



oY1 - Développement du projet Yocto

Construire une image Linux embarquée en utilisant Yocto

Objectifs

- Utilisation et personnalisation de Yocto
- Créer des plateformes Linux embarquées basées sur Yocto
- Utiliser Yocto pour développer des applications

Les travaux pratiques sont effectués sur une carte ARM QEMU

Nous utilisons une version récente de Yocto

Pré-requis

- Bonnes connaissances en programmation C (voir notre cours [oL2 - Langage C pour les MCUs embarqués](#))
- Connaissance des systèmes embarqués Linux (voir notre cours [oD1 - Linux embarqué](#))
- De préférence, connaissance de la programmation utilisateur Linux (voir notre cours [oD0 - Programmation en mode utilisateur Linux](#))
- Vous pouvez également être intéressé par le [cours Yocto Expert](#) ou le cours combiné (cours [oY12 - Usage complet du projet Yocto](#))

Environnement du cours

- Cours théorique
 - Support de cours au format PDF (en anglais)
 - Cours dispensé via le système de visioconférence Teams
 - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique
- Activités pratiques
 - Les activités pratiques représentent de 40% à 50% de la durée du cours
 - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
 - Un PC Linux en ligne par stagiaire pour les activités pratiques
 - Exemples de code, exercices et solutions
 - Le formateur a accès aux PC en ligne des stagiaires pour l'assistance technique et pédagogique
 - Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante
- Une machine virtuelle préconfigurée téléchargeable pour les activités pratiques après le cours
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus

Durée

- Totale : 18 heures
- 3 sessions, 6 heures +/-30 min chacune (hors temps de pause)
- De 40% à 50% du temps de formation est consacré aux activités pratiques
- Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante

Course Outline

Première session

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

The Yocto build system

- Build system objectives
 - Building deployable images
 - Layers and layer priorities
 - Directory layout
 - Configuration files (local, machine and distribution)
 - The bitbake tool
- Using Yocto
 - Building a package
 - Building an image (root file system + u-boot + kernel)
- Miscellaneous tools around Yocto
 - Yocto SDK
 - Extensible SDK

Exercise : Building a root file system using Yocto

Exercise : Use bitbake commands to build package & images

Exercise : Building a root file system using Yocto

Exercise : Build an extensible SDK for the generated image

Exercise : Deploy the generated image

Yocto package 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
- Adding a new layer
 - Layer structure
 - Various kinds of layers

Exercise : Adding a new layer

Deuxième session

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
- Debugging recipes
 - Debugging recipe selection
 - Debugging dependencies
 - Debugging tasks
- Defining packaging
 - Package splitting
- Automatically starting a program

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

Exercise : Writing and debugging a package recipe for an autotools-based package

Exercise : Starting a program at boot (systemd)

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
- Defining new tasks
 - Task declaration
 - Coding tasks

Exercise : Adding patches and dependencies to a community package

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

Troisième session

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

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, ...)
- Users and groups management
- Package management
 - rpm
 - opkg

Exercise : Writing and building an image recipe

Exercise : Add new users to the image

Exercise : Create an image with package support for OTA deployment

Exercise : Test OTA update on the generated image