Refresher to Embedded Linux & Intro to the Yocto Project

Robert Berger - Reliable Embedded Systems e.U.
Consulting Training Engineering
https://www.ReliableEmbeddedSystems.com
robert.berger@ReliableEmbeddedSystems.com

Course Objectives

After a quick “Refresher to Embedded Linux (2 days)” to provide an understanding of the essentials to utilize the “Yocto Project (3 days)”. After the “Refresher” we’ll see how BSP/framework maintainers would use the “Yocto Project”. We’ll also see how developers can use it, who might not even want/need to know they are using it.

Description

This five day training class uses hands-on exercises combined with instruction to illustrate the concepts of "Embedded Linux" and the "Yocto Project". It is designed to bring you quickly up to speed. It answers frequently asked questions like:

/> What is Linux?
/> Where to get u-boot/the kernel from? How to configure/build/install it?
/> Why to use "upstream"?
/> How does interprocess communication work and what to use/avoid?
/> Is it really necessary to use another version of the toolchain/libraries/packages for each and every Linux project and an top of that to follow a different work-flow?
/> Can you ensure that the development environment is identical for all developers/suppliers and that you can still produce identical builds like today in 10+ years from now?
/> Can the YP help you with Open Source license audits or do you prefer a copyright troll [1] instead?
/> … and much more

Hands-on sessions are performed on the host, in a docker container and on some target hardware (e.g. i.mx6 [2]). After the training you will be able to download a docker container based on Ubuntu with all dependencies pre-installed plus the examples in order to work with the course material in your own labs. Please note that the first two days “Refresher to Embedded Linux” assume that you already have (Embedded) Linux experience to follow the “Yocto training”. For the first two days, we briefly look at how Embedded Linux works and configure / build, among other things, the Linux kernel. In case you search for an Embedded Linux beginner course we’ll recommend “Embedded Linux - From System Architecture to Real-Time (5 days)” [3] or “Introduction to Embedded Linux in Theory and Practice - a Crash Course (3 days)” [4].
Prerequisites

/> Basic familiarity with using a Linux system (e.g. Ubuntu) as an end user in user space
/> Basic familiarity with a command line shell
/> Basic knowledge of user/kernel space programming with Linux
/> Intermediate C programming knowledge
/> It helps to be familiar with “Embedded Linux - From System Architecture to Real-Time (5 days)” [3] or “Introduction to Embedded Linux in Theory and Practice - a Crash Course (3 days)” [4] but even if you are not we’ll try to get you up to speed in the first two days.
/> It might be helpful if you attended “Embedded Linux Kernel Internals and Device Drivers (5 days)” [5], but that’s not really a prerequisite. It’s sufficient to know how to build the Linux kernel, kernel drivers in/out of tree and the fdt from the kernel side of things to follow this training and we’ll have a look at this as well during the first two days.

Who should attend?

You think about using Linux or already use Linux for your projects and have probably heard about the Yocto Project, but did not dare to have a closer look into it, or had difficulties using it. Besides you want a refresher on Linux interprocess communication and real-time. You don’t know whether and how your daily workflow can be accommodated in the YP and generally find the YP rather complicated. Why do we need all this since up to know everything was (supposedly) much easier? After the training you should be able to decide whether you need the YP or not. The workshop is aimed at software-, development-, system engineers, testers, administrators, engineers and other parties interested in the YP, with a basic knowledge of Embedded Linux.
### Delivery options

All training material is in English, but the delivery of it can be in English or in German, as you wish, worldwide. During the training we’ll provide:

<table>
<thead>
<tr>
<th></th>
<th>on-site</th>
<th>remote</th>
<th>on-site/remote combination</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>public</strong> 1 subscriber from a single company</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>private</strong> from 3 subscribers in single booking</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>e.g. from single company</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GTR(^1) from 3 subscribers instructor driven</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>workbook</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>in English/pdf printed on request</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>host machine</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>laptop</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>target board</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>remote access to host machine and board</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(web) screen-sharing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(web) audio/video</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(web) whiteboard</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(web) chat</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(web) Q&amp;A</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>(phone) audio-bridge</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

For possible training delivery options check here [6] and/or ask here [7]. The workbooks typically stay with the students after the training. For on-site training we normally place two trainees on one workspace so they can help each other.

