

UTILISATION DE YOCTO POUR LINUX EMBARQUÉ

FFCNCERCERXIO25

PRIX : 2 680 €

DURÉE : 3 JOURS

Pauses et déjeuners offerts

PRÉSENTATION

Yocto est l'outil de référence pour la construction d'images Linux embarqué. Largement adopté par l'industrie, il est également utilisé par les fabricants de matériel (modules ARM, cartes), qui fournissent leur BSP (Board Support Package) sous forme de métadonnées (recettes) Yocto. Sa maîtrise est donc indispensable pour adapter un BSP aux besoins d'un projet industriel et intégrer des applications sur la plateforme matérielle cible.

Après une introduction aux principes de Linux embarqué, nous verrons comment créer et adapter des recettes, puis comment exploiter les principaux outils de l'écosystème Yocto (SDK, validation de recettes et validation 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 par la pratique

Fondamentaux et architecture de Yocto

- Historique (de l'OpenEmbedded à Yocto)
- Principaux concepts : BitBake, métadonnées, couches, héritage, etc.

Première construction d'une image embarquée

- Création de la distribution core-image-minimal pour QEMU/ARM
- Répertoires générés (deploy et work)
- Réglage et optimisation avec local.conf et bblayers.conf

Développement et organisation des couches

- Création d'une couche de test
- Création d'une recette à l'aide d'un simple « Makefile »
- Utilisation des classes Autotools / CMake

Gestion des paquets et des dépendances

- Gestion des colis (OPKG)
- Dépendances statiques et dynamiques

Personnalisation avancée du système

- Priorité de couche, étendre les recettes (bbappend) : application de patches, 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 »

Déploiement et validation

- 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

Productivité, industrialisation et qualité

- Utilisation de « Devtool »
- Créer un service Sysvinit ou Systemd
- Utilisation de CI (« ptest » et « testimage »)

Synthèse et conclusion



AVANCÉ



ATELIER



RÉALISABLE
EN ANGLAIS



INTERNET DES OBJETS, SYSTÈMES
CONNECTÉS ET LEURS APPLICATIONS

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)

Paul GOULPIÉ

Référent technique chez Smile ECS. Expert technique Linux embarqué avec près de 20 ans d'expérience. Spécialisé dans la création et le durcissement de distributions Yocto, l'intégration BSP, le Secure Boot et la cybersécurité embarquée. Il intervient au quotidien sur des problématiques complexes de kernel, CI/CD GitLab et chaînes de confiance, avec des contributions actives dans l'écosystème open source Yocto. Il collabore directement avec des fabricants tels que NXP, STMicroelectronics et NVIDIA.

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.