HOW ENGINEERING LEADERS CAN USE THE YOCTO PROJECT TO SOLVE COMMON LINUX EMBEDDED CHALLENGES

THE LINUX FOUNDATION TRAINING PUBLICATION

RUDOLF STREIF
DIRECTOR OF EMBEDDED SOLUTIONS
THE LINUX FOUNDATION

THE LINUX FOUNDATION

The Yocto Project can help you and your engineering team get the next product development project off the ground faster and be in better position to finish it successfully, on time and within budget.

If you are leading an engineering organization, chances are that you either familiar with or will experience one of these common scenarios:

Scenario 1: You and your team are gathered in a meeting room discussing the base architecture for your next product development project. Your staff engineers are proposing a Linux-based system, have already prepared a case study and are now presenting their evaluation matrix comparing the Linux solution with other potential approaches including your current solution. The presented data looks convincing, your team is excited and ready to move forward with Linux and open source. However, you are concerned about how to make the transition, educate the entire team, and scale the case study for production.

Scenario 2: Your company has already been shipping a Linux-based product for a while. Your team has chosen one of the standard desktop or server Linux distributions to get started quickly and meet the project deadlines. It made a lot of sense at the time, although you and your engineers were aware that the footprint and functionality of a full-fledged Linux distribution will potentially be too large for next product generations. Now the time has arrived and your customers and sales team are asking for more nimble solutions at a lower price point.

Scenario 3: Your company has been shipping Linux-based products for a while using a home-grown operating system stack created by a combination of open source tools and a build system that your team has developed around them. While this solution worked well in the beginning, your products are now outgrowing the system. They are requiring more functionality which in-turn is creating an increasingly complex matrix of dependencies. The time your team spends on maintaining the solution is almost exponentially taking a larger and larger percentage of the total development time. You are also looking for a more formalized build environment since your current system seems to be largely dependent on the skills of whoever is using it, causing various integration and regression issues.

No matter what your actual situation is, the core set of challenges which keep you awake at night, and for which you as the engineering leader will have to respond to adequately, remains pretty much the same. Whether you are considering Linux-based system development for the first time, or are looking for a better approach after having learned your initial lessons, chances are that you are facing a multitude of common challenges:

- **Controlling your Linux operating system stack**
- **Maintenance**
- **Build system and tooling**
- **Open source licensing requirements**
- **Support**
- **Ramping up and scaling your organization**

In the following sections, we will explore how the Yocto Project can help you and your engineering team get the next product development project off the ground faster and be in better position to finish it successfully, on time and within budget.

CONTROLLING YOUR LINUX OPERATING SYSTEM STACK

The Yocto project creates a custom Linux distribution for you and provides you with a set of common configurations to choose from.

The two principal approaches for building a Linux operating system stack for your product are the following:

- TOP-DOWN: Leveraging an existing Linux distribution and scaling it according to product requirements;
- BOTTOM-UP: Building a custom Linux distribution for your product starting with the kernel and adding packages as needed;

Both of these options have their advantages and their challenges. Let's explore.

Top Down

Leveraging an existing Linux distribution that you can download and install on your target hardware jump starts your development. However, what do you do if the CPU architecture of your hardware is not supported, peripheral devices have no drivers, and other problems typically found with embedded systems? Furthermore, how do you scale the distribution to your needs? All of those distributions come with a package management system that lets you install and uninstall components. While they are handling the dependencies, it remains a cumbersome process at the end of which you will have to create a file system image to install on your target hardware when going into production.

Bottom-Up

Building a custom Linux distribution from scratch gives you the most control over your operating system stack including customizing and optimizing the Linux kernel potentially for multiple architectures, adding device drivers, and more. However, it is not a trivial task and the tools traditionally available have been limited.

Enter: The Yocto Project

The Yocto Project combines the best of both worlds. While the Yocto Project is not an (embedded) Linux distribution but creates a custom one for you, what it does provide is a set of common configurations to choose from. This includes a minimal system with console login, a system with a basic graphical user interface for mobile devices and even a system that is compliant with the Linux Standard Base (LSB), to get your team started quickly. After selecting your initial configuration and your target system, which can be an emulated target or actual hardware, the Yocto Project fetches all the necessary source code for the components that comprise the system, builds its own toolchain and then uses that toolchain to build all the other software components. Within a couple of hours or less, depending on your build system, the Yocto Project creates boot loader, kernel and root file system images according to your configuration that you can either launch in an emulator or transfer to actual hardware.

