Two Years Experience of Industrial-grade Open Source Base Layer Development and its Future

Yoshitake Kobayashi, Technical Steering Committee Chair
Open Source Summit Japan 2018
June 20-22, 2018
What is CIP?
What is CIP?

• One of the most conservative open source projects in the Linux Foundation

• One of the most important projects for our civilization
What is CIP?

• One of the most conservative open source projects in the Linux Foundation

• CIP aims to
  • Provide an open source base layer for CIP related embedded systems
  • Work closely with the upstream community

• CIP does not aim to
  • Create a new Linux distribution
Our Civilization runs on Linux
Transport

- Rail automation
- Vehicle control
- Automatic ticket gates

Energy

- Power Generation
- Turbine Control

Industry

- Industry automation
- CNC control
- Industrial communication

Others

- Healthcare
- Building automation
- Broadcasting
There are issues to be solved…
A Power Plant System:
25-60 years products life-cycle
Very reluctant to perform product updates and upgrades of hardware and base software platform
Power Plant Control Example

3 – 5 years development time

0.5 – 4 years customer specific extensions

6 – 8 years supply time

15+ years hardware maintenance after latest shipment

20 – 60 years product lifetime
CIP started from April 2016
2\frac{1}{4} \text{ years}
Possible product life time and CIP current age.

60 years product life time

Things change a lot during this product life time
Industrial IoT: Edge and Fog Computing

Functionality is moving from the cloud to the “Edge”

- Increasing number of networked industrial-grade devices
- Security management requires harmonized software landscape

Application examples on IIoT infrastructure

- Plant analytics
- SCADA functionality
- Plant (device) mgmt.
- Local / real-time analytics
- IoT Gateways

Data collection
- Pre-processing
- Sensor / actor connectivity

IoT: Internet of Things
IIoT: Industrial IoT
SCADA: Supervisory Control And Data Acquisition

Open Source Summit Japan 2018
The Problems we face ...

• The systems that support our modern civilization need to **survive for a VERY LONG TIME.** Until now the corresponding industrial grade super long term maintenance has been **done individually by each company.**

• These systems not only have to survive for a long time, they must be “**INDUSTRIAL GRADE**” (robust, secure and reliable). And at the same time the industry will also need to **catch up with the latest technology trends.**
The Solutions we need ...

- **We need a Collaborative framework** to maintain the same open source based system for many, many, many years to keep it secure, robust and reliable.

- **AND most importantly, we need to do this collaboratively in the upstream communities**, not locally.
CIP is our solution...

Establishing an **Open Source Base Layer** of industrial-grade software to enable the use and implementation of software building blocks for Civil Infrastructure Systems

[https://www.cip-project.org/](https://www.cip-project.org/)

since April 2016
CIP timeline
The backbone of CIP are the member companies

Member companies

Open source projects (Upstream work)

Budget

Optional: funding of selected projects

Developers, maintainers

Contribution & usage / integration

CIP source code repositories

CIP Super Long Term Support Project
What is CIP, again?
What is “Open Source Base Layer (OSBL)”?

- OSBL is a set of industrial grade core open source software components, tools and methods

  - Open source based reference implementation
  - Start from a minimal set for controllers in industrial grade systems

Non-CIP packages
- Linux distribution (e.g. Debian) may extend/include CIP packages.

CIP Reference
- Filesystem image with SDK (CIP Core packages)

CIP SLTS Kernel

CIP Reference Hardware

User space
CIP activities and status
Scope of activities

User space

- App container infrastructure (mid-term)
- App Framework (optionally, mid-term)

Middleware/Libraries

- Domain Specific communication (e.g. OPC UA)
- Shared config. & logging
- Multimedia
- Safe & Secure Update
- Monitoring
- Security
- Real-time support
- Real-time / safe virtualization
- CIP Core Packages

Linux Kernel

- Super Long Term Supported Kernel (STLS)

On-device software stack

Tools

- Build environment (e.g. bitbake, dpkg)
- Test automation
- Tracing & reporting tools
- Configuration management
- Device management (update, download)
- Application life-cycle management

Concepts

- Functional safety architecture/strategy, including compliance w/ standards (e.g., NERC CIP, IEC61508)
- Long-term support Strategy: security patch management
- Standardization collaborative effort with others
- License clearing
- Export Control Classification

Kernel space

- Secure & Update
- Monitoring
- Security
- Real-time support
- Real-time / safe virtualization

Product development and maintenance

Open Source Summit Japan 2018 23
CIP Activities

1 Kernel maintenance
   • The first action taken by the CIP project is to select and maintain Linux kernels for very long time. To achieve goal a group of experts has been assigned.

