



Y1 - Yocto Project Development

Building a Linux Embedded image using Yocto

Objectives

- Using and customizing Yocto
- Creating Yocto-based Embedded Linux platforms
- Using Yocto to develop components

We use a recent Yocto version

Labs are conducted on qemu or on target boards, that can be:

Dual Cortex/A7-based "STM32MP15-DISCO" boards from STMicroelectronics.

Quad Cortex/A9-based "SabreLite" boards from NXP.

Quad Cortex/A53-based "imx8q-evk" boards from NXP.

Prerequisite

- Good C programming skills
- Knowledge of Linux Embedded systems (see our [D1 - Embedded Linux with Buildroot and Yocto](#) course)
- Preferably knowledge of Linux user programming (see our [D0 - Linux user mode programming](#) course)
- You may be interested also by the Yocto project expert ([Y2 - Yocto Project Expert](#) course) or the combined ([Y12 - Comprehensive Yocto Project Usage](#) course)

Course environment

- Printed course material (in English)
- One Linux PC for two trainees.
- One target platform for two trainees

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Day

Introduction to Yocto

- Overview of Yocto
 - History
 - Yocto, Open Embedded and Poky
 - Purpose of the Yocto project
 - The main projects
- Yocto architecture
 - Overview
 - Recipes and classes
 - Tasks

Exercise: Setting up the labs environment

The Yocto build system

- Build system objectives
 - Building deployable images

Exercise: Building a root file system using Yocto

- Build system architecture
 - Layers and layer priorities
 - Directory layout
 - Configuration files
 - ▶ Local
 - ▶ Machine
 - ▶ Distro
 - The bitbake tool
 - Common options
- Using Yocto
 - The bitbake tool
 - ▶ Common options
 - ▶ Base commands
 - Building a package
 - Building an image (root file system + u-boot + kernel)

Exercise: Use bitbake commands to build packages and images

- Miscellaneous tools around Yocto
 - Yocto Application Development Toolkit (ADT)
 - Toaster

Exercise: Deploy the generated image

Yocto recipes structure

- Recipe architecture
 - Tasks
 - Task dependencies
 - Recipe dependencies
- The bitbake language
 - Standard variables and functions
 - Classes and recipes
 - The base Yocto classes
 - Main bitbake commands

Exercise: Examine and understand real-life configuration files

Second Day

Writing package recipes for Yocto

- Various kind of recipes and classes
 - bare program
 - Makefile-based package
 - autotools-based package
 - u-boot
 - kernel
 - out-of-tree module
- Recipe creation strategies
 - From scratch
 - Using devtool
 - Using recipetool
 - From an existing, similar, recipe

Exercise: Writing a recipe for a local user-maintained package

Exercise: Writing a recipe for a local out-of-tree module

- Debugging recipes
 - Debugging recipe selection
 - Debugging dependencies
 - debugging tasks

Exercise: Writing and debugging a recipe for an autotools, git-maintained, package

- Defining packaging
 - Package splitting

Exercise: Writing and debugging a recipe for an autotools library package

- Automatically starting a program (class update-rc.d)

Exercise: Starting an ssh daemon on the target

Modifying recipes

- Customizing an existing package recipe (.bbappend)
- Recipe dependencies
- Creating and adding patches
 - Creating a patch for a community-provided component
 - Creating a patch for an user-maintained component

Exercise: Adding patches and dependencies to a community package

- Defining new tasks
 - Task declaration
 - Coding tasks

Exercise: Adding a rootfsinstall task to directly copy the output of an user package in the rootfs image

Development process using the extensible SDK and devtool

- Using devtool to create a package and its recipe
- Using devtool to modify an existing package and recipe
- Using devtool to update a recipe to build a new version of a package

Exercise: Create, test and modify a recipe for an existing package using devtool

Third Day

Creating new kinds of recipes

- Creating classes
 - Creating new, independent, classes
 - Inheriting from an existing class

Exercise: Create a class to generalize the “rootfsinstall” task

Exercise: Create a class to build firmware packages (for example using an MCU toolchain)

Creating a root file system

- Building a root file system with Yocto
 - Creating a custom root file system
- Writing an image recipe
 - Selecting the packages to build
 - Selecting the file system types
 - The various kinds of images
- Inheriting and customizing images
 - Customizing system configuration files (network, mount points, &)

Exercise: Writing and building an image recipe

Exercise: Creating a JFFS2, UBIFS or EXT2 image with Yocto

- Package management
 - rpm
 - opkg

Exercise: Create an image with package support for OTA deployment

Exercise: Test OTA update on the generated image

Writing tasks in Python

- Introduction to python
- Using python in Yocto
 - The main bitbake classes
 - Defining variable values in Python
 - Writing tasks in Python

Exercise: Writing a task and customizing a recipe in Python