Status of safety-critical FOSS

January 31, 2016

Jeremiah C. Foster
(Carbon Based Life Form)
Twitter: @miahfost

Overview

- Safety-critical for FOSS folks
- What are the challenges?
- Can FOSS become "safety-critical"?
- What are the implications of using copyleft licenses in safety-critical software?
- What is the roadmap?
- What resources to engage in the discussion is available?

I'd like very much to engage the audience on the subject matter, particularly on the dynamic of copyleft and how it may provide a greater level of transparency than proprietary software not just into the actual code but in the entire process and supply chain and hardware.

What is safety-critical?

Thus spake Wikipedia;

Born from industries with a need for process control, it’s a “basic functional safety standard applicable to all kinds of industry.” Functional safety is part of the overall safety relating to equipment and to the control of equipment which depends on the correct functioning of the safety-related system.

Functional safety relies on **active** systems.

- The detection of smoke by sensors and the ensuing intelligent activation of a fire suppression system
- The activation of a level switch in a tank containing a flammable liquid, when a potentially dangerous level has been reached, which causes a valve to be closed to prevent further liquid entering the tank and thereby preventing the liquid in the tank from overflowing.

"When designed and implemented correctly, software is often the first, and sometimes the best, hazard detection and prevention mechanism in the system."

-- NASA

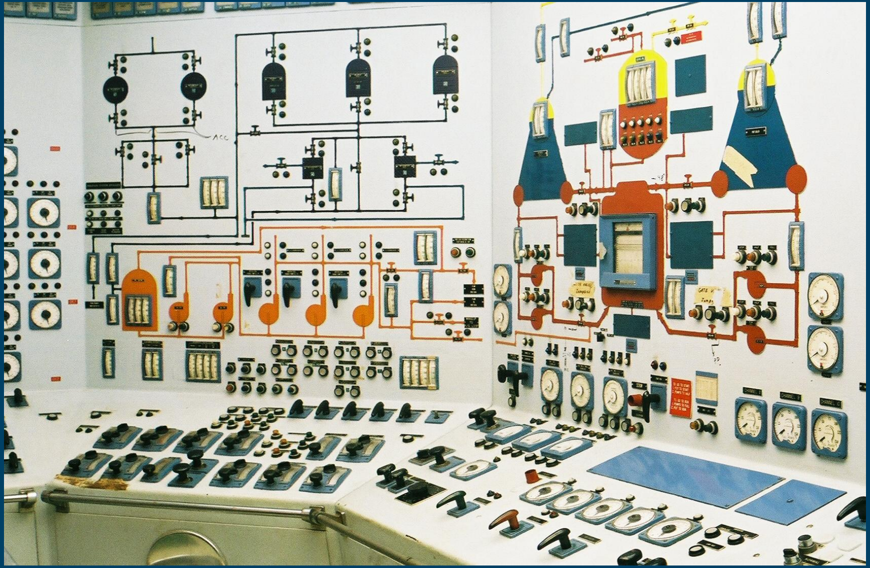


Photo credit: Wikipedia https://commons.wikimedia.org/wiki/File:Nuclear_Ship_Savannah_-_Reactor_Control_Room_-_Center_and_Left_Panels.jpg

Standardization, there’s lots of it

IEC 61508	• One standard to rule them all • "Making electrotechnology work for you"
ISO 26262	• Functional safety features form an integral part of each automotive product development phase, ranging from the specification, to design, implementation, integration, verification, validation, and production release.
IEC 61513	• Provides requirements and recommendations for the instrumentation and control for systems important to safety of nuclear power plants.
IEC 62061	• Machinery specific implementation of IEC 61508

Is it even possible to ‘certify’ GNU/Linux at a safety-critical level?

Safety-critical software is usually reasoned about in Safety Integrity Levels, or SIL. Systems get certified at a given SIL. A SIL rating describes the chance of failure, or the “Probability of Failure on Demand” (PFD)

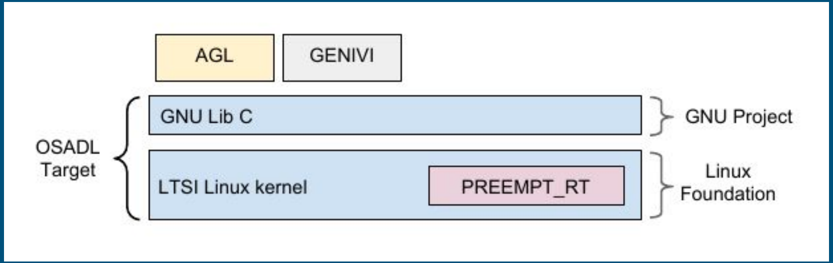
Safety Integrity Level	Safety	Probability of Failure on Demand	Risk Reduction factor
SIL 4	> 99.99%	0.001% to 0.01%	100,000 to 10,000
SIL 3	99.9% to 99.99%	0.01% to 0.1%	10,000 to 1,000
SIL 2	99% to 99.9%	0.1 to 1%	1,000 to 100
SIL 1	90% to 99%	1 to 10%	100 to 10

OSADL approach -- SIL2LinuxMP

A safety-critical system certified for ASIL B use:

“The SIL2LinuxMP project aims at the certification of the base components of an embedded GNU/Linux RTOS running on a single-core or multi-core industrial COTS computer board. Base components are boot loader, root filesystem, Linux kernel and C library bindings to access the Linux kernel.“

- This might be a Debian based system, though Yocto has been discussed
- No userland components except for some small system tools for inspection, etc.
- Build toolchain *is* included



An example of what the stack might look like and where the current areas of responsibility a delegated.

Linux Foundation has raised funds for bringing in PREEMPT_RT into mainline and OSADL will join the LF as a “gold” member of the RTL group.

Please see: https://wiki.linuxfoundation.org/realtime/start

http://www.linuxfoundation.org/collaborate/workgroups/real-time

https://www.osadl.org/Single-View.111+M5c0795180e5.0.html

Nicholas McGuire's work in OSADL

Talk from June 2015

- The main reason for looking at linux is that most safety systems do not have multi-core support. You cannot have concurrent kernel process in multiple cores with the tiny, safety oriented Real Time OSes. Also, surprisingly, RTOSes don't seem to have the same quality of the security standards, there Linux is advanced.
- The Linux system that OSADL is looking at roughly 350,000 lines of code
- Linux is surprisingly close to POSIX which turns out to be important because the key issue is not necessarily "perfect" code, it's the design process and the ability to fix errors in the design phase, your design process is key.
- Dangerous to drive safety qualification by features and functionality. Select qualifiable components instead, a bit of a backwards process.
- Because the LTSI kernel change rates are relatively low they are good targets for qualification
- Known problems can be addressed - Open Source / open processes and a responsive community is why we can address these issues in GNU/Linux.
- ~1000 patches from SIL2linuxMP over one year
- Linux kernel process almost exactly fits ISO 9001 (Quality Management)

Concerns

GPLv3, § 6

Installation information requires disclosure of secret encryption keys used to sign the boot image.

Inability to comply

Embedded systems will not (cannot?) comply due to safety requirements. (End users should not be modifying software on safety-critical systems due to systemic and potentially catastrophic risk.)

How to resolve?

- Exception to § 6
- “Regulation”
- Use of GPLv2 software, i.e. older versions
- “Marketing and Education”

Premise: Copyleft may be an effective mechanism to ensure a necessary level of transparency as well as adequate protection for source code that is expensive to produce and maintain