2 Testing
   • PREEMPT_RT patches are added to the CIP kernel

3 Testing
   • Civil infrastructure industry has high stability, reliability and security standards in order to ensure continuity of safety critical systems. The CIP Testing project has been formed to address this reality. So far the efforts are focused on testing the CIP kernel. In the future they will be extended to the complete CIP platform.

4 CIP Core
   • This project focus to create reference minimal file system images that allow testing the CIP Core packages: a set of industrial-grade components that require super long-term maintenance.
CIP SLTS Kernel development

CIP SLTS (linux-4.4.y-cip), Maintenance period 10 years and more

- Official CIP SLTS kernel tree based on linux-stable.git
- Maintainer: Ben Hutchings, from Codethink
CIP SLTS Kernel development

• Kernel maintenance policy
  • [https://wiki.linuxfoundation.org/civilinfrastructureplatform/cipkernelmaintenance](https://wiki.linuxfoundation.org/civilinfrastructureplatform/cipkernelmaintenance)
  • Follow the stable kernel development rule as the basis
  • Feature backports are acceptable
    • **CIP has “Upstream first” policy**
      • All features has to be in upstream kernel before backport to CIP kernel
  • Validation will be done by CIP test infrastructure and/or members
Out-of-tree drivers

• In general, all out-of-tree drivers are unsupported by CIP

• Users can use CIP kernel with out-of-tree drivers
  • If a bug is found in such a modified kernel, users will first demonstrate that it exists in the CIP kernel source release in order for the CIP maintainers to act on it.
CIP SLTS Kernel development

- First CIP kernel (Linux 4.4.42-cip1) released on Jan 13th, 2017
- The latest CIP kernel (Linux 4.4.138-cip25) released on June 20th, 2018
- CIP kernel release period
  - Ben Hutchings, from Codethink, releases a kernel version approx. every 4 to 6 weeks. Depends on the amount and relevance of upstream patches
    - Collaborates in the 4.4 LTS process directly
- CIP kernel release cycles are 1-2 month
- CIP review patches and run tests
CIP SLTS Kernel development

• What’s new in CIP kernel (4.4.138-cip25)
  • Add video (DU, VSP, VIN), CAN, PCIe, audio, PWM, thermal, and clocksource (CMT) support for R8A7743 and iWave board
  • Bug fixes for the sgtl5000 codec driver used on the iWave board
  • Update shmobile_defconfig and multi_v7_defconfig to enable more of the above drivers
  • Merge fixes from stable versions 4.4.131-4.4.138 inclusive
  • Fix stable regressions and add missing dependencies

• Other backported features on 4.4.y-CIP
  • Kernel Self Protection Project related features
    • Address Space Layout Randomization for user space process (ASLR)
    • GCC’s undefined behaviour Sanitizer (UBSAN)
    • Faster page poisoning
  • Board support patches for Siemens IoT2000 series
4.4-stable review patch. If anyone has any objections, please let me know.

------------------
From: Christoph Hellwig <hch@lst.de>
commit f507b54dccf8000c517d740bc45f20c74532d18 upstream.

The job structure is allocated as part of the request, so we should not free it in the error path of bsg_prepare_job.

