About us

• Kazuhiro Hayashi <kazuhiro3.hayashi@toshiba.co.jp>
  • Working for Toshiba Corporation
  • (In-house) Embedded Linux development for products, consultant
  • Member of CIP, developing tools of CIP Core
  • Contribute to Linux and other OSS projects

• Jan Kiszka <jan.kiszka@siemens.com>
  • Working for Siemens Corporate Technology
  • (In-house) Embedded Linux consultant & developer
  • Member of CIP, isar-cip-core development, some CIP kernel backports
  • Maintainer of and contributor to various OSS projects
Agenda

- Requirements in Industrial Product Development
- CIP as the solution
- What is “CIP Core”?
- CIP Core implementation
  - Isar
  - Deby
- Building products on top of CIP Core
- Future plans
Requirements in Industrial Product Development

Start

- Porting
- OS/middleware development
- Application development
- Installation
- Testing
- Development release

Development

- Product Certification
- Publication

Release

- Bug fix, security fix
- System update

Maintenance

- Architecture & board support
- Customizability, Scalability
- Standalone SDK
- Ready-to-use image generation
- Stability

- License clearing

- Long-term maintenance
- Verified & automated update

- Quick release
- Modularity for multiple projects
What to Do First…

- Select appropriate base system
  - Linux distribution
- Provide tools for integrating the base system into products
  - Build system, etc.

- Architecture & board support
- Customizability, Scalability
- Standalone SDK
- Ready-to-use image generation
- Stability

- License clearing
- Long-term support
  - Verified & automated update
- Quick release
- Modularity for multiple projects
CIP is the Solution

Industrial grade
- Reliability
- Functional Safety
- Real-time capabilities

Sustainability
- Product life-cycles of decades
- Backwards compatibility
- Standards

Security
- Security & vulnerability management
- Firmware updates
- Minimize risk of regressions
CIP is the Solution

Establishes an “Open Source Base Layer (OSBL)”

- CIP Core packages (tens)
- CIP kernel (10+ years maintenance, based on LTS kernels)
- additional packages (hundreds)
- company-specific middleware and applications
CIP Core

- One of the CIP projects focusing on user land software and tools
- Goals
  - Define a list of “CIP Core packages” maintained for long-term
  - Provide a reference implementation including “CIP Core packages”
  - Test the implementation on the “CIP reference hardware”

<table>
<thead>
<tr>
<th>Project</th>
<th>SLTS Kernel</th>
<th>Real-time</th>
<th>Testing</th>
<th>CIP Core</th>
<th>Security WG(*)</th>
<th>Software update WG</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

CIP Projects and its scopes

- Industrial grade
- Sustainability
- Security

(*): Workgroup
CIP Core: Position in CIP Projects

- **CIP Core Reference Implementation**
  - Build Tool
  - SLTS kernel
  - Real-Time
  - CIP Reference Hardware

- **Testing (CI)**
  - Deploy
  -Discussion & Decide
  - Build & Integrate

- **CIP Core Packages**
  - Request Packages & Configurations
  - Software update WG
  - Security WG

- **CIP members**
- **CIP Reference Hardware**
- **Tested on targets**
CIP Core: Implementation

• Debian-based implementation
  • Mature, high-quality, mainstream distro.
  • Many new & old architecture supports
  • Suitable for small and big installations
  • Security updates

• Profiles

<table>
<thead>
<tr>
<th>Approach</th>
<th>Generic profile</th>
<th>Tiny profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>Binary packages</td>
<td>Source packages</td>
</tr>
<tr>
<td></td>
<td>Isar</td>
<td>Deby</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Build tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isar</td>
</tr>
<tr>
<td>meta-debian</td>
</tr>
</tbody>
</table>

- Add packages, send patches, etc.
- Funding to Debian LTS

Binary packages
Source packages

Use
Use
Contribution

CIP Core
CIP Core Project
Implement

Generic profile
Tiny profile

Embedded Linux Conference Europe 2019
Deby (meta-debian)

• Yocto Project extension for using Debian source packages
  • Source code: Debian
  • Build system: Yocto Project

• Goals
  • Achieve stability and long-term support with the Yocto advantages

• Features: Yocto based flexibility & extensibility
  • High customizability by own recipes
  • Small footprint (Around 2MB)
  • Various target CPUs and tunings
  • Adaptation to BSP layers provided by board vendors

• Repositories
  • Upstream: https://github.com/meta-debian/meta-debian
  • CIP Core: https://gitlab.com/cip-project/cip-core/deby
Deby: How it works

