UTILISATION DE YOCTO POUR LINUX EMBARQUÉ

AVANCÉ | ATELIER | EN RÉALISABLE EN ANGLAIS

FFCNCERCERXIO25 | PRIX : 2 670 € | DURÉE : 3 JOURS

Pauses et déjeuners offerts

PRÉSENTATION

Yocto est le principal outil de construction d'images "Linux Embarqué". Il est largement utilisé dans l'industrie et les fournisseurs de matériel (modules ARM, cartes) livrent leur "BSP" (Board Support Package) sous la forme de méta données (recettes) Yocto. La maîtrise de l'outil est donc indispensable à qui veut adapter ce BSP à un projet industriel et intégrer des applications à la cible matérielle.
Après une introduction aux principes de "Linux embarqué", nous verrons comment créer ou adapter des recettes et comment utiliser les outils Yocto (SDK, Validation de recettes ou d'images Linux).

OBJECTIFS

- Expliquer les principes de Linux embarqué, Yocto et OpenEmbedded
- Écrire des "recettes" Yocto basées sur les standards de l'Open Source (Autotools, CMake, Pilotes Linux (modules/drivers), Device Tree)
- Personnaliser des recettes Yocto existantes
- Utiliser des outils Yocto (SDK, Devtool, ptest, testimage, etc.)
- Construire un exemple de device Yocto utilisant le protocole MQTT (protocole de référence dans l'IoT)

PROGRAMME

Introduction

Linux embarqué
- Rappels GNU/Linux
- Licences GPL/LGPL
- Présentation de la compilation croisée
- Compilation croisée du noyau Linux
- BusyBox
- Utiliser un "build system" : avantages et inconvénients, principaux outils (Buildroot, Yocto/OpenEmbedded, etc.)

Yocto, avec travaux pratiques
- Historique (OpenEmbedded, Yocto)
- Principaux concepts : BitBake, métadonnées, couches, héritage, etc.
- Création de la distribution core-image minimal pour QEMU/ARM
- Répertoires générés (deploy et work)
- Réglage et optimisation avec local.conf et bblayers.conf
- Création d'une couche de test
- Création d'une recette à l'aide d'un simple «Makefile»
- Utilisation des classes Autotools / CMake
- Gestion des colis (OPKG)
- Dépendances statiques et dynamiques
- Priorité de couche, étendre les recettes (.bbappend) : application de patchs,utilisation de fragments de configuration
- Intégration de l'arborescence des appareils
- Recettes et modules du noyau (utilisant la classe «module»)
- Images personnalisées et la classe « packagegroup »
- Test d'une image personnalisée avec NFS-Root
- Création d'une «distribution» personnalisée
- Construire et utiliser la chaîne d'outils croisée (SDK/eSDK) : débogage à distance avec gdb/gdbserver
- Utilisation de «Devtool»
- Créer un service SysvInit ou Systemd
- Utilisation de CI («ptest» et «testimage»)

Synthèse et conclusion

PUBLIC/PRÉREQUIS

Développeurs Linux et chefs de projets Linux.
Les participants doivent avoir une base technique d'utilisation de Linux comme plateforme de développement, d'utilisation du "shell" Linux (bash) et des notions de base en langage C pour tirer pleinement profit de cette formation.

RESPONSABLE(S)

Stefano ZACCHIROLI
Enseignant-chercheur à Télécom Paris, en biens communs numériques, génie logiciel open source, informatique, sécurité et chaîne d'approvisionnement des logiciels. Co-fondateur & CTO de Software Heritage, la plus grande archive publique de code source. Développeur Debian et ancien projet Debian leader. Ancien administrateur de Open Source Initiative (OSI) et récipiendaire O'Reilly Open Source Award.

Pierre FICHEUX
CTO de la division Smile ECS (Embedded & Connected Systems). Auteur de 5 livres sur Linux embarqué (éditions Eyrolles, de 2002 à 2017) et de livres blancs édités par Smile, consacrés aux logiciels open source pour l'embarqué et l'IoT (Linux embarqué, Android, Linux RT). Il enseigne également le développement des drivers Linux, Linux embarqué et temps réel dans plusieurs écoles d'ingénieurs.

MODALITÉS PÉDAGOGIQUES

Exposés théoriques, travaux pratiques, étude de cas, retours d'expérience d'experts dans l'industrie.
La formation a le gros avantage d'associer un travail pratique à chaque concept présenté. Il n'est pas nécessaire de disposer d'une carte de développement car nous utilisons l'émulateur open source QEMU.
Elle est basée sur une machine virtuelle (VirtualBox) ce qui garantit l'installation sur des PC Linux ou Windows. Elle peut être réalisée sur place, à distance ou de manière hybride.