Développement de systèmes Linux embarqué

Formation sur site, 5 jours

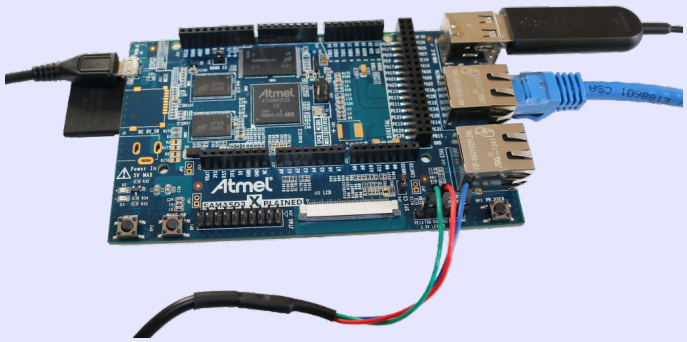
Titre	**Développement de systèmes Linux embarqué**
Objectifs	• Comprendre l'architecture générale des systèmes Linux embarqué. • Comprendre le rôle et le fonctionnement d'une chaîne de compilation croisée, et mettre en place sa propre chaîne de compilation. • Comprendre le processus de démarrage d'un système embarqué, les principaux chargeurs de démarrage, et mettre en place son propre chargeur de démarrage. • Comprendre le rôle et l'architecture générale du noyau Linux, comment le configurer, le compiler et l'installer sur un système embarqué. • Comprendre le principe et le contenu du système de fichiers racine dans Linux, et créer son propre système de fichiers racine à partir de zéro. • Découvrir les différents systèmes de fichiers pour périphériques de stockage en mode bloc et flash, et les utiliser dans un système embarqué. • Découvrir les principaux composants logiciels open-source pour les systèmes embarqués, comprendre les contraintes de licences, comment intégrer et cross-compiler des composants open-source, et expérimenter avec la compilation croisée de bibliothèques • Découvrir les principaux systèmes de build Linux embarqué, et expérimenter l'un d'entre eux. • Comprendre les principes et outils pour le développement et le debug d'applications sur systèmes Linux embarqué. • Découvrir les différentes solutions disponibles pour le temps-réel dans les systèmes Linux embarqué.
Supports	Vérifiez que le contenu de la formation correspond à vos besoins : `https://bootlin.com/doc/training/embedded-linux`.
Durée	**Cinq** jours - 40 h (8 h par jour) 50% de présentations et 50% de travaux pratiques.
Formateur	Un des ingénieurs mentionnés sur : `https://bootlin.com/training/trainers/`
Langue	Présentations : Français Supports : Anglais

Public ciblé	Ingénieurs développant des systèmes embarqués reposant sur Linux et des composants open-source.
Pré-requis	**Connaissance et pratique des commandes UNIX ou GNU/Linux** Les personnes n'ayant pas ces connaissances doivent s'autoformer, par exemple en utilisant nos supports de formation disponibles en ligne : `https://bootlin.com/blog/command-line/`
Variante	Version réduite de la formation au développement de systèmes Linux embarqué, (**durée de 4 jours**), dans laquelle ont été supprimés les sujets suivants : • Gestion de stockage de type flash et systèmes de fichiers spécialisés • Implémentation de contraintes temps-réel avec Linux `https://bootlin.com/doc/training/embedded-linux/embedded-linux-4d-agenda-fr.pdf`.
Équipement nécessaire	**Pour les sessions en présentiel dans les locaux de nos clients, notre client doit fournir**: • Projecteur vidéo • Un ordinateur sur chaque bureau (pour une ou deux personnes), avec au moins 8 Go de RAM et Ubuntu Linux 20.04 installé dans une **partition dédiée d'au moins 30 Go**. • Les distributions autres que Ubuntu Linux 20.04 ne sont pas supportées, et l'utilisation de Linux dans une machine virtuelle n'est également pas supportée. • **Connexion à Internet rapide et sans filtrage**: au moins 50 Mbit/s de bande passante en téléchargement, et pas de filtrage des sites Web et protocoles. • **Les ordinateurs contenant des données importantes doivent être sauvegardés** avant d'être utilisés dans nos sessions.
Supports	Copie électronique des présentations et travaux pratiques. Version électronique des données pour les travaux pratiques.

