Formation développement noyau et pilotes Linux

Séminaire de formation en ligne

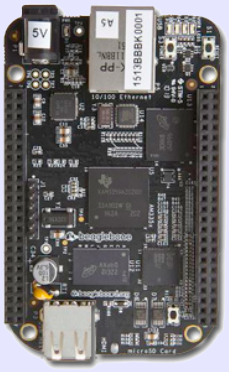
Titre	**Formation développement noyau et pilotes Linux**
Aperçu	Comprendre le noyau Linux Développer des pilotes de périphérique pour le noyau Linux Débogage du noyau Linux Portage du noyau Linux sur un nouveau matériel Travailler avec la communauté de développeurs du noyau Linux Travaux pratiques sur carte électronique ARM BeagleBone Black (ou sa variante ”Wireless”).
Supports	Vérifiez que le contenu de la formation correspond à vos besoins : `https://bootlin.com/doc/training/linux-kernel`.
Durée	**Sept** demi-journées - 28 h (4 h par demi-journée) 80% de présentations et 20% de démonstrations.
Formateur	Un des ingénieurs mentionnés sur : `https://bootlin.com/training/trainers/`
Langue	Présentations : Français Supports : Anglais
Public ciblé	Ingénieurs développant des systèmes reposant sur le noyau Linux. Ingénieurs supportant des développeurs Linux embarqué.
Pré-requis	**Connaissances en programmation en langage C** En particulier, les participants devront comprendre la création et la gestion de types et de structures de données complexes, de pointeurs vers de tels symboles, et de pointeurs de fonctions. **Connaissance 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/`) **Connaissances en développement Linux embarqué**. Suivre au préalable notre Formation Linux Embarqué (`https://bootlin.com/fr/formation/linux-embarque/`) n’est pas un pré-requis, mais sera très utile pour mieux comprendre l’environnement de développement et les manipulations avec le matériel embarqué, pour pouvoir se concentrer sur la programmation du code du noyau Linux.

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.

Matériel

La plateforme matérielle utilisée pendant les démonstrations de cette formation est la carte **BeagleBone Black**, dont voici les caractéristiques :

- Un processeur ARM AM335x de Texas Instruments (à base de Cortex-A8), avec accélération 3D, etc.
- 512 Mo de RAM
- 2 Go de stockage eMMC embarqué sur la carte
  (4 Go avec la révision C)
- USB hôte et device
- Sortie HDMI
- Connecteurs à 2 x 46 broches, pour accéder aux UARTs, aux bus SPI, aux bus I2C, et à d'autres entrées/sorties du processeur.



Démos

Les travaux pratiques de cette formation font appel aux périphériques matériels suivants, pour illustrer le développement de pilotes de périphériques pour Linux :

- Une manette Nunchuk pour console Wii, qui est connectée à la BeagleBone Black via le bus I2C. Son pilote utilisera le sous-système *input* du noyau Linux.
- Un port série (UART) supplémentaire, dont les registres sont mappés en mémoire, et pour lequel on utilisera le sous-système *misc* de Linux.

Bien que nos explications cibleront spécifiquement les sous-systèmes de Linux utilisés pour réaliser ces pilotes, celles-ci seront toujours suffisamment génériques pour faire comprendre la philosophie d'ensemble de la conception du noyau Linux. Un tel apprentissage sera donc applicable bien au delà des périphériques I2C, d'entrée ou mappés en mémoire.

1ère demi-journée

Cours - Introduction au noyau Linux

- Fonctionnalités et rôle du noyau.
- Comprendre le processus de développement du noyau.
- Contraintes juridiques liées aux pilotes de périphériques.
- L'interface noyau / espace utilisateur (/proc et /sys).
- Pilotes de périphériques en espace utilisateur.

