Y12 - Usage complet du projet Yocto

Objectifs

- Utilisation et personnalisation de Yocto
- Créer des plateformes Linux embarquées basées sur Yocto
- Utiliser Yocto pour développer des applications
- Personnalisation du BSP
- Construction de modules
- Configuration du cache source

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 L2 - C language for Embedded MCUs)
- Connaissance des systèmes embarqués Linux (voir notre cours D1 - Linux embarqué avec Buildroot et Yocto)
- De préférence, connaissance de la programmation utilisateur Linux (voir notre cours D0 - Programmation en mode utilisateur Linux)

Environnement du cours

- Cours théorique
  - Support de cours imprimé et au format PDF (en anglais)
  - 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 pour deux stagiaires pour les activités pratiques
  - Une plateforme cible pour deux stagiaires (sauf en cas d'utilisation de qemu)
  - Accès à un serveur cloud privé
  - Exemples de code, exercices et solutions
  - Le formateur assiste les stagiaires pendant les exercices
- 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.

Plan du cours

Premier jour

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 jour

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 jour

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

Quatrième jour

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

Develop and debug applications using SDK and eclipse

- Adding eclipse remote debug packages
- Configuring eclipse

Exercise : Create remote debugging session using eclipse

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

Porting Yocto

- Porting Yocto to a new board
- BSP architecture
    - Selecting and configuring u-boot recipe
    - Selecting and configuring kernel recipe
- Adding a new BSP layer (yocto-bsp create)

Exercise : Creating a new BSP layer

Cinquième jour

BSP Development

- Adding a custom u-boot to Yocto
- Customizing the Yocto kernel recipe
    - Setting the default configuration
    - Adding patches
    - Specifying the kernel sources
- Configuring Linux Kernel
    - Using menuconfig
    - Using patches

    - Creating Configuration Fragments
    - Validating Configuration
- Kernel device tree

Exercise : Create u-boot and kernel recipes to use custom versions, test the result
Exercise : Patch kernel and activate new options using a fragment
Exercise : Create and use a new device tree

Out-of-Tree Modules

- Adding modules to image
- Creating an out-of-tree module
- Kernel modules with eSDK

Exercise : Build and test modules

Tailoring the build system

- Setting up a Yocto source cache
    - Local, per system, cache setup
    - Setting up a global, network wide, cache
- Customizing the build system
    - Using a prebuilt toolchain
    - Using a pre-compiled kernel
- Optimizing Yocto build times
    - Using prebuilt, binary, packages
    - Using shared compilation caches

Exercise : Setting up a global source cache
Exercise : Setting up an optimized build environment and rebuilding an image