---

\(^1\)guaranteed to run
\(^2\)discounts apply when you book 3 seats and more
\(^3\)questions and answers
\(^4\)add/remove chapters from my existing trainings
\(^5\)while training is in progress
Course Outline

Day 1

Introduction

/> Introduction | History

Eval Board

/> How does Linux boot on a PC and on the Eval Board? | Boot Sequence | SD card partitions | See it booting

Stuff needed

/> git | u-boot

/> scripting | fancy stuff | checkout | config | build | install

/> kernel

/> checkout | config - kconfig | build - kbuild | kernel modules | install

/> Flat device tree (fdt)

/> investigate | build | install

Kernel Modules

/> init/exit | Licensing | tainted module/kernel | EXPORT_SYMBOL | out of tree .ko makefile

/> module-init-tools

/> put module in kernel tree

/> parameter passing

Character Driver

/> Device Files | Device Types | Device Nodes major/minor | Kernel Driver Interface

/> Device Driver: Intro | Registration | Initialization | Open/Release

/> Miscellaneous Character Drivers

User Space Debugging

/> Simple Tools

/> ls | ltrace | strace | procfs | top | netstat | syslog | ...

/> Advanced Tools

/> What's a debugger?

/> gdb: target gdb | gdb remote debugging

Kernel Debugging (optional)

/> Debugging Intro | kgdb/kdb + agent-proxy | JTAG
Day 2

Processes and Inter Process Communication

/> Linux architecture | operating system | scheduler - priorities - scheduling classes | process - task - thread | errno | fork | process termination | process states | zombies | watch processes | simple IPC

/> shell redirection | shelling out | tmpfiles

/> advanced IPC

/> pipes | signals | interrupted system calls | message queues | semaphores | mutex | shared memory | sockets | select | poll

/> other IPC methods | IPC techniques to avoid

Real-Time

/> prerequisites

/> interrupts | re-entrant code

/> Real-Time Intro

/> What is Real-Time? | What is hard Real-Time?

/> Real-Time Linux

/> Hypervisor | Dual kernel | Fully Preemptive Kernel

/> Degrees of Real-Time behavior

/> explicit/implicit preemption points | real-time preemption patch | fully preemptive kernel | hard real-time extensions

/> Dual Kernel: Adeos/Xenomai

/> Adeos patch | Xenomai | patch/config/build kernel | run it on board

/> Real-Time Myths

Day 3

Yocto Introduction

/> What is Yocto? | What is the YP? (features/challenges) | Some tools under the YP umbrella (Poky | BitBake | OE-Core | Metadata) | Why use the YP?

How to become part of our Yocto Project community (optional)

/> First steps | How to get in touch | Participate | Contribute | Social Media | Events

Development Environment

/> What is needed for a YP build? | Layer versions/dependencies | host dependencies | config files | host/kernel dependencies
The YP Workflow

/> Intro

/> Workflow | OE architecture

/> Configuration (User | Metadata(Recipes) | Machine(BSP) | Distribution Policy)

/> Features | Recipe Versioning | Layers

/> Sources

/> Source fetching | Patching | Configure/Compile/Staging | SSTATE | Pseudo | recipetool | Examples of Recipes | PACKAGECONFIG | Packages and their contents | Output analysis/Packaging | Image Generation | SDK Generation | Tasks

/> Customizing Images (Intro | local.conf | IMAGE_FEATURES | custom image recipes (.bb files) | custom packagegroups)

Day 4

BitBake

/> History

/> Syntax (Variable Expansion | Variable Assignment | Pre-/Append | Removal | Variable Flags | Conditional Syntax - OVERRIDES)

/> BitBake Debugging (debug level | find recipes/images/packagegroups | BitBake environment/tasks/logging | force build/specific task | cleansstate | invalidate stamp | devshell | dependencies | packages | kill all BitBake instances | BitBake graphical wrapper)

/> Cleaning (gain disk space | rebuild)

Layers

/> Intro | bitbake-layers tool | dynamic layers

BSP

/> Intro | System Development Workflow | BSP Developer's Guide (bsp-tool - ported to recent Poky versions) | BSP creation | non-mainline kernel patches

Kernel

/> Intro | System Development Workflow | Kernel Development Manual (defconfig | defconfig + configuration fragment | in tree kmod | out of tree kmod | fdt classic | fdt with devicetree.bbclass | ...)