Cours - Les sources du noyau

- Spécificités du développement noyau
- Conventions de codage
- Récupération des sources du noyau
- Aperçu des sources du noyau
- Outils de navigation dans les sources : cscope, Elixir

Démo - Code source du noyau

- Récupération des sources du noyau
- Effectuer des recherches dans les sources du noyau Linux : recherche de définitions C, de paramètres de configuration et d'autres informations.
- En utilisant la ligne de commande Unix et des outils de navigation dans le code.

Cours – Configuration, compilation et démarrage du noyau Linux

- Configuration du noyau.
- Compilation native et croisée. Fichiers générés.
- Démarrage du noyau. Paramètres de démarrage.
- Montage du système de fichiers racine par NFS.

2ème demi-journée

Démo – Configuration, compilation croisée et démarrage sur NFS

En utilisant la carte BeagleBone Black

- Configuration, compilation croisée et démarrage du noyau Linux avec support de NFS.

Cours – Modules noyau Linux

- Pilotes de périphériques Linux
- Un module simple
- Contraintes de programmation
- Chargement et déchargement de modules
- Dépendances entre modules
- Ajouter du code source à l'arbre du noyau

Démo - Développement de module

En utilisant la carte BeagleBone Black

- Écriture d'un module noyau offrant quelques fonctionnalités
- Accès aux informations internes du noyau depuis le module
- Mise en place de l'environnement de compilation

Lecture - Le "device model" de Linux

- Comprendre comment le noyau est conçu pour supporter les pilotes de périphériques
- Le "device model"
- Connexion entre périphériques et pilotes.
- Périphériques "platform", le "Device Tree"
- Interface en espace utilisateur avec `/sys`

3ème demi-journée

Démo - Device model de Linux pour un pilote I2C

En utilisant la carte BeagleBone Black

- Réalisation d'un pilote qui s'enregistre comme un pilote I2C.
- Modification du Device Tree pour déclarer un périphérique I2C.
- Faire en sorte que le pilote soit appelé quand le périphérique I2C est énuméré au démarrage.

Cours - Introduction à l'API I2C

- Le sous-système I2C du noyau
- Détails sur l'API fournie aux pilotes du noyau pour interagir avec les périphériques I2C.

Cours - "Pin muxing" (multiplexage d'entrées-sorties)

- Comprendre l'infrastructure *pinctrl* du noyau.
- Comprendre comment configurer le multiplexage des entrées/sorties.

Démo - Communiquer avec le Nunchuk via I2C

En utilisant la carte BeagleBone Black

- Configurer le pin muxing pour le bus I2C utilisé pour communiquer avec le Nunchuk
- Étendre le pilote I2C commencé à la démo précédente pour communiquer avec le Nunchuk à travers le bus I2C.

4ème demi-journée

Cours - Infrastructures du noyau

- Périphériques de type bloc et caractère
- Interaction entre applications en espace utilisateur et le noyau
- Détails sur les pilotes caractère, `file_operations`, `ioctl()`, etc.
- Échange de données vers ou depuis l'espace utilisateur
- Le principe des infrastructures du noyau

Cours - Le sous-système input

- Principe du sous-système *input* du noyau
- API offerte aux pilotes du noyau pour exposer des fonctionnalités de périphériques d'entrée aux applications en espace utilisateur.
- API en espace utilisateur offerte par le sous-système *input*

Démo - Exposer la fonctionnalité du Nunchuk en espace utilisateur

En utilisant la carte BeagleBone Black

- Extension du pilote du Nunchuk pour exposer les fonctionnalités du Nunchuk aux applications en espace utilisateur, comme un périphérique d'entrée.
- S'assurer du bon fonctionnement du Nunchuk via `evtest`

Cours - Gestion de la mémoire

- Linux : gestion de la mémoire. Espaces d'adressages physique et virtuel, séparation noyau et espace utilisateur.
- Implémentation de la gestion de la mémoire dans Linux.
- Allocation avec `kmalloc()`.
- Allocation par pages.
- Allocation avec `vmalloc()`.

