ECE 471 – Embedded Systems
Lecture 26

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

9 November 2018

Announcements

- HW#8 was due
- HW#9 was assigned, due a week from Wed (before Thanksgiving)
- Project topics due today
- Midterm Upcoming, Review next Week
- No class Monday (Veterans' Day)

HW#9

- Re-use i2c display code from earlier homework
- Re-use temp code (either TMP36 or the 1-wire)
- Display the temperature on LED.
- 4 cases
  - 00.0 to 99.9
  - -99 to 0
  - 100 to 999
  - Error
- Will require you making numbers if you haven't already,

but otherwise should be straightforward re-use of old code.

Software Bugs

- Not all bugs are security issues
- Coding bugs can have disastrous effects

Spacecraft

- Mariner 1 (1962) – rocket off course due to mis-transcribed specification into FORTRAN, missing overbar
- Apollo 11 (1969) – landing on moon.
  - 36k ROM (rope), 2k RAM, 70lbs, 55W, 5600 3-input NOR
  - Processor normally loaded with 85% load. DELTAH program run which take 10%. But buggy radar device was stealing 13% even though in standby mode.

- Multiple 1202 overload alarms
- Mini real-time OS with priority killed low-priority tasks so things still worked.

- Ariane 5 Flight 501 (1996) – famous. $370 million.
  - Old code copied from Ariane 4. Horizontal acceleration
  - Could not trigger on Ariane 4 (accel never that large)
  - Could trigger on more powerful Ariane 5
  - Conversion from 64-bit float to 16-bit signed int overflowed. Trap
  - Primary guidance computer crashed
  - Secondary computer, but ran same code, crashed

  - Sent debug messages after crash, autopilot read those as velocity data
  - Destructed 37s after launch
  - Written in ADA
- NASA Mars Polar Lander (1999)
  - likely mistook turbulence vibrations for landing and shut off engine 40m above surface
- NASA Mars Climate Orbiter
  - ground software using lbf (pound/foot) units, craft expecting Newtons
- NASA Mars Spirit rover (2004)

- ◦ temporarily disabled due to too many files on flash drive
- ◦ Constantly rebooting
- ◦ Radio could understand some commands directly, could reboot with flash disabled.
- ◦ Fixed when deleted some unneeded files.
- ◦ Eventually reformat.
- ◦ Issue is 90 day design period, lasted years (until 2010)

- ExoMars Schiaparelli Lander (2016)
  - ◦ Bad data to inertial measurement unit for 1 second
  - ◦ thought this meant it was below ground level, released

parachute when still 3.7km up.

- Had valid data from radar

Medical Example

- Therac-25 radiation treatment machine, 1985-1987
- 6 accidents, patients given 100x dose. Three died High power beam activated w/o spreader too. **Older machines had hardware interlock**, this one in software. Race condition. If 8-bit counter overflow just as entering manual over-ride, it would happen.
- Triggering the bug
  - To trigger, had to press X (mistake), up (to correct), E (to set proper) then "Enter" all within 8 seconds.

This was considered an improbable series of keypresses.

- ◦ This missed during testing as it took a while for operators to get used to using machines enough to type that fast.
- ◦ Used increment rather than move to set flag, this meant sometimes it wrapped from 255 to 0, disabling safety checks
- ◦ Written in Assembly Language

Things that went wrong with design

- ◦ Software not independently reviewed
- ◦ No reliability modeling or risk management

- Something wrong: Printed “MALFUNCTION” and error number 1 to 64 which was not documented in manual. Press P to clear.
- Operators not believe complaints from patients.
- The setup was not tested until after it was installed at hospital.
- cut-and-pasted software from earlier model that had hardware interlocks
- Concurrent (parallel) operation with race conditions

Medical Response

- IEC 62304 – medical device software – software lifecycle
  - Quality management system – establish the requirements needed for such a device, then design methods to be sure it meets these
  - Avoid reusing software of unknown pedigree (don't just cut and paste from stackoverflow)
  - Risk management – determining what all the risks involved are, then determine ways to avoid or minimize them
  - Software safety classification

Class A: no injury possible

Class B: Nonserious injury possible

Class C: serious injury or death possible

Software sorted into these areas. Class A do not require the same precautions as the others.

Financial

- Knight Capital. Upgrade 7 of 8 machines, missed last. Re-used a flag definition with new software. Caused massive selloff, $440 million

Power

- 2003 Blackout
  - Power plant fail. Cause more current down transmission lines in Ohio. Heat, expand, touch tree, short out.
  - Race condition in Unix XA/21 management system, so alarms not go off
  - Eventually primary system fail as too many alarms queue up
  - Backup server also fail

- ◦ During failure, screens take 59s (instead of 1s) to update
- ◦ Blackout of most of NY and a lot of north east.

Good Design Practices

Code Safety Standards

- Avionics: DO-178C (1992 for B)
- Industrial: IEC 61508 (1998)
- Railway: CENELEC EN 50128 (2001)
- Nuclear: IEC 61513 (2001)
- Medical: IEC 62304 (2006)
- Automotive: ISO 26262 (2011)

Other notes

- Top down vs Bottom up Design
  Spec out whole thing and how they work first
  Start with core part and just keep adding to it until it works

- Requirements/Specifications?

Writing Good (Embedded) C Code

- Various books. Common one: MISRA: Guidelines for the Use of the C Language in Critical Systems
- Comment your code!
- Strict, common code formatting (indentation)
- More exact variable types (int32_t not int) Size can vary on machine, and on operating system
- Subset to avoid undefined behavior

- Tool that enforces the coding standards
- Good to write safe code even if it isn't meant for a safe application. Why? Good practice. Also who knows who or when your code might be copied into another project.

Good Test Practices

- Unit testing
- Test Driven Development – tests written before the code happens, needs to pass the tests before done
- Fuzzing
- Device Hardening?

Good Documentation Practices

- Comment your code
- Write documentation! Make sure it matches code! There are some tools that can auto-generate documentation from special code comments
- Use source control (git, subversion, mercurial)
- Use good commit messages in your source control