Jonni Bidwell always suspected Linux would save the world. Industry experts Yoshitake Kobayashi and Urs Gleim all but confirmed his hunch...
The Civil Infrastructure Platform (CIP) is a Linux Foundation initiative. It aims to establish a base layer of industrial-grade software to power critical services such as energy, water, transportation and communications — the lifeblood of today’s civilisation.

Many of these projects run on open source software, and many more will do so in the future. Yet it’s completely unfeasible to update the software running these things every five years (the current life-span of LTS distros), and many of these systems are looking at life-spans beyond 50 years. So the CIP introduces the idea of a super long-term support (SLTS) kernel.

CIP started in April 2016 and the initial supporters were Hitachi, Siemens and Toshiba. Since then, a number of other companies have joined the CIP. It’s an edge-to-cloud connectivity provider that provides solutions for factory automation, smart cities and monitoring. Also in January you’ll probably remember the Spectre and Meltdown attacks — which affect embedded ARM and Intel hardware as well as desktop CPUs — were disclosed. So the relevant patches are being backported to the CIP SLTS kernel. Some hardening features from the Kernel Self-Protection Project have been added, too.

We’ve also had a few CIP kernel releases. In April Ben Hutchings released the latest one: 4.4.126-cip22. Kernel 4.4 was released in January 2016, and was initially earmarked for long-term support until February 2018. That was extended in September 2017 to February 2022. It’s possible that someone else will maintain it after this period (as happened with Ben Hutchings’ tour de force support of the 3.2 kernel, which after six years came to an end in May). This would in some sense make the CIP team’s lives easier, but is far from a free ride.

Keep up with developments at the Linux Civil Infrastructure Platform by visiting www.cip-project.org.

Linux, and it’s not an exaggeration. Things like railway infrastructure, healthcare and industrial automation, these all have long-running systems. We’re talking between 10 and 40 years, maybe even longer.

So we can’t afford to change the software, say, every two years (as a cautious desktop user might). This is especially the case where safety certifications are involved — transport networks and power generation for example. Here, it can take close to two decades just to put a new system into service. So a better strategy is to apply security patches and small updates. For CIP the idea is to stick with one version of the Linux kernel and maintain it for as long as we can. The CIP kernel is much more focused on embedded devices than other long-term initiatives. So we have support for all the embedded board ports, but in most cases don’t support desktop PCs or servers. The systems we’re dealing with all run on dedicated hardware and embedded chipsets, so that’s what we support.

Many of these kinds of systems have already been using Linux for some time, and individual companies have already been working on their own super long-term maintenance. What we want is everyone working on the same platform, which will avoid some duplication of effort. But most importantly, we want this work to be done collaboratively with the upstream communities, not locally.

Linux Format: Linux is running in all kinds of places, and lives depend on some of those applications. What is the Civil Infrastructure Platform and how is it going to help civilisation going forward?

Yoshi Kobayashi (YK) and Urs Gleim: (UG): Yes, early on in our presentation we have a slide entitled Our Civilization is Run by Linux, and it’s not an exaggeration. Things like railway infrastructure, healthcare and industrial automation, these all have long-running systems. We’re talking between 10 and 40 years, maybe even longer.

So we can’t afford to change the software, say, every two years (as a cautious desktop user might). This is especially the case where safety certifications are involved — transport networks and power generation for example. Here, it can take close to two decades just to put a new system into service. So a better strategy is to apply security patches and small updates.

For CIP the idea is to stick with one version of the Linux kernel and maintain it for as long as we can. The CIP kernel is much more focused on embedded devices than other long-term initiatives. So we have support for all the embedded board ports, but in most cases don’t support desktop PCs or servers. The systems we’re dealing with all run on dedicated hardware and embedded chipsets, so that’s what we support.

Many of these kinds of systems have already been using Linux for some time, and individual companies have already been working on their own super long-term maintenance. What we want is everyone working on the same platform, which will avoid some duplication of effort. But most importantly, we want this work to be done collaboratively with the upstream communities, not locally.