Day 5

Software Development Kit

/> Intro | Cross-Development Toolchain | Sysroot | BBCLASSEXTEND | Multilib | The QEMU Emulator | SDK: Eclipse Yocto Plug-in (deprecated) | User Space Tools | Installing SDKs & toolchains

/> Cross-Toolchains/SDKs
Building a Cross-Toolchain installer
Using the Standard SDK (Cross-Toolchain + Makefile/Autotools/Autotools lib + App | recipes)
Building/Using the Extensible SDK

Package Management

> SW update vs. Package Management | Working with Packages | IPK | creating a package feed | installing a package with opkg on the target

Licensing

> Intro | Add custom license | Add commercial license | Firmware License | Open Source License Compliance

Devtool

> Intro | Add recipe/Build/Deploy | Create/Add layer | Finish | Modify/Update-Recipe | Build/Run | Build Image

Related Courses

> “Embedded Linux - From System Architecture to Real-Time (5 days)”
  https://rlbl.me/elisa
  https://rlbl.me/elisa-en-pdf

> “Refresher to Embedded Linux & Intro to the Yocto Project (5 days)”
  https://rlbl.me/intely
  https://rlbl.me/intely-r-en-pdf

> “Embedded Linux Kernel Internals and Device Drivers (5 days)”
  https://rlbl.me/ldd
  https://rlbl.me/ldd-en-pdf

> “The Yocto Project - A thorough Overview (4 days)”
  https://rlbl.me/yocto
  https://rlbl.me/yocto-r-en-pdf

> “Introduction to Embedded Linux & Real-Time, bird’s eye view of the Yocto Project (4 days)”
  https://rlbl.me/entirety
  https://rlbl.me/entirety-en-pdf

> “Embedded Linux Hardware Interfacing (4 days) - coming soon”
  https://rlbl.me/elisha

> “Compact Linux Driver development (4 days) - coming soon”
  https://rlbl.me/cold

> “Introduction to Embedded Linux in Theory and Practice - a Crash Course (3 days)”
  https://rlbl.me/elin
  https://rlbl.me/elin-en-pdf

> “(Embedded) Linux debugging (3 days) ”
  https://rlbl.me/linde
  https://rlbl.me/linde-r-en-pdf
References

[1] “The rise of copyright trolls”
   https://lwn.net/Articles/721458/

[2] “Target Hardware”
   https://rlbl.me/hw

   https://rlbl.me/elisa

[4] “Introduction to Embedded Linux in Theory and Practice - a Crash Course (3 days)”
   https://rlbl.me/elin

[5] “Embedded Linux Kernel Internals and Device Drivers (5 days)”
   https://rlbl.me/ldd

   https://rlbl.me/delivery

   https://rlbl.me/contact
Since 1993, Robert Berger gathered practical and managerial experience in software design and development for embedded systems with and without hard real-time requirements. Since the beginning of the 21st century, he has used Linux on desktop and server class machines, but mainly for embedded practices (automotive, industrial control, robotics, telecoms, consumer electronics, etc.). Robert regularly attends international events such as “Embedded World”, “Embedded Software Engineering Kongress”, “Embedded Systems Conference”, “Embedded Linux Conference” and “Yocto Project Summit” as an expert and lecturer. His specialty is mainly training, but also consulting (in German or English) worldwide. Robert’s expertise ranges from small real-time systems (FreeRTOS) to systems with multiple processors/cores and embedded Linux (user-, kernel-space, device drivers, hardware interfacing, debugging, multi-core, Yocto Project) with a focus on free and open source software. Robert is a globe-trotter. He is CEO & Embedded Software Evangelist at Reliable Embedded Systems e.U. which is based in St. Barbara, Austria.

Thank you for your interest!

For inquiries please send an email to: training@ReliableEmbeddedSystems.com