After the initial build, components included in the system can easily be added and removed by modifying the build recipes, either by editing them directly or using a graphical user interface. Recipes are organized in layers that provide separation of your own recipes from Yocto Project core recipes and board support packages (BSP). Yocto BSPs follow certain conventions which makes them interchangeable. Simple configuration using a single setting allows building the same operating system stack for different target hardware.

MAINTENANCE

The Yocto Project pays particular attention to the Linux kernel and applies patches and security updates to it's repositories. This means that with every build, you will receive a fully patched and up-to-date kernel.

Linux distribution maintainers spend considerable time and effort looking for patches and new releases of components included in their distributions. If your team has built a custom Linux distribution for your products using your own build environment, you know that with the number of components included in your distribution, this task becomes exponentially more difficult and time consuming. Not only do you have to evaluate each patch and new revision of a component, but you will also have to take packages into consideration that are dependent on that component, as well as packages it is dependent on.

The Yocto Project facilitates maintenance for your team in two ways:

1. Recipes to build components are permanently updated by the Yocto Project team for newer versions as well as any relevant patches. Additionally, recipes for prior versions remain part of the build system and are updated with patches as necessary.
2. The Yocto Project maintains its own source repository mirrors from which component sources are retrieved.

The former allows your team to either use the latest version of a component or stick to a previous version according to your product development cycle requirements. The latter ensures that even if the upstream project of a component makes changes to their repositories and eventually chooses to discontinue prior versions, your team will still be able to retrieve the component source of prior versions.

Furthermore, you can maintain your own source repositories from which your Yocto Project build environments can draw, which makes your build solution entirely self-sufficient.

The Yocto Project team pays particular attention to the Linux kernel by maintaining chosen kernel versions from www.kernel.org inside the Yocto Project repositories and applying patches and security updates to them. When building the Linux kernel, the Yocto Project draws from its kernel repositories which means that with every build, you will receive a fully patched and up-to-date kernel. That even includes patches to the Linux Foundation's Long Term Stable Initiative (LTSI) kernel versions that have not yet been accepted by the mainline kernel. One less thing for your team to worry about.

BUILD SYSTEM AND TOOLING

The Yocto Project builds cross and target toolchains for you as part of it's regular build process.

Embedded system software development requires a different set of tools than native software development, where the build system and target system are mostly identical. That commonly starts with a cross toolchain and goes all the way to remote debugging and profiling on target devices. Furthermore, integration with software configuration management (SCM) systems and quality assurance (QA) needs to be considered.

Many engineering teams end up spending considerable time building their own tools and/or integrating tools that they have received from hardware and software vendors into their workflow. The Yocto Project builds cross and target toolchains as part of its regular build process. If desired, it also builds a remote debugger and performance profiling tools for the target and automatically includes them with the file system image. Such tools include: LatencyTOP, PowerTOP, Oprofile, Perf, SystemTap, and Lttng-ust.

The Yocto Project also integrates with virtually any source code management (SCM) system, including but not limited to: GIT, Subversion, CVS, Perforce, Bazar, etc. So, no matter what SCM your team is using, the Yocto Project will be able to integrate with it.

OPEN SOURCE LICENSING REQUIREMENTS

The licensing information the Yocto Project automatically creates when building your custom Linux operating system stack relieves your team from the arduous work of collecting and assembling it manually.

A common misconception is that open source licensing is a book sealed with seven seals. Fortunately, the reality of open source licensing is much more accessible. If you are shipping a product that includes components licensed through one or more open source licenses:

- Provide a manifest of the software packages included with your product together with the licenses they are using;
- Provide the actual license text.

The Yocto Project facilitates open source license management in multiple ways:

- Every Yocto Project recipe must provide information about the license used by the component the recipe is building;
- Every Yocto Project recipe must provide an MD5 checksum calculated over the actual license text;
- Using license information and checksum, the Yocto Project verifies the correctness of the license, monitors changes to the license text, and creates a license manifest for every software package included in an image;
- Future versions of the Yocto Project will also support the Software Package Data Exchange (SPDX) specification, a standard format for exchanging component information, associated licenses and copyrights for a software package;
- The Yocto Project does not impose any licensing requirements by itself on the output it creates.

The licensing information the Yocto Project automatically creates when building your custom Linux operating system stack relieves your team from the arduous work of collecting and assembling it manually.