Linux Format: When did the project start and who were the initial industrial backers?

YK: CIP started in April 2016 and the initial supporters were Hitachi, Siemens and Toshiba. Since then, a number of other
partners have come on board: Renesas Electronics, Codethink and Plat’Home. They support CIP by contributing directly to upstream projects and funding work related to the CIP’s goals.

LXF: How do you go about testing and certifying different boards? This sounds like an awful lot of work...

YK: These systems run on lots of different hardware platforms, so each one needs to be individually tested and certified. But there’s a lot of similarity across the tests that each board needs to run. So for example, the kernel has to be tested on every board, and there’s a common software stack that runs these tests. So yes, we have to test all the different hardware, but by sharing these common parts, the workload is reduced.

UG: Part of the CIP’s role is to produce this testing infrastructure too, so everyone is using the same test tools, the same test automation. It’s important to harmonise all these testing infrastructures, because once that’s in place it’s not that much work to support additional dedicated boards. There are a few talks on this, one of the projects is called Board at Desk (B at D—see box, below), which enables you to easily set up a development environment on your local workstation, which is connected via serial link to a board. B at D can currently be tested on our reference board: the Beaglebone Black and the Renesas RZ/G1M.

LXF: Security is obviously a big issue for these kind of systems. How are you going to tackle this?

YK: We’ll port all the security patches from the Linux kernel community to our CIP kernel and test them on the relevant hardware – hopefully that’s not too onerous.

UG: The other concern is userland, above the kernel layer. We have what we call the CIP base layer, which is our common software stack. We want as much of this stack to be shared across the hardware. This would be too much work for just one team, so collaboration is important here: collaboration with both Linux distributions and other projects.

UG: The most work in terms of security is done in the server area. Google and other big companies are the major contributors here. There are a few relevant projects, the Core Infrastructure Initiative (CII, see www.coreinfrastructure.org) and the Kernel Self Protection Project (see https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project) for example. We take things from those and backport them to the CIP kernel. The CIP kernel is based on Kernel 4.4, and we backport security patches and features from newer kernels to it.

LXF: Reproducible builds, particularly in Debian, have been getting a lot of attention. Might these be used in the CIP? What extra tooling and testing would be required?

YK: Reproducible builds is a future topic for the CIP Project. We haven’t started work on this yet but plan to soon. Currently, we’re focusing on Debian activities and long-term support. In the next year, you’ll see more collaboration and strategy with Debian and plans for both groups: work done in CIP context and/or supporting the Debian efforts on this.

LXF: The release of CIP Core was just announced at this conference [in October 2017]. Is this like a distribution for your kernel efforts?
We’re focused on the kernel first. The CIP Core sits on top of that and provides a minimal reference filesystem. It’s not a proper distribution by any stretch of the imagination, mind. We have only the basic packages, a shell, some protocols (provided by OpenSSL), a bootloader, glibc, busybox, binutils and compilers. That’s really it for the moment, so you can see it’s not really comparable to Ubuntu or something.

The CIP core is the lowest common denominator; individual systems will add whatever they need. The release is like a first milestone to create our software stack. This gives us our kernel and candidates for userland packages. It’s based on Debian source code and binaries. We decided to collaborate with the Debian community because they already provide long-term support.

Note that the community provides support for five years (through the Debian LTS effort), whereas CIP needs to be supported for at least ten years. We need to work with the community to bridge that support gap. Once we gain experience with this kind of support longevity, future kernels will have even longer support periods. CIP is a platinum-level supporter of Debian LTS.

Other Internet of Things projects have been talking about doing updates atomically, rather than per-package. Can you say if this approach be adopted by CIP Core?

At the moment that sort of thing would have to be implemented on top of the CIP base layer, so we’re not strongly focused on that at the moment. But IoT use cases, and package/firmware updates in general, are important, so we may introduce something like this in future.

Yes, this will be an option for projects later on. Internally, in our company for example, we have different solutions depending on the projects: some based on packages, some based on binary updates with bindiff, and so on.