Cours - Entrées-sorties avec le matériel

- Enregistrement des plages de ports d'E/S et de mémoire d'E/S.
- Accès aux plages de ports d'E/S et de mémoire d'E/S.
- Barrières mémoire.

Démo - Pilote "platform" minimal et accès à la mémoire d'E/S

En utilisant la carte BeagleBone Black

- Réalisation d'un pilote "platform" minimal
- Modification du Device Tree pour ajouter un nouveau port série.
- Réservation des adresses d'E/S utilisées par le port série.
- Lecture et écriture des registres du périphérique, pour envoyer des caractères sur le port série.

5ème demi-journée

Cours - Le sous-système misc

- Utilité du sous-système *misc* du noyau
- API du sous-système *misc*, à la fois du côté du noyau, et du côté de l'espace utilisateur.

Démo - Pilote de port série en écriture seule

En utilisant la carte BeagleBone Black

- Extension du pilote commencé dans la démo précédente, en enregistrant celui-ci dans le sous-système *misc*
- Implémentation de l'écriture vers le port série en utilisant le sous-système *misc*
- Tests d'écriture depuis l'espace utilisateur

Cours - Processus, ordonnancement, sommeil et interruptions	Démo - Mise en sommeil et gestion d'interruptions dans un pilote de périphérique
• Gestion des processus dans le noyau Linux. • L'ordonnanceur du noyau Linux et la mise en sommeil des processus. • Gestion des interruptions dans les pilotes de périphérique : enregistrement et développement des gestionnaires d'interruption, exécution différée de tâches.	*En utilisant la carte BeagleBone Black* • Ajout de la fonctionnalité de lecture au pilote caractère développé précédemment. • Enregistrement d'un gestionnaire d'interruption. • Attente de la disponibilité de données dans l'opération `read()` • Réveil lorsque les données deviennent disponibles.

6ème demi-journée

Cours - Verrouillage	Démo - Verrouillage
• Problématique de l'accès concurrent à des ressources partagées • Primitives de verrouillage : mutexes, sémaphores, spinlocks. • Opérations atomiques. • Problèmes typiques de verrouillage. • Utilisation du validateur de verrouillage pour identifier les sources de problèmes.	*En utilisant la carte BeagleBone Black* • Ajout de mécanismes de verrouillage au pilote en cours

Cours - Techniques de débogage noyau	Démo - Investigation de bugs noyau
• Débogage avec printk • Utilisation de debugfs • Analyse d'un oops noyau • Utilisation de kgdb, un débogueur noyau • Utilisation des commandes SysRq • Débogage en utilisant une sonde JTAG	*En utilisant la carte BeagleBone Black* • Étude d'un pilote incorrect. • Analyse du message d'erreur et recherche du problème dans le code source.

Cours - Support de cartes et de SoC ARM

- Comprendre l'organisation du code supportant la plateforme ARM
- Comprendre comment le noyau peut être porté vers un nouveau matériel

7ème demi-journée

Cours - Gestion de l'énergie

- Vue d'ensemble des fonctionnalités de gestion d'énergie du noyau Linux.
- Sujets abordés : horloges, mise en veille et réveil, ajustement automatique de la fréquence, économie d'énergie dans la boucle idle, "runtime power management", régulateurs, etc.

Cours - Le processus de développement du noyau Linux

- Organisation de la communauté du noyau Linux
- Le processus de développement : versions bêta, versions stables, versions long-terme, etc.
- Licences et aspects légaux.
- Comment soumettre des contributions de code à la communauté.
- Ressources pour le développement noyau: livres, sites Internet, conférences

Cours - S’il reste du temps

- DMA
- mmap
- Introduction à Git

Démo - S’il reste du temps

- Se familiariser avec git en contribuant à un vrai projet: le noyau Linux
- Envoyez vos patches aux mainteneurs et aux listes de discussion.

Questions / réponses

- Questions / réponses avec les participants autour du noyau Linux
- Des présentations supplémentaires s’il reste du temps, selon les sujets qui intéressent le plus les participants.