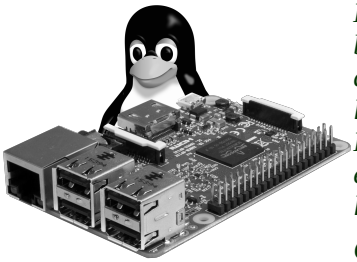
Logilin formations

Linux embarqué avec Yocto Project



L'utilisation croissante de Linux dans les systèmes embarqués va de pair avec un besoin de fiabilité et de pérennité de l'installation. Pour cela ***Yocto Project*** *est aujourd'hui le système de construction le plus adapté aux nécessités de l'environnement industriel. Héritier d'outils réputés comme Buildroot ou PXTdist, il offre une richesse incomparable en terme d'applications disponibles et des possibilités inégalées pour garantir la pérennité et la portabilité de l'environnement Linux embarqué produit.*

Ce cours vous propose de découvrir les rouages de Yocto Project, d'en maîtriser l'utilisation courante (génération et personnalisation d'images Linux embarqué) afin de développer des applications pour Linux embarqué.

Si vous avez déjà une bonne pratique de Yocto Project, nous vous conseillerons plutôt notre cours « Yocto Project Avancé », qui explore en profondeur des points supplémentaires de l'utilisation de Yocto (classes, distro, devtool, machine, device tree, wks, etc.)

Organisation

Audience

Nous limitons habituellement le nombre de stagiaires dans nos sessions à 4 personnes au maximum pour garantir des échanges fluides et conviviaux.

Les sessions à distance se déroulent sur **plateforme Zoom**. Le seul matériel nécessaire est un ordinateur avec une connexion Internet et un micro. Nous conseillons un ensemble casque + micro pour limiter le bruit de fond. Nous suggérons également l'emploi d'une webcam si l'environnement le permet.

Pré-requis

Connaissance de Linux (niveau utilisateur).

Durée

3 jours (21 heures)

Travaux pratiques

Les exercices se déroulent sur des PC Linux accessibles à distance (connexion SSH / PuTTY / Tera Term) et émulateur Qemu. Les démonstrations sont présentées sur cartes Raspberry Pi 4.

Thèmes abordés

Créer un système Linux embarqué avec Yocto Project : environnement Linux embarqué, production d'une image, analyse de son contenu.

Personnalisation de l'image : configuration du système, ajout de *packages* divers.

Configuration avancée : extension de recettes, création de *patches*, *kernel* et *device tree*.

Développement du code métier : installation du SDK, compilation du code applicatif, débogage et mise au point, intégration du code métier dans l'image.

Plan détaillé

I – Créer un système Linux embarqué avec Yocto Project

Environnement Linux embarqué

Concepts, composant, outils de génération, Buildroot et Yocto Project.

Production d'une image standard

Environnement de travail, Poky, configuration, *layers* spécifiques, outil *bitbake*…

Travaux pratiques : préparation de l'environnement de Yocto, production d'une image pour émulateur Arm, pour carte BeagleBone, pour Raspberry Pi.

Composition d'un système Linux embarqué

Matériel, *bootloader*, noyau Linux, processus init, scripts de démarrage.

Travaux pratiques : Lancement de l'image sur l'émulateur ARM, installation de l'image sur cible matérielle.

II - Personnalisation du système embarqué

Découverte et analyse du système

Connexion, systèmes de fichiers, arborescence standard, *boot*.

Personnalisation de l'image

Recette d'image personnelle, administration du système, syntaxe des recettes

Travaux pratiques : création d'une image personnalisée, configuration des utilisateurs et mots de passe.

Ajout de packages

Packages standards de Yocto, layer de OpenEmbedded, configuration de Busybox

Travaux pratiques : ajout de packages de Poky, du dépôt méte-openembbeded, personnalisation de Busybox, configuration du rootfs en lecture-seule.

III - Configuration avancée du système

Extension de recettes

Fichiers *.bbappend*, surcharge de fichiers de recettes, configuration réseau statique.

Travaux pratiques : surcharge d'un fichier de recette

Création et application de patches

Remplacement d'un fichier de donnée, modification d'un fichier source à compiler

Travaux pratiques : création d'un patch sur un fichier de recette, sur un fichier compilé.

Noyau Linux et *Device Tree*

Choix et configuration du noyau, type et version du kernel, paramétrage, principe du *Device Tree*.

Travaux pratiques : Configuration du noyau standard, patch pour le kernel, patch pour le device tree

IV - Développement du code métier

Intégration de scripts personnalisés

Cross-compilation du code métier

Chaîne de compilation Gnu GCC

Travaux pratiques : extraction de la toolchain.

Débogage distant

Gdbserver, outils d'aide au débogage

Travaux pratiques : débogage avec Gdbserver

Intégration du code métier

Recettes, nom, contenu

Travaux pratiques : développer une recette pour son code métier

Lancement d’un application au démarrage

Scripts de démarrage

Travaux pratiques : intégration d’une application au démarrage

Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

Travaux pratiques : expérimentations libres suivant les demandes des stagiaires.

MàJ 28/12/2023