Signed-off-by: Christoph Hellwig <hch@lst.de>
Reviewed-by: Ming Lei <ming lei@redhat.com>
Signed-off-by: Jens Axboe <axboe@kernel.dk>
Signed-off-by: Greg Kroah-Hartman <gregkh@linuxfoundation.org>

---
block/bsg-lib.c | 1 - 1 file changed, 1 deletion(-)

--- a/block/bsg-lib.c
+++ b/block/bsg-lib.c
@@ -147,7 +147,6 @@ static int bsg_create_job(struct device
 failjob_rls_rqst_payload:
     kfree(job->request_payload.sg_list);
 failjob_rls_job:
-    kfree(job);
     return -ENOMEM;
}
CIP launch a CIP kernel team

• CIP kernel team structure
  • Maintainer / Mentor
  • Developers
• Roll of CIP kernel team
  • Review patches for stable kernel
  • Test
  • Feedback to upstream

• Upstream first
CIP kernel team

On Fri, 2018-06-15 at 13:24 +0000, Daniel Sangorin wrote:

Hi Greg,

```
/* Intel-defind CPU features, CPUID level 0000000001h (ex), word 4 */
---
# a/arch/x86/include/asf/fpu/Internal.h
+++ b/arch/x86/include/asf/fpu/Internal.h
@@ -58,7 +58,7 @@ extern u8 sfpu_get_supported_features_
 */
static __always_inline __pure bool use_eager_fpu(void)
{
    return static_cpu_has_safe(x86_FEATURE_EAGER_FPU);
    return true;
}
```

Since this function returns always true then we can remove the code depending on lazy fi.

Actually this has already been done in *x86/fpu: Remove use_eager_fpu()*

Ref: `https://patchwork.kernel.org/patch/9363883/`

```
static void __init fpu_init_parse early_param(void)
{
    if (cmdline_find_option_bool(cmdline, "eagerfpu-off")) {
        eagerfpu = DISABLE;
        fpu_clear_eager_fpu_features();
    }
}
```

Since this patch removes the kernel boot parameter "eagerfpu", maybe we should remove it from the documentation.

```
*eagerfpu"
```

This has also been done by commit *"x86/fpu: finish excising 'eagerfpu'"

Ref: `https://patchwork.kernel.org/patch/9380673/`

I will try backporting those patches unless anyone has any objections.

This does seem like a good idea--there is quite a bit of dead code left
and it may be hard to backport any further bug fixes in this area
without that removal.

Ben.

---
CIP kernel team : Recent activities for 4.4-stable

- RE: [PATCH 4.4 10/24] x86/fpu: Hard-disable lazy FPU mode
  - https://lkml.org/lkml/2018/6/15/6

- Backport of pipe.c patch to 4.4.y that fixes LTP fcntl35 test error

- Re: [PATCH 4.4 00/92] 4.4.133-stable review
  - https://www.spinics.net/lists/kernel/msg2812317.html

- Please apply this XFS patch on 4.4
Next SLTS kernel version

Next CIP SLTS kernel

Mainline 4.4

Stable (linux-stable)

Backported patches
Maintained by Ben Hutchings

CIP SLTS (linux-4.4.y-cip)

Stable (linux-stable-x.y)

NEXT CIP SLTS ( ? )

Approx. 2-3 years

CIP will pick up next version

Stop backporting. Focus to security fix only

Feature backports

Backported patches

Open Source Summit Japan 2018 34
CIP next SLTS kernel version

- CIP plans to pickup next SLTS kernel version in 2018 or early next year
  - If everything goes fine, 4.20 (or 5.0) will become next CIP SLTS kernel (still tentative)
  - Kernel version alignment with other project is a key
    - LTS / LTSI / AGL / Debian

"Using Linux for Long Term - Community Status and the Way We Go,"
Tsugikazu Shibata, NEC, Open Source Summit Japan 2018
CIP SLTS real-time support

- CIP is a Gold Member of the Real Time Linux Project
  - Work together with the RTL Project
  - Daniel Wagner from Siemens is working to become the maintainer of 4.4.y-stable-rt, the base version of the CIP Kernel.

- More information
  - [https://wiki.linuxfoundation.org/realtime/rtl/start](https://wiki.linuxfoundation.org/realtime/rtl/start)
CIP SLTS Real-time support

CIP SLTS Real time kernel is available

- CIP kernel tree based on linux-stable-rt and patches from CIP SLTS
  - Maintainer: Daniel Wagner, from Siemens AG
- Validated by CIP
CIP SLTS real-time support

- Daniel is now maintaining 4.4-stable-rt
  - The latest 4.4-rt is 4.4.137-rt154 released on June 15th
- The latest CIP RT kernel 4.4.130-cip23-rt18 released on May 28th
CIP testing

Milestones of CIP testing and current status

1. Board at desk - single dev
   • A setup that allows a developer to test the CIP kernel on selected CIP hardware platforms connected locally to her development machine using kernelCI tools.

