Space Grade Linux

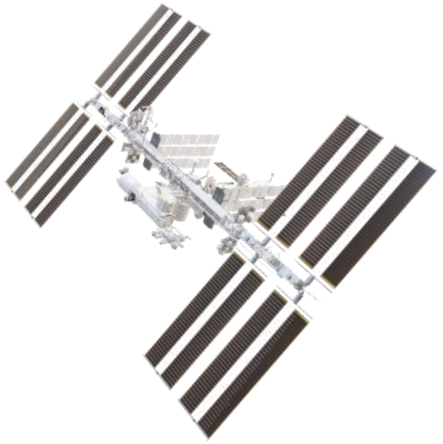
FSW25 Introduction - Michael Monaghan, NASA Goddard Space Flight Center

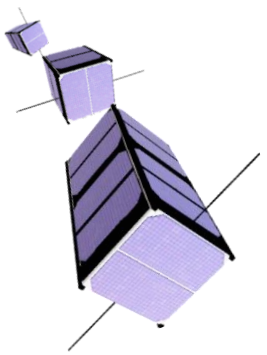
Why use Linux?

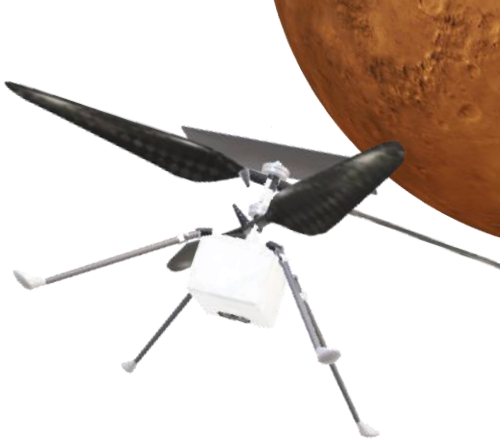
Space Industry is adopting Linux, **Fast!**

- Free and open source
- Abundant workforce expertise
- Unparalleled software ecosystem
- Strong embedded hardware support
- Extensively reviewed and tested
- Adaptable to real-time or performance-centric workloads

Who is using Linux?







Space

- NASA
  - Ingenuity
  - International Space Station Experiments
  - Starling
  - Countless CubeSats
- SpaceX
  - Falcon 9
  - Dragon
  - Starlink (>30k Linux nodes!)

Earth

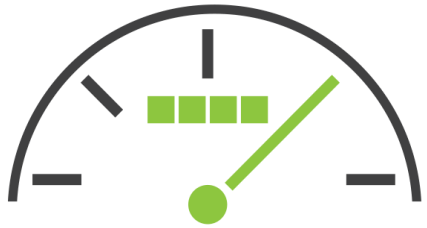
- Automotive
  - Major Autopilot/Advanced driver assistance platforms
- Telecom
  - Majority of LTE and 5G chipsets.
  - LTE and 5G infrastructure

…

What is SGL?

Space Grade Linux is a collaborative, open-source project bringing together space agencies, industry, and academia to establish a trusted ecosystem of Linux-based, open-source software for mission critical spacecraft operations.

Inspiration



AUTOMOTIVE GRADE LINUX

- Revolutionized automotive software.
- Brought competitors together to collaborate on common code.
- Quality of life improvements beyond the automotive industry.

Purpose of SGL

Reduce barrier to entry, cost, and error.

- Stop reinventing the wheel!
- Reuse software and collaborate.
- Minimize time to launch.
- Condense required expertise.
- Tap into collective knowledge.

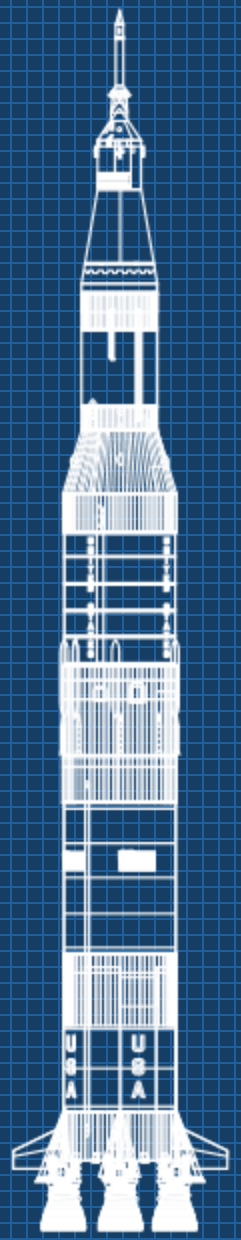
Facilitate modern workloads in space

- Prepare for next-gen high performance spacecraft processors.
- Tap into unrivaled performance and software support offered by Linux
- Support nearly all AI/ML frameworks out of the box.
- Address challenges presented by spacecraft environments.

Inspire Confidence in Linux

- Establish a trusted ecosystem.
- Demonstrate reliability outside terrestrial applications.
- Serve as a baseline for future certification.

SGL Blueprint



Your Mission (Software Payload)

- *Mission Specific Flight Software (cFS Apps, FPGA design, drivers…)*

Space Grade Linux

- Userspace
  - Software Framework (e.g. cFS)
  - Linux-specific framework apps.
  - Daemons and tools (systemd, udev…)
  - Dependencies and libraries (busybox, glibc…)
  - Dependency and library config
  - Distribution policy and config
- Board Support Package
  - Kernel & Config
  - Drivers
  - Bootloader (e.g. u-boot)
  - Bootloader additions (resilient boot flow, update mechanisms)
  - Hardware Platform Support
- Baseline Certification
  - Kernel + Userspace
  - Security hardening

How?

Yocto Project® / OpenEmbeded

- Build system and interface to contribute.
  - Custom kernels, bootloaders, board support packages, software packages, distribution configurations, etc.
- Vast preexisting software and hardware support.
  - layers.openembedded.org
- Widely used across industry.

Focus Areas

Software

Yocto Linux Disto tailored to space.

Stop re-inventing the wheel.

Reduce time to launch.

Establish a common code base.

Support cutting edge open-source software, such as for AI/ML.

Hardware

Yocto BSP for common space processors.

Facilitate high-performance mission processing needs.

Target next-gen space processors.

Meet aggressive SWaP constraints.

Safety

High design assurance levels.

Agency certifiable for Mission Critical Ops.

Establish a track record.

System security and integrity by design.

Protection against malicious actors.

Status

Planning phase

- Discussions
- Knowledge sharing

Initial pull requests

- https://github.com/elisa-tech/meta-sgl

SGL Software Constellations

- Akin to flavors, spins, editions, etc.
- Configurations to support different use cases.
  - Flight software frameworks.
    - cFS, Fprime, Space ROS, etc.
  - Distribution configurations.
    - Systemd?
    - Minimal or traditional userspace?
    - glibc or musl?
    - Dynamic or static linking?

cFS-One Constellation

(work in progress)

A middle ground between RTOS and classic Linux

- cFS as init/PID 1.
- Bare bones.
  - Minimal kernel config.
  - Easier certification/verification.
- Minimal footprint.
- Safely extensible.
  - Common Libraries, executables.
  - Container runtimes, Virtual machines?

Questions and Comments