Matériel

La plateforme matérielle utilisée pendant les travaux pratiques de cette formation est la carte SAMA5D3 Xplained de Microchip, dont voici les caractéristiques :

- Un processeur ARM Cortex A5 de Microchip (SAMA5D36)
- Alimenté par USB
- 256 Mo de RAM DDR2
- 256 Mo de flash NAND
- 2 ports Ethernet (Gigabit + 100 Mbit)
- 2 ports USB 2.0 hôte
- 1 port USB device
- 1 port MMC/SD
- Port série 3.3 V (comme Beaglebone Black)
- Connecteur compatible Arduino R3
- Divers : JTAG, boutons, LEDs



1er jour - Matin

Cours – Introduction à Linux embarqué

- Avantages de Linux par rapport aux autres OS pour l'embarqué. Raisons pour choisir Linux.
- Aperçu général : comprendre l'architecture d'un système Linux embarqué. Aperçu des principaux éléments dans un système typique.

Le reste de la formation étudie chacun de ces éléments en détail.

Cours - Environnement de développement	Cours - Chaîne de compilation croisée et bibliothèque standard C
• Système d'exploitation et outils sur la station de travail pour le développement de systèmes Linux embarqué. • Astuces pour l'utilisation de Linux sur station de travail.	• Les composants d'une chaîne de compilation croisée. • Choisir une bibliothèque standard C. • Le contenu de la bibliothèque standard C. • Les chaînes de compilation croisée prêtes à l'emploi. • La construction automatisée d'une chaîne de compilation croisée.

1er Jour - Après-midi

TP – Chaîne de compilation croisée	Cours – Chargeurs de démarrage
• Configuration de Crosstool-NG • Exécution pour construire une chaîne de compilation croisée personnalisée reposant sur la uClibc.	• Chargeurs de démarrage existants • Fonctionnalités des chargeurs de démarrage • Installation d'un chargeur de démarrage • Focus sur U-Boot

TP - U-Boot	Cours – Noyau Linux
Utilisation de la carte SAMA5D3 Xplained de Microchip • Mise en place de la communication série avec la carte. • Configuration, compilation et installation du chargeur de démarrage de premier niveau et d'U-Boot sur la carte Xplained. • Familiarisation avec l'environnement et les commandes d'U-Boot. • Mise en place de la communication TFTP avec la carte. Utilisation des commandes TFTP d'U-Boot.	• Rôle et architecture générale du noyau Linux. • Fonctionnalités disponibles dans le noyau Linux, en insistant sur les fonctionnalités utiles dans les systèmes embarqués. • L'interface entre le noyau et les applications. • Récupérer les sources. • Comprendre les versions du noyau. • Utilisation de la commande patch.

2ème jour - Matin

TP - Sources du noyau	Cours – Configuration et compilation du noyau Linux
• Téléchargement des sources • Application de patches	• Configuration du noyau. • Utilisation de configurations prêtes à l'emploi pour des cartes embarquées. • Compilation du noyau. • Fichiers générés à l'issue de la compilation. • Utilisation des modules noyau.

TP - Compilation croisée du noyau et démarrage sur la carte

Utilisation de la carte Xplained de Microchip

- Configuration du noyau Linux et compilation croisée pour la carte ARM.
- Mise en place d'un serveur TFTP sur la station de développement.
- Téléchargement du noyau en utilisant le client TFTP d'U-Boot.
- Démarrage du noyau depuis la RAM.
- Copie du noyau vers la flash et démarrage depuis la flash.
- Stockage des paramètres de démarrage en flash et automatisation du démarrage du noyau.

2ème jour - Après-midi

Cours – Système de fichier racine

- Les systèmes de fichiers dans Linux.
- Rôle et organisation du système de fichiers racine.
- Localisation de ce système de fichiers: sur espace de stockage, en mémoire, sur le réseau.
- Les fichiers device, les systèmes de fichiers virtuels.
- Contenu type d'un système de fichiers.

Cours - BusyBox

- Présentation de BusyBox. Intérêt pour les systèmes embarqués.
- CConfiguration, compilation et installation.

TP – Construction d’un minuscule système Linux embarqué avec BusyBox

Utilisation de la carte Xplained de Microchip

- Construction à partir de zéro d’un système de fichiers racine contenant un système Linux embarqué
- Mise en place d’un noyau permettant de démarrer le système depuis un répertoire mis à disposition par la station de développement au travers de NFS.
- Passage de paramètres au noyau pour le démarrage avec NFS.
- Création complète du système de fichiers à partir de zéro : installation de BusyBox, création des fichiers spéciaux pour les périphériques.
- Initialisation du système en utilisant le programme init de BusyBox.
- Utilisation du serveur HTTP de BusyBox.
- Contrôle de la cible à partir d’un navigateur Web sur la station de développement.
- Mise en place des bibliothèques partagées sur la cible et développement d’une application d’exemple.