2. CIP kernel testing
   • Test the CIP kernel on a regular basis and share the results with other CIP community members.

3. Define kernel testing as a service within CIP
   • Define the testing environment within CIP assuming that, in some cases, some members may share the tests, test results or laboratories while others may not.

4. From kernel testing to system testing
   • Once the testing environment is ready and works for the kernel, explore how to extend it to the entire CIP platform.

https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting
CIP testing

CIP Testing project
(https://wiki.linuxfoundation.org/civilinfrastructureplatform/ciptesting)

• B@D designed to:
  • Test Linux kernels and base systems locally.
  • On hardware connected to your dev machine.

• B@D features
  • Based on kernelci.org
  • Linux and Windows 10 as Host OS supported.
  • Shipped as VM and Vagrant based environments.
  • Results and logs sharing capabilities.

• Check the source code involved
  • https://gitlab.com/cip-project/cip-testing/board-at-desk-single-dev/tree/master
The latest status for CIP testing

1. Updated to include the latest kernelCI
2. Supporting Renesas iw20gm
3. Shared testing results
   • https://lists.cip-project.org/pipermail/cip-testing-results/

• Next steps
  • Deployment through containers
  • Building a centralized service to collect results
    • https://kernelci.ciplatform.org/
    • https://lava.ciplatform.org/
  • Collaboration with other testing effort
  • Increasing the test coverage
Current status of the Base layer development

1. Define an initial component set
2. Define component version
3. Contribute to upstream projects
4. Start maintenance for SLTS
**CIP Core Packages**

**Debian as a CIP primary reference distribution**

- What does the primary distribution means?
  - CIP will select CIP Core package from Debian packages
  - CIP would like to work with Debian community

- Status
  - Supporting to solving security related issues
    - Adding support for staging repositories to security-master
      - [https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=817286](https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=817286)
  - Status: Paperwork is ongoing

- CIP joined **Debian-LTS as Platinum level**
4 Initial focus for CIP Core Packages

An example of minimal package set for CIP base layer

Candidates for initial component set

- **Kernel**
  - Linux kernel + backported patches
  - PREEMPT_RT patch
- **Bootloader**
  - U-boot
- **Shells / Utilities**
  - Busybox
- **Base libraries**
  - Glibc
- **Tool Chain**
  - Binutils
  - GCC
- **Security**
  - OpenSSL

Keep these packages for Reproducible build

- **Flex**
- **Bison**
- **autoconf**
- **automake**
- **bc**
- **bison**
- **Bzip2**
- **Curl**
- **Db**
- **Dbus**
- **Expat**
- **Flex**
- **gawk**
- **Gdb**
- **Git**
- **Glib**
- **Gmp**
- **Gzip**
- **gettext**
- **Kbd**
- **Libibverbs**
- **Libtool**
- **Libxml2**
- **Mpcib**
- **Mpfr4**
- **Ncurses**
- **Make**
- **M4**
- **pax-utils**
- **Pciutils**
- **Perl**
- **pkg-config**
- **Popt**
- **Procs**
- **Quilt**
- **Readline**
- **sysfsutils**
- **Tar**
- **Unifdef**
- **Zlib**

**NOTE:** The maintenance effort varies considerably for different packages.
CIP Core

- CIP Core is a CIP official project
  - CIP Core aims to provide a way to create and test installable images
- Goal
  - **Input:** Debian sources/binaries and cip kernel
  - **Build mechanism:** Bitbake and/or Debian build system
  - **Output:** Minimum deployable base system image for testing
- Current status
  - Minimal rootfs can be built for the following hardware
    - Renesas RZ/G1M (iwg20m)
    - BeagleBone Black
    - Cyclone-V
    - QEMUx86

Source code: [https://gitlab.com/cip-project/cip-core](https://gitlab.com/cip-project/cip-core)
## Potential build tools for CIP Core (Comparison Elbe, Isar and Deby)

