oY2 - Expert en projet Yocto

Utilisation et adaptation avancées du Yocto Project

Objectifs

- Utilisation et personnalisation de Yocto
- Créer des plateformes Linux embarquées basées sur Yocto
- Utiliser Yocto pour développer des composants
- Construire à partir de modules arborescents
- Configurer le cache des sources

Les travaux pratiques sont effectués sur une carte ARM QEMU
Nous utilisons une version récente de Yocto

Pré-requis

- Bonnes connaissances en programmation C
- Connaissance des systèmes embarqués Linux (voir notre cours oD1 - Linux embarqué)
- Connaissance du développement du projet Yocto (voir notre cours oY1 - Développement du projet Yocto)
- De préférence, connaissance de la programmation utilisateur Linux (voir notre cours oD0 - Programmation en mode utilisateur Linux)

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 : 12 heures
- 2 sessions, 6 heures +/- 30 min chacune (hors temps de pause)
- De 50% à 60% du temps de formation est consacré aux activités pratiques
- Certains travaux pratiquent peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante

Course Outline

Première session

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

Deuxième session

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