Développement de systèmes Linux embarqué

Séminaire en ligne, 6 sessions de 4 heures

Dernière mise à jour: 06 September 2022

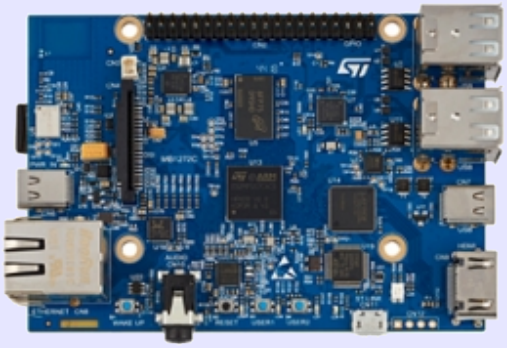
Titre	**Développement de systèmes Linux embarqué**
Objectifs opérationnels	• Être capable d'appréhender l'architecture générale d'un système Linux embarqué. • Être capable de sélectionner, construire, mettre en oeuvre et utiliser une chaîne de compilation croisée. • Être capable de comprendre la séquence d'un démarrage d'un système Linux embarqué et de mettre en oeuvre et d'utiliser le chargeur de démarrage U-Boot. • Être capable de sélectionner une version du noyau Linux, de configurer, de compiler et d'installer le noyau Linux sur un système embarqué. • Être capable de créer à partir de zéro un système de fichiers racine Linux, en comprenant les différents éléments qui le composent: répertoires, applications, bibliothèques, fichiers de configuration. • Être capable de choisir et de mettre en oeuvre les principaux systèmes de fichiers Linux pour périphérique de stockage en mode bloc, et de connaître leurs principales caractéristiques. • Être capable de sélectionner, de cross-compiler et d'intégrer des composants logiciels open-source (bibliothèques, applications) dans un système Linux embarqué, et de traiter la mise en conformité avec les licences open-source. • Être capable de mettre en oeuvre un système de build Linux embarqué, pour construire un système complet pour une plateforme embarquée. • Être capable de développer et débugger des applications sur un système Linux embarqué.
Durée	**Six** demi-journées - 24 h (4 h par demi-journée)
Méthodes pédagogiques	• Présentations animées par le formateur, par visioconférence. Les participants peuvent poser des questions à tout instant. • Démonstrations pratiques réalisées par le formateur, basés sur les travaux pratiques de la formation, par vidéo-conférence. Les participants peuvent poser des questions à tout instant. Optionnellement, les participants qui ont accès aux accessoires matériels de la formation peuvent reproduire par eux-même les travaux pratiques. • Messagerie instantanée pour questions entre les sessions (réponse sous 24h, hors week-end et jours fériés) • Version électronique des supports de présentation, des instructions et des données de travaux pratiques. Les supports sont librement disponibles sur bootlin.com/doc/training/embedded-linux.

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 participants doivent être à l'aise avec l'utilisation de la ligne de commande Linux. Les participants manquant d'expérience sur ce sujet doivent se former par eux-mêmes, par exemple en utilisant nos supports de formation disponible à l'adresse bootlin.com/blog/command-line/. • **Niveau minimal requis en anglais: B1**, d'après le *Common European Framework of References for Languages*, pour nos sessions animées en anglais. Voir bootlin.com/pub/training/cefr-grid.pdf pour une auto-évaluation.
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
Modalités d'évaluation	Seuls les participants qui auront assisté à l'intégralité des journées de formation, et qui auront obtenu plus de 50% de réponses correctes à l'évaluation finale recevront une attestation individuelle de formation de la part de Bootlin.
Handicap	Les participants en situation de handicap qui ont des besoins spécifiques sont invités à nous contacter à l'adresse *training@bootlin.com* afin de discuter des adaptations nécessaires à la formation.

Matériel pour démonstrations pratiques

La plateforme matérielle utilisée pendant les démonstrations de cette formation est la carte *STM32MP157D-DK1 Discovery de STMicroelectronics*, dont voici les caractéristiques :

- Processeur STM32MP157D avec deux coeurs ARM Cortex-A7
- Alimentation par USB-C powered
- 512 Mo de RAM DDR3L
- Port gigabit Ethernet
- 4 ports USB hôte 2.0
- 1 port USB-C OTG
- 1 connecteur micro SD
- Debugger ST-LINK/V2-1 intégré à la carte
- Broches compatibles Arduino Uno v3
- Codec audio
- Divers: boutons, LEDs



Travaux pratiques optionels sur plateforme virtuelle

Pour les participants intéressés, nous proposons des travaux pratiques optionels, à réaliser entre les sessions de formation, qui utilisent la plateforme ARM Vexpress Cortex-A9 émulée par QEMU. S'appuyant sur une plateforme émulée, aucun matériel spécifique n'est nécessaire.

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

1ère demi-journée

Cours – Introduction à Linux embarqué

- Introduction au Logiciel Libre
- Atouts du Logiciel Libre pour les systèmes embarqués
- Exemples de systèmes embarqués fonctionnant sous Linux
- Besoins en CPU, RAM et stockage
- Choix d'une plateforme matérielle
- Architecture du système: composants principaux
- Différentes tâches pour développer un système embarqué

Cours - Environnement de développement

- Système d'exploitation et outils sur la station de travail pour le développement de systèmes Linux embarqué.

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.

Démonstration – Chaîne de compilation croisée

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

Cours – Chargeurs de démarrage

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

2ème demi-journée

Démonstration - U-boot

Utilisation de la carte STM32MP1 Discovery Kit 1

- 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.
- 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.

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.
- Processus de publication du noyau Linux. Versions "Long Term Support".
- Utilisation de la commande patch.

Démo - Sources du noyau

- Téléchargement des sources
- Application de patches

Cours – Configuration et compilation du noyau Linux	Démonstration - Compilation croisée du noyau et démarrage sur la carte
• 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.	*Utilisation de la carte STM32MP1 Discovery Kit 1* • 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.

3ème demi-journée

Cours – Système de fichiers racine	Cours - BusyBox
• 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.	• Présentation de BusyBox. Intérêt pour les systèmes embarqués. • CConfiguration, compilation et installation.

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

Utilisation de la carte STM32MP1 Discovery Kit 1

- 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

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.

Démonstration - Systèmes de fichiers bloc

Utilisation de la carte STM32MP1 Discovery Kit 1

- 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.

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.

5ème demi-journée

Cours – Compilation croisée de bibliothèques et d'applications	Démonstration – 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.	*Utilisation de la carte STM32MP1 Discovery Kit 1* • 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

Cours - Outils de construction de systèmes	Démonstration - Construction d'un système avec Buildroot
• Outils de la communauté pour la construction automatisée de systèmes. • Exemple de Buildroot.	*Utilisation de la carte STM32MP1 Discovery Kit 1* • 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

Cours - Développement et déboguage d'application	Démonstration – 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. • Débogueurs : déboguage d'applications à distance avec gdb et gdbserver, analyse post-mortem d'une application. • Outils pour tracer et profiler des applications.	*Utilisation de la carte STM32MP1 Discovery Kit 1* • 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 • Exploitation d'un *core dump* pour identifier à quel endroit une application s'est "plantée".