<table>
<thead>
<tr>
<th></th>
<th>Elbe</th>
<th>Isar</th>
<th>Deby</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base system</strong></td>
<td>Debian binary packages (no rebuilding)</td>
<td></td>
<td>Binary packages <strong>cross-built</strong> from Debian <strong>source</strong> packages</td>
</tr>
<tr>
<td><strong>Build system</strong></td>
<td>Custom</td>
<td></td>
<td>Bitbake</td>
</tr>
<tr>
<td><strong>Host tools</strong></td>
<td>Debian: debootstrap, qemu, elbe-pbuilder</td>
<td>Debian: multistrap, dpkg-buildpackage, qemu</td>
<td>Poky</td>
</tr>
<tr>
<td><strong>Metadata</strong></td>
<td>✓ ELBE-XML for project description</td>
<td>✓ Recipes for building product packages</td>
<td>✓ Common function to unpack Debian source packages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Recipes for image generation</td>
<td><strong>Full recipes</strong> for cross-building every Debian source package</td>
</tr>
<tr>
<td><strong>Compilation</strong></td>
<td>Native</td>
<td></td>
<td>Cross</td>
</tr>
<tr>
<td><strong>Benefits</strong></td>
<td>✓ Re-use Debian binaries and QA</td>
<td></td>
<td>✓ Affinity with Poky recipes</td>
</tr>
<tr>
<td></td>
<td>✓ Fast (re-use, parallel builds)</td>
<td></td>
<td>✓ <strong>Fully customizable</strong></td>
</tr>
<tr>
<td></td>
<td>✓ Lower development costs</td>
<td></td>
<td>✓ No need to keep binary pkgs</td>
</tr>
<tr>
<td><strong>Common features</strong></td>
<td>✓ Based on Debian packages (stability, long-term maintenance)</td>
<td>✓ Generate images by installing binary packages</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓ Manage multiple products using custom configuration</td>
<td></td>
</tr>
</tbody>
</table>


---

Open Source Summit Japan 2018 46
Contributing to Debian-cross

- RFC to Debian-cross

RFC: Patches for supporting cross-building

To: debian-cross@lists.debian.org
Subject: RFC: Patches for supporting cross-building
From: Kazuhiro Hayashi <Kazuhiro.Hayashi@toshiba.co.jp>
Date: Fri, 15 Jun 2018 23:09:58 +0900
Message-ID: <jzL1e216co7-8616-87fb-7fa6-1ac80f4aeb33@toshiba.co.jp>

Hello,

I’d like to share patches for 70 Debian packages with this WL.
https://github.com/meta-debian/debian-cross-patches
The patches are intended to make the packages cross-buildable.
According to the previous summary [1], these packages are still failed for cross-building.

Here are several questions about the patches and future contributing to Debian community:

- The target packages of above patches are selected based on the information in 2016 [2].
  Can I find any other latest cross-building results?

## Gaps and Common Goals between Debian and CIP

<table>
<thead>
<tr>
<th>Debian</th>
<th>CIP requires</th>
<th>Chance to collaborate with Debian</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Support</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Term: 3+2 years by Debian-LTS</td>
<td>- Term: 10+ years</td>
<td>- Longer term maintenance for limited number of packages (CIP joined Debian-LTS)</td>
</tr>
<tr>
<td>- Num of source pkgs: over 25000 (67776 binary pkgs)</td>
<td>- Num of pkgs: 10+ (minimum)</td>
<td>- Contributing to Debian-cross (RFC posted to Debian-cross)</td>
</tr>
<tr>
<td><strong>Build</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Should support native build</td>
<td>- Need to have both native and cross build</td>
<td>- Exchange and share the license review results</td>
</tr>
<tr>
<td>- Working on cross build packaging (Debian-cross)</td>
<td>- Binary / Source code should be managed and reproducible</td>
<td>- Contributing test cases to upstream</td>
</tr>
<tr>
<td>- Reproducible build</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OSS license compliance</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- DEP-5 adoption is ongoing</td>
<td>- Generate reports automatically</td>
<td></td>
</tr>
<tr>
<td><strong>Testing</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Packages has to be tested</td>
<td>- Easy to redistribute</td>
<td></td>
</tr>
<tr>
<td>- autopkgtest</td>
<td>- All packages should be tested in timely manner</td>
<td></td>
</tr>
</tbody>
</table>
Under discussion in CIP
Cybersecurity Standard for Industry (IEC62443)

- **ISA/IEC-62443**
  - Specifications to develop secure Industrial Automation and Control Systems
  - Certification requirements include
    - Development procedure
    - Tests
    - Certification scheme

- **How CIP is involved**
  - Developing the following items which may help to get a certification with Linux based system (also CIP based system)
    - Documents for recommended settings and configuration for open source packages
    - Test cases and tools
Q: 20 years from now?

