Outline

- Context
- Process
- Conclusions
Goal of SIL2LinuxMP

- Generic qualification approach
- Suitable for up to SIL2 (IEC 61508 Ed 2)
- Support multicore systems
- Mainline kernel + glibc + tools
- Methods suitable for pre-existing SW intensive systems
SIL2LinuxMP Context

IEC 61508 Ed2
ISO 26262
EN 50128
IEC 17799
IEC 15408
IEC 62433
EN 50159
POSIX 1003.1
POSIX 1003.13
susV3
The Goal

Outline

Context

GNU/Linux for safety-related systems - SIL2LinuxMP

Nicholas Mc Guire <safety@osadl.org>
Basic constraints

- Minimize kernel $\iff$ follow mainline
- Minimize safety related runtime env
  - glibc
  - busybox runtime environment
  - Handle cgroups "manually" $\iff$ minimal launcher
- Compliant development of safety related applications
- Push the full-featured (non-safe) OS into a container
- Minimize/control sharing of resources between safe/non-safe tasks
Arch 4 - prototype architecture

SIL 2 base-system

Monitoring

Admission

busybox

glibc

SIL 2 minimal container

Safe-App A

glibc

SIL 2 minimal container

Safe-App B

glibc

SIL 0

Debian container

CPU 0

Rambank0...n

CPU 1

Rambankn+1...m

CPU 2

Rambankm+1...l

CPU 3

Rambankl+1...k
Arch 4 - prototype architecture

think of it as a "distributed system on one chip"
Selection has been formalized in the context of 61508-1 Ed 2 as Clause 7.X "E/E/PE safety-related software element selection" - pending review by TueV Rheinland.
Adjusted software DLC

Figure 6 – Software systematic capability and the development lifecycle (the V-model)
Adjusted software DLC
Example: Isolation Techniques

Available technologies to improve non-interference

- Control Groups
- Namespaces
- Separate filesystem (images/media)
- Replicated glibc/busybox
- Limit system calls (seccomp)
- Real devices managed by core-system
- PALLOC - partitioning allocator
- ABI diversity

Functionality + level of assurance -> safety functional capability
Big picture of DLC/SLC

Target System
- Use-Case
- DRM
- HAZOP/FMEA
- Overall safety requirements

DLC/SC
- 7.2 Concept
- 7.3 Scope
- 7.4 Hazard/Risk Analysis
- 7.5 Safety func. requirements
- 7.6 Allocation
- 7.7 X Selection

Pre-Existing Elements
- candidate elements
  -> safety contribution potential
- safety potential dependency tree
- potential architecture selection of intended safety functions

First system concept consolidation phase -- preliminary architecture

Partitioning of safety functions

Allocation of elements to partitions: layered protection architecture

Conceptual ESD of failure model

Methods of analysis

PRA

Certi Data Package

GNU/Linux for safety-related systems - SIL2LinuxMP
Nicholas Mc Guire
<safety@osadl.org>

Outline

Context
Conclusions

- If you want to utilize FLOSS -> fix the processes first
- IEC 61508 was not really conceived with selection as primary strategy in mind - but it is doable.
- IEC 61508 is robust enough to provide a solid foundation for formalizing element selection (Route 3_S) as primary strategy
- The process adjustments are in review (TueV Rheinland) ... lets see
- Based on the final processes the method set will be selected
- Applying this to GNU/Linux RTOS will not be trivial - but looks doable
Thanks!

http://www.osadl.org/SIL2