Debian sources

meta-debian (Deby)

pkgB.bb  pkgC.bb  pkgD.bb

debian-package.bbclass

meta-bspX

pkgB.bbappend

meta-bspY

app.bb

Extra sources

Upstream sources

poky

pkgA.bb

.bitbake

kernel  rootfs

SDK

QEMU

kernel  rootfs

U-Boot  SDK

Board X

kernel  rootfs

U-Boot  SDK

Board Y

Debian sources

Extra sources

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend

Debian sources

pkgB.bbappend
Isar

• Integration System for Automated Root filesystem generation
  https://github.com/ilbers/isar

• Goals
  • Build systems in a Debian way
  • Developer-centric workflow: One-command building
  • Make customizations easy and repeatable
  • Efficient building

• The best of both worlds
  • Debian: Tested binary packages, tools, security updates
  • OpenEmbedded / Yocto: bitbake, recipes, layers

• Reuse Yocto knowledge of developers
Image Generation Sequence of Isar

1. debootstrap Debian for target, also for host if cross-building
2. Create buildchroots (target and host)
3. Build custom Debian packages
   • pre-debianized packages
   • ad-hoc debianized packages (customizations, u-boot, kernel, ...)
4. Assemble rootfs
   • debootstrap output
   • external packages
   • self-built packages
5. Run images (typically wic)
   • Filesystem image generation
   • Partitioning
   • Bootloader installation and configuration
Example: Building Images for BeagleBone Black

• Generic profile (Isar)

```bash
$ git clone https://gitlab.com/cip-project/cip-core/isar-cip-core && cd isar-cip-core
$ wget https://raw.githubusercontent.com/siemens/kas/master/kas-docker
$ chmod a+x kas-docker
$ ./kas-docker --isar build kas.yml:board-bbb.yml
$ dd if=/path/to/cip-core-image-cip-core-buster-bbb.wic.img of=/dev/mmcblk0 ...
```

• Tiny profile (Deby)

```bash
$ git clone https://gitlab.com/cip-project/cip-core/deby && cd deby
$ ./scripts/setup-kas-docker.sh
$ ./kas-docker build kas-bbb.yml
```
## Preferred Use Cases

<table>
<thead>
<tr>
<th></th>
<th>Isar (Generic Profile)</th>
<th>Deby (Tiny Profile)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Available Packages</td>
<td>= Debian</td>
<td>App. 50 (+ Yocto Extension)</td>
</tr>
<tr>
<td>Footprint</td>
<td>&gt; 100MB</td>
<td>2MB - 100MB</td>
</tr>
<tr>
<td>Compatibility</td>
<td>Debian (Binary packages)</td>
<td>Yocto Project (Recipes)</td>
</tr>
<tr>
<td>Required skill set</td>
<td>Debian (Packaging) bitbake</td>
<td>Yocto Project</td>
</tr>
<tr>
<td>Build time (minimal image)</td>
<td>Around 10min</td>
<td>Around 1h</td>
</tr>
<tr>
<td>Customization needs</td>
<td>Selected packages</td>
<td>Up to toolchain settings</td>
</tr>
<tr>
<td>Fitted systems (Examples)</td>
<td>IoT gateways, edge devices, industrial controllers ...</td>
<td>Small IoT devices ...</td>
</tr>
</tbody>
</table>
CIP Core: Testing Architecture

- **Pull updates**
  - **isar**
  - **meta-debian**

- **Build Request**
  - **isar-cip-core**
  - **GitLab (cip-project/cip-core)**
  - **deby**

- **Deploy**
  - **Build Server (GitLab Runners on AWS)**
  - **Artifact Storage (AWS S3)**
  - **Built Artifacts** (Kernel, Boot loader, rootfs)

- **Test Request**
  - **LAVA Master**
  - **Job scheduling & Summarizing results**

- **Control**
  - **LAVA Worker**

- **CIP Reference Hardware**
Building Products on Top of CIP Core

• Requirements
  • Customize base layer
  • Add product applications

• BitBake layering
  • Append changes to lower layers easily
  • Reuse company / product common elements
  • Available in both CIP Core profiles

• Current approaches in CIP members
  • Building products using this pattern on upstream projects (Isar, Deby)
  • Quicker integration of CIP results desired

(Example of using “Isar”)
Future Plans for CIP Core Implementation

• Enable direct use in product development
  • Regular releases of tested layer with dependencies
  • Mirroring of source & binary dependencies

• Provide image corresponding to CIP package list

• Integrate and test results of other CIP workgroups
  • Robust system update (Software update WG)
  • Functions to meet cybersecurity standard requirements (Security WG)
Summary

• CIP provides long-term maintained Open Source base layer, consisting of kernel and essential packages
• CIP Core defines package set and ensures integration
• Two implementation flavors available
  • Deby for smaller, Yocto/OE-compatible projects
  • isar-cip-core for medium to larger, Debian-compatible projects
• More product-ready features to come, from software update to security hardening
Questions