- Y2038 is a blocking issue for 20 year commitments
- CIP members had a meeting during ELC-EU with key developers

Jan 19th, 2038

60 years product life time
Other items currently under discussion in CIP

• Functional safety
• Software updates for industrial systems

• CIP joined EdgeX Foundry as Associate member
Summary and conclusion
CIP activities by timeline

Members
CIP kernel
Testing
RT Linux
CIP Core
CIP-RT kernel
EdgeX
Debian

Topics to be discussed

Open Source Summit Japan 2018
### Relationship between CIP base layer and open source projects

<table>
<thead>
<tr>
<th>Use case</th>
</tr>
</thead>
</table>
| **Non-CIP packages**  
Linux distribution (e.g. Debian) may extend/include CIP packages. |

| CIP Core  
B@D  
CIP-RT kernel  
CIP kernel |
|-----------|
| CIP Reference  
Filesystem image with SDK  
(CIP Core packages) |

<table>
<thead>
<tr>
<th>CIP SLTS Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIP Reference Hardware</td>
</tr>
</tbody>
</table>
Summary

• CIP today focuses on
  • **Kernel maintenance:** maintaining Linux kernels for very long time including real-time support
  • **Testing:** providing a test infrastructure and evolve tests
  • **CIP Core packages:** a set of industrial-grade components that require super long-term maintenance including the required build tool chains
  • **Security:** Improving to have security features and to follow Cyber Security Standard
  • **Collaboration:** Linux stable, Debian/Debian-LTS, Real Time Linux, EdgeX Foundry

• More activities under discussion
Conclusion

• Our Civilization needs an Open Source Base Layer of industrial-grade software
  • CIP provides this, using Linux
• Sustainability is ensured by
  • The backing of big industrial and semiconductor companies
  • Close cooperation with and building with mature Open Source projects (Debian, PREEMPT_RT, kernelci, ...)
  • Providing suitable tool chains
  • Ensuring in-depth tests
• Contribution and collaboration are the key CIP activities
Thank you!
Questions?
Contact Information and Resources

To get the latest information, please contact:

- CIP Mailing list: cip-dev@lists.cip-project.org

Other resources

- CIP Web site: https://www.cip-project.org
- CIP Wiki: https://wiki.linuxfoundation.org/civilinfrastructureplatform/
- CIP source code
  - CIP GitLab: http://www.gitlab.com/cip-project
Vision: Technical topics and related projects

*Topics will be added or removed to reflect CIP technical interests*

### Application support
- App Framework
- HMI Framework
- FW update
- App deployment

### Middleware / Tools
- Coherent Security Mechanisms
- Configuration/Device management
- Multimedia
- Common issues

#### Linux Kernel
- Security
  - Anomaly detection
  - Live patching
  - LSM
  - SELinux

- Functional Safety
  - Monitoring/error detection
  - SIL2LinuxMP
  - SIL3 support

- Userland Isolation
  - LXC
  - Cgroups

- Kernel Isolation
  - SafeG
  - Jailhouse

- Monitoring / Tracing
  - Ftrace
  - ktap

- Real-time support
  - GPGPU/FPGA real-time
  - RT/non-RT communication
  - Xenomai

- Monitoring / Tracing
  - PREEMPT-RT

- Heterogeneous Computing
  - SoC FPGA

### Infrastructure and Services
- Build and production
  - Yocto Project
  - Debian build system
- Testing
  - LTP
  - kselftest
  - KernelCI
  - Fuego
- Support
  - SLTS
  - Backwards compatibility
- Development process
  - SIL2 support
  - SIL3 support
- Legal topics
  - SPDX
  - FOSSology
- License Clearing
  - Export Control

### Hardware / SoC (multiarchitecture)

**Legend**
- To be specified / implemented by CIP
- Integration / cooperation