we can't download the data we need to modulate our oral output," Macintosh said.

The designation of Geekonics as a language reflects a growing awareness of our nation's lingual diversity, experts say. Other groups pushing for their own languages and/or vernaculars to be declared official viewed the Geekonics vote as a step in the right direction.

"This is just, like, OK, you know, the most totally kewl thing, like, ever," said Jennifer Heather Notat-Albright, chairwoman of the

Committee for the Advancement of Valleyonics, headquartered in Southern California. "I mean, like, you know?" she added.

They're happy in Dixie. "Yeee-hah," said Buford "Kudzu" Davis, president of the Dixionics Coalition. "Y'all gotta know I'm as happy as a tick on a sleeping bloodhound about this." Spokesmen for several subchapters of Dixionics -- including Alabonics, Tennesonics and Louisionics -- also said they approved of the decision.

Bill Flack, public information officer for the Blue Ribbon Task Force on Bureaucratonics said that organization would not comment on the San Jose vote until it convened a summit meeting, studied the impact, assessed the feasibility, finalized a report and drafted a comprehensive action plan, which, once it clears the appropriate subcommittees and is voted on, will be made public to those who submit the proper information-request forms.

Those involved in the lingual-diversity movement believe that only by enacting many different English languages, in addition to all the foreign ones practiced here, can we all end up happily speaking the same boring one, becoming a nation that is both unified in its diversity, and diversified in its unity. Others say that makes no sense at all. In any language.

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.

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.