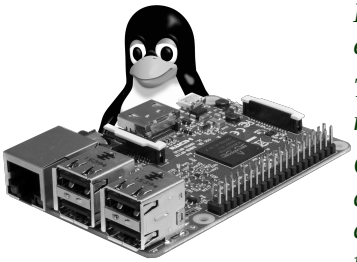
logilin
Logilin formations

Yocto Project avancé



Le système de construction Yocto Project permet d'obtenir des systèmes embarqués basés sur Linux robustes et fiables.

Toutefois la structure même de Yocto Project en fait un outil complexe à maîtriser.

Cette formation est destinée à des personnes ayant déjà une première expérience avec Yocto, et souhaitant approfondir leurs connaissances en explorant des aspects avancés de cette puissante plateforme. Les participants idéaux pour cette formation sont des développeurs embarqués, des ingénieurs système, des responsables de projet et toute personne travaillant dans le domaine des systèmes embarqués Linux. Préalablement familiarisés avec les concepts de base de Yocto Project, les participants auront l'opportunité d'élargir leurs compétences en se plongeant dans des sujets plus avancés

Si vous souhaitez une formation de découverte et prise en main de Yocto, nous vous conseillerons plutôt notre cours « Linux embarqué avec Yocto Project », qui développe les concepts de base considérés ici comme acquis.

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 basique de Yocto Project (savoir créer une image standard, connaître les notions de layer, de recettes, d'image, savoir utiliser bitbake).

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

Rappels sur Yocto Project : *build systems*, arborescence de travail, commandes principales.

Utilisation de devtool : recherche d'information, création et maintenance de recette, de patch.

Approfondissement des recettes : classes, distro, PACKAGECONFIG.

Support du matériel : machine, bootloader, kernel, device tree, partitionnement, overlayfs.

Plan détaillé

I – Rappels sur Yocto Project

Linux embarqué et *build systems*

Concepts, comparatif avec Buildroot.

Arborescence de travail recommandé

Emplacement des layers, des builds, des répertoires *downloads* et *shared-state cache*

Travaux pratiques : préparation de l'environnement et production d'une image standard.

Rappels des commandes principales

bitbake, *bitbake-layers*, *devtool*, *recipetool*, *runqemu*. Contenu typique d'un layer.

Travaux pratiques : production d'une image au contenu customisée sur un layer personnel.

II – Utilisation de Devtool

Recherche d'informations

Recherche de recettes, d'emplacements, de versions, d'extensions de recettes.

Création de recette

Création de recette pour *Autotools*. Lancement au boot avec *sysVinit*, avec *Systemd*, projet *Early-init*.

Travaux pratiques : création de recettes pour *Cmake* et *Makefile*. Démarrage automatique au boot.

Création de patches

Utilisation de *devtool* pour création de patch. Ajout de patches successifs.

Travaux pratiques : création de patches sur *busybox* et le *kernel* Linux.

III – Approfondissement du contenu des recettes

Classes

Fichiers de classes, héritage, utilisations, classes intéressantes.

Travaux pratiques : création d'une classe personnelle et héritage dans une recette

Distro

Rôle et choix de *distro*, utilisation de *DISTRO_FEATURES*, choix de la *libC* et du système d'initialisation.

Travaux pratiques : création d'une distro personnalisée et utilisation dans une image.

PACKAGECONFIG

Principe de PACKAGECONFIG, sélection des options.

Travaux pratiques : Configuration des options pour compiler *Nano*.

IV – Support du matériel (BSP)

Machine

Fichier de configuration *machine*, variables concernées, *kernel*, *bootloader*, types d'image.

Travaux pratiques : fichier de configuration pour une machine personnalisée

Bootloader

Choix du bootloader, structure de U-boot, variables d'environnements et scripts.

Travaux pratiques : patch sur U-boot pour ajouter des variables personnalisées.

Kernel et device tree

Choix de version, configuration, *fragments* et *defconfig*, principe du device tree.

Travaux pratiques : Config kernel et intégration dans le fichier *machine, device tree* personnalisé.

Partitionnement et overlay

Principe et syntaxe des fichiers WKS. Classe *etc-overlayfs* ou script personnel.

Travaux pratiques : ajout partition de données en lecture-écriture et montage d'un *overlay* sur */etc*.

Conclusion

Discussions libres sur l'ensemble des thèmes abordés.

Travaux pratiques : expérimentations libres suivant les demandes des stagiaires.

MàJ 16/02/2024