Y1 - 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 composants

Nous utilisons une version récente de Yocto
les travaux pratiques sont effectués sur qemu ou sur des cartes cibles, qui peuvent être :
Cartes "STM32MP15-DISCO" à base de double Cortex/A7 de STMicroelectronics
Cartes "SabreLite" à base de Quad Cortex/A9 de NXP
Cartes "imx8q-evk" de NXP basées sur le Quad Cortex/A53

Pré-requis

- Bonnes connaissances en programmation C
- 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)
- Vous pouvez également être intéressé par le cours Yocto Expert (cours Y2 - Expert en projet Yocto) ou le cours combiné (cours oY12 - Usage complet du projet Yocto)

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.
  - Exemples de code, exercices et solutions
  - Un PC (Linux ou Windows) par binôme de stagiaires (si plus de 6 stagiaires) pour les activités pratiques avec, si approprié, une carte cible embarquée.
  - Le formateur accède aux PC des stagiaires pour l'assistance technique et pédagogique.
- Une machine virtuelle préconfigurée téléchargeable pour refaire 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.

Modalités d'évaluation

- Les prérequis indiqués ci-dessus sont évalués avant la formation par l'encadrement technique du stagiaire dans son entreprise, ou par le stagiaire lui-même dans le cas exceptionnel d'un stagiaire individuel.

- Les progrès des stagiaires sont évalués de deux façons différentes, suivant le cours:
  - Pour les cours se prêtant à des exercices pratiques, les résultats des exercices sont vérifiés par le formateur, qui aide si nécessaire les stagiaires à les réaliser en apportant des précisions supplémentaires.
  - Des quizz sont proposés en fin des sections ne comportant pas d'exercices pratiques pour vérifier que les stagiaires ont assimilé les points présentés
- En fin de formation, chaque stagiaire reçoit une attestation et un certificat attestant qu'il a suivi le cours avec succès.
  - En cas de problème dû à un manque de prérequis de la part du stagiaire, constaté lors de la formation, une formation différente ou complémentaire lui est proposée, en général pour conforter ses prérequis, en accord avec son responsable en entreprise le cas échéant.

Plan

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

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*

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

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*

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*
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*

Renseignements pratiques

Durée : 3 jours
Prix : 2400 € HT