SUPPORT

An entire ecosystem of organizations that are using, depending on, supporting, developing, and providing resources for the Yocto Project has evolved.

The prototypical support mechanisms for open source projects are mailing lists and wikis. And of course, the Yocto Project provides those as well as a fairly detailed set of documentation you can find at yoctoproject.org/documentation. However, where do you turn if your team has a very specific problem? The stereotypical response on the mailing list is "Just read the source code, it's all there" but that is far from being satisfactory.

An entire ecosystem of organizations that are using, depending on, supporting, developing, and providing resources for the Yocto Project has evolved (yoctoproject.org/ecosystem). These include semiconductor companies developing BSPs for their hardware, toolchain companies offering integrated development solutions, consultancies providing engineering services, and much more. These organizations are available and ready to support you and your team.

Ramping Up and Scaling Your Organization

Your team can only adopt new methods and tools effectively and efficiently when combined with the proper education and training. This is where the Linux Foundation's course offerings provide your team with the necessary edge to ramp up and scale more quickly. After attending our course Building Embedded Linux with the Yocto Project (LF405), your team members will be equipped with a solid understanding of embedded system development with the Yocto Project, including the Poky reference build system and BitBake, the use of emulators, building the boot loader, kernel and file system images for multiple architectures and the creation of board support packages.

Other Linux Foundation developer training classes include: Embedded Linux Development (LF411), Developing Linux Device Drivers (LF331), and Linux Kernel Internals and Debugging (LF320).

For a complete list visit http://training.linuxfoundation.org/linux-courses.

For your convenience, you and your team have several training options to choose from:

- CORPORATE ON-SITE LINUX TRAINING: Targeted training, customized to your teams needs, at your site, a training facility of your choice, or, for certain classes, in an online classroom.
- OPEN-ENROLLMENT LINUX TRAINING: Courses held at training facilities throughout the U.S. and around the world, and, for certain classes, online.
- TRAINING AT LINUX FOUNDATION EVENTS: Condensed versions of our on-site and open-enrollment training courses offered in conjunction with our events such as the Embedded Linux Conference, LinuxCon, Collaboration Summit and the Enterprise End User Summit.

To learn more about training from the Linux experts for you and your team visit http://training.linuxfoundation.org.

ABOUT THE AUTHOR

Rudolf Streif manages The Linux Foundation's initiatives for embedded solutions working with the community to provide environments and platforms for embedded Linux systems. Mr. Streif has an extensive background in embedded software development and bringing products based on Linux to market.

WHY TRAIN WITH THE LINUX FOUNDATION

We needed someone who could fully engage with Ph.D.-level developers. We had no doubt that we'd found the right instructors.

Dana Krokosky, *Compunetix*

The willingness of the Linux Foundation to customize the course to our needs was the biggest determining factor for choosing them.

Matthew Cheng, *Broadcom*

The Linux Foundation offers several embedded Linux training courses:

Embedded Linux Development (LF411)
Get advanced Linux training on the key steps to developing an embedded Linux product. Gain real world experience through extensive hands-on practice with target devices. Learn More:

Building Embedded Linux With The Yocto Project (LF405)
Gain a solid understanding of embedded development using the Yocto Project, including the Poky build process and Bitbake, the use of emulators, building images for multiple architectures and the creation of board support packages (BSP).

Introduction to Embedded Android Development (LF308)
This class will teach you how the Android build system works and how to add a completely new device definition, how to customise the components that go into the build, how to obtain and build a Linux kernel with Android additions and how to load it onto the new target board and configure the boot process.

Distribution-Flexible

The Linux Foundation's courses are built to be distribution-flexible, allowing companies or students to easily use any of the big three distribution families: Debian, Fedora or OpenSUSE. If your company runs one of these Linux distributions and needs an instructor who can speak deeply on it, we have a Linux expert who knows your distribution well and is comfortable using it as the basis for any corporate Linux training. For our open enrollment students who take our online training or classroom training, our goal is to help them, first and foremost, to become Linux professionals, rather than focusing on how to use one particular set of tools.

Technically-Advanced

The Linux Foundation's training program has a clear advantage. As the company that employs Linux founder Linus Torvalds, we are fortunate in our ability to leverage close relationships with many of the top members of the Linux community, including Linux kernel maintainers. This led to the most comprehensive Linux training on the market, delivered through rigorous five-day courses taught by Linux experts who bring their real world experiences to every class.

For more information about our Linux training, please visit **training.linuxfoundation.org** and contact us today.