Formation au développement de systèmes Linux embarqué

Séminaire de formation en ligne

Titre	**Formation au développement de systèmes Linux embarqué**
Aperçu	Chaînes de compilation croisée, bibliothèques standard C pour l'embarqué. Chargeurs de démarrage (bootloaders). Configuration et compilation du noyau Linux. Applications et bibliothèques légères pour systèmes embarqués Systèmes de fichiers pour stockage de type bloc Gestion de stockage de type flash et systèmes de fichiers spécialisés Outils de développement de systèmes embarqués Linux. Développement et mise au point d'applications sur le système embarqué. Contraintes temps-réel et Linux embarqué. Démonstrations pratiques avec une carte ARM.
Supports	Vérifiez que le contenu de la formation correspond à vos besoins : `https://bootlin.com/doc/training/embedded-linux`.
Durée	**Sept** demi-journées - 28 h (4 h par demi-journée) 75% de présentations et 25% de démonstrations.
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 peuvent s'autoformer, par exemple en utilisant nos supports de formation disponibles en ligne : `https://bootlin.com/blog/command-line/`

Équipement nécessaire	• Ordinateur avec le système d'exploitation de votre choix, équipé du navigateur Google Chrome ou Chromium pour la conférence vidéo. • Une webcam et un micro (de préférence un casque avec micro) • Une connexion à Internet à haut débit
Supports	Version électronique des présentations, des instructions et des données pour les démos et travaux pratiques.

Matériel virtuel

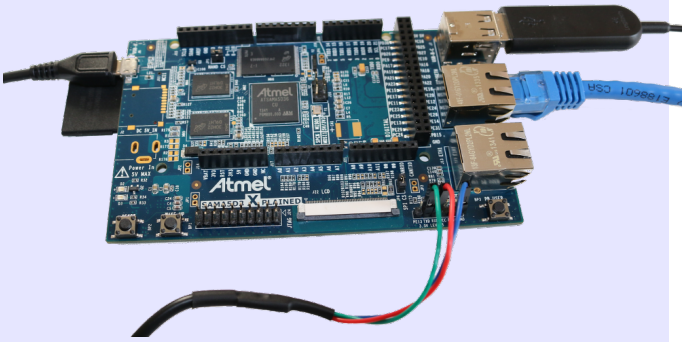
Des travaux pratiques facultatifs (entre les sessions) sont proposés sur carte ARM Vexpress A9 board émulée par QEMU.

Crédits image (Wikipedia) : `https://frama.link/mW71eosa`

Matériel pour démonstrations pratiques

La plateforme matérielle utilisée pendant les démonstrations 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



1ère demi-journée

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

- 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.

Cours - Chaîne de compilation croisée et bibliothèque standard C

- 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.

Pendant le temps entre les sessions

TP facultatif – Chaîne de compilation croisée

- Configuration de Crosstool-NG
- Exécution pour construire une chaîne de compilation croisée personnalisée reposant sur la uClibc.

2ème demi-journée

Démo de solution - Chaîne de compilation croisée

Correspondant au TP facultatif ci-dessus

Cours – Chargeurs de démarrage

- Chargeurs de démarrage existants
- Fonctionnalités des chargeurs de démarrage
- Installation d'un chargeur de démarrage
- Focus sur U-Boot

Cours – Noyau Linux

- 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.

Cours – Configuration et compilation du noyau Linux

- 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.

Pendant le temps entre les sessions

TP facultatif - U-boot

Utilisation de la carte ARM Vexpress A9 émulée par QEMU

- Configuration d'U-Boot pour votre carte virtuelle
- Compilation d'U-Boot avec votre nouvelle chaîne de compilation croisée
- Démarrage d'U-Boot directement par QEMU et tests
- Stockage de l'environnement d'U-Boot dans une carte SD émulée
- Mise en réseau de votre PC et de la machine émulée par QEMU
- Mise en oeuvre de tftp pour le transfert de fichiers entre le PC et U-Boot dans QEMU
- Setup tftp for transferring files between the host and U-Boot

TP facultatif - Sources du noyau

- Téléchargement des sources
- Application de patches

Démo - Compilation croisée du noyau et démarrage sur la carte

Utilisation de la carte ARM Vexpress A9 émulée par QEMU

- Configuration du noyau Linux et compilation croisée pour la carte ARM.
- Téléchargement du noyau en utilisant le client TFTP d'U-Boot.
- Démarrage du noyau depuis la RAM.
- Automatisation du démarrage du noyau dans U-Boot.

3ème demi-journée

Démo - U-Boot

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.

Démo - Sources du noyau

- Téléchargement des sources
- Application de patches

Démo - 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.

Cours – Système de fichiers 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.

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.

Pendant le temps entre les sessions

Démo – Construction d'un minuscule système Linux embarqué avec BusyBox

Utilisation de la carte ARM Vexpress A9 émulée par QEMU

- 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.

4ème demi-journée

Démo – Construction d'un minuscule système Linux embarqué avec BusyBox

Implémentation de la même fonctionalité qu'avec QEMU, mais sur la carte Microchip Xplained

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.

Pendant le temps entre les sessions

TP facultatif - Systèmes de fichiers bloc

Utilisation de la carte ARM Vexpress A9 émulée par QEMU

- Démarrage d'un système avec un assemblage de plusieurs systèmes de fichiers : SquashFS pour les applications, ext4 pour la configuration et les données utilisateur et tmpfs pour les fichiers temporaires.

5ème demi-journée

Démo – Systèmes de fichiers bloc

Implémentation de la même fonctionalité qu'avec QEMU, mais sur la carte Microchip Xplained

Démo – 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.

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.

Pendant le temps entre les sessions

TP factultatif – Compilation croisée de bibliothèques et d'applications

Utilisation de la carte ARM Vexpress A9 émulée par QEMU

- 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.

6ème demi-journée

Démo – Compilation croisée de bibliothèques et d'applications

Implémentation de la même fonctionalité qu'avec QEMU, mais sur la carte Microchip Xplained

Cours - Outils de construction de systèmes

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

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.

Pendant le temps entre les sessions

Démo - Construction d'un système avec Buildroot

Utilisation de la carte ARM Vexpress A9 émulée par QEMU

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

Démo – Développement et déboguage d'application

Utilisation de la carte ARM Vexpress A9 émulée par QEMU

- 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

7ème demi-journée

Démo – Construction d'un système avec Buildroot

Implémentation de la même fonctionalité qu'avec QEMU, mais sur la carte Microchip Xplained

Démo – Développement et déboguage d'application

Implémentation de la même fonctionalité qu'avec QEMU, mais sur la carte Microchip Xplained

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.

Démo - 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.

Questions / réponses

- Questions et réponses avec les participants à propos des sujets abordés.
- Présentations supplémentaires s'il reste du temps, en fonction des demandes de la majorité des participants.

Temps supplémentaire possible

Du temps supplémentaire (jusqu'à 4 heures) pourrait être proposé si le programme ne tenait pas en 7 demi-journées, selon le temps passé à répondre aux questions des participants.