3ème jour - Matin

Cours - Système de fichiers bloc

- Systèmes de fichiers pour périphériques bloc.
- Utilité des systèmes de fichiers journalisés.
- Systèmes de fichiers en lecture seule.
- Systèmes de fichiers en RAM.
- Création de chacun de ces systèmes de fichiers.
- Suggestions pour les systèmes embarqués.

TP - Système de fichiers bloc

En utilisant la carte ARM Xplained

- Créer des partitions sur le stockage bloc.
- Démarrage d’un système avec un assemblage de plusieurs systèmes de fichiers : SquashFS pour les applications, ext3 pour la configuration et les données utilisateur et tmpfs pour les fichiers temporaires.

3ème jour - Après-midi

Cours - Système de fichiers pour flash

- Le sous-système Memory Technology Devices du noyau Linux.
- Les systèmes de fichiers pour le stockage MTD : JFFS2, YAFFS2, UBIFS.
- Options de configuration du noyau.
- Partitions MTD.
- Etude en détail de la meilleure solution du moment, UBI et UBIFS: préparation, flashage et mise en oeuvre d'images d'espace UBI.

TP – Systèmes de fichiers pour flash

Sur la carte ARM Xplained

- Définition de partitions dans U-Boot pour votre stockage flash interne, au lieu d'utiliser des offsets bruts.
- Partage de ces définitions avec Linux.
- Creation d'une image UBI sur votre station de travail, flashage depuis U-Boot et démarrage de Linux sur un des volumes UBI via le système de fichiers UBIFS.

4ème jour - Matin

Cours – Réutilisation de composants open-source existants pour le système embarqué

- Motivations pour la réutilisation de composants existants.
- Trouver et choisir des composants libres et open-source existants.
- Les licences de Logiciels Libres et leurs conditions.
- Aperçu de composants typiquement utilisés dans les systèmes Linux embarqués : bibliothèques et systèmes graphiques (framebuffer, GTK, Qt, etc.), utilitaires système, bibliothèques et utilitaires réseau, bibliothèques multimédia, etc.
- Construction du système et intégration des composants.

Cours – Compilation croisée de bibliothèques et d'applications

- Configuration, compilation croisée et installation de bibliothèques et d'applications pour un système embarqué
- Détails sur le système de compilation utilisé dans la plupart des composants open-source.
- Aperçu des principaux problèmes rencontrés lors de la réutilisation des composants.

4ème jour - Après-midi

TP – Compilation croisée de bibliothèques et d'applications.

Si suffisamment de temps disponible

- Construction d'un système avec les bibliothèques ALSA et une application de lecture audio.
- Compilation et installation manuelle de plusieurs composants open-source.
- Apprentissage des principales techniques et des problèmes principaux.

Cours - Outils de construction de systèmes

- Outils de la communauté pour la construction automatisée de systèmes.
- Exemple de Buildroot.

TP - Construction d'un système avec Buildroot

Utilisation de la carte Xplained de Microchip

- Utilisation de Buildroot pour construire de façon automatisée un système similaire à celui du TP précédent.
- Voir à quel point cela est plus simple
- Optionnel: rajout d'un paquet dans Buildroot

5ème jour - Matin

Cours - Développement et déboguage d'application

- Langages de programmations et bibliothèques disponibles.
- Aperçu de la bibliothèque C pour le développement d'applications.
- Systèmes de construction pour votre application, comment utiliser des bibliothèques existantes dans votre application.
- Environnements de développement intégrés (IDE) et lecteur de code source.
- Débogueurs : déboguage d'applications à distance avec gdb et gdbserver, analyse post-mortem d'une application.
- Analyseurs de code, analyseurs mémoire, outils de profiling.

TP – Développement et déboguage d'application

Sur la carte Xplained de Microchip

- Développement et compilation d'une application basée sur la bibliothèque ncurses.
- Utilisation de strace, ltrace et gdbserver pour déboguer une application de mauvaise qualité sur le système embarqué

5ème jour - Après-midi

Cours - Linux et le temps réel

Utile pour de nombreux types de systèmes, industriels ou multimédia.

- Comprendre les sources de latence dans le noyau Linux standard.
- Solutions temps réel mou : améliorations apportées au noyau mainline
- Comprendre et utiliser les patches RT preempt pour le noyau Linux.
- Déboguage temps-réel du noyau. Mesure et analyse de la latence.
- Xenomai, une solution temps réel dur pour Linux : fonctionnalités, concepts, implémentation et exemples.

TP - Tests de latence Linux

- Tests sur la carte ARM Xplained.
- Mesure de latence sur Linux standard.
- Mesure de latence sur un noyau Linux incluant les patches `PREEMPT_RT`.
- Mise en place de Xenomai.
- Mesure de latence avec Xenomai.