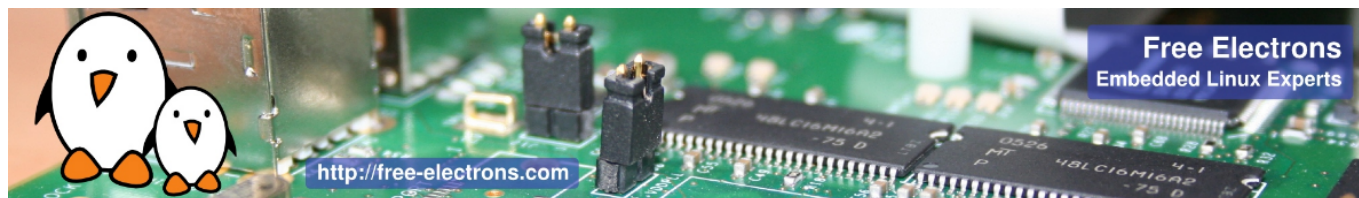


Formation développement noyau et pilotes Linux

Session de 5 jours

Titre	Formation développement noyau et pilotes Linux
Aperçu	<p>Comprendre le noyau Linux</p> <p>Développer des pilotes de périphérique pour le noyau Linux</p> <p>Débogage du noyau Linux</p> <p>Portage du noyau Linux sur un nouveau matériel</p> <p>Travailler avec la communauté de développeurs du noyau Linux</p> <p>Travaux pratiques sur carte électronique ARM BeagleBone Black</p>
Supports	<p>Vérifiez que le contenu de la formation correspond à vos besoins :</p> <p>http://free-electrons.com/doc/training/linux-kernel.</p>
Durée	<p>Cinq jours - 40 h (8 h par jour)</p> <p>50% de présentations et 50% de travaux pratiques.</p>
Formateur	<p>Un des ingénieurs mentionnés sur :</p> <p>http://free-electrons.com/training/trainers/</p>
Langue	<p>Présentations : Français</p> <p>Supports : Anglais</p>
Public ciblé	<p>Ingénieurs développant des systèmes reposant sur le noyau Linux.</p> <p>Ingénieurs supportant des développeurs Linux embarqué.</p>
Pré-requis	<p>Expérience solide en programmation en langage C</p> <p>En particulier, les participants devront maîtriser la création et la gestion de types et de structures de données complexes, de pointeurs vers de tels symboles, et de pointeurs de fonctions.</p> <p>Connaissance et pratique des commandes Unix ou GNU/Linux</p> <p>Les personnes n'ayant pas ces connaissances doivent s'autoformer, par exemple en utilisant nos supports de formations disponibles en ligne : (http://free-electrons.com/docs/command-line/</p> <p>Expérience en développement Linux embarqué.</p> <p>Suivre au préalable notre Formation Linux Embarqué (http://free-electrons.com/fr/formation/linux-embarque/)</p> <p>n'est pas un pré-requis absolu, mais sera très utile à toutes personnes manquant d'expérience en Linux embarqué. Cela leur permettra de mieux comprendre l'environnement de développement et les manipulations mises en oeuvre dans les travaux pratiques, pour pouvoir se concentrer sur la programmation du code du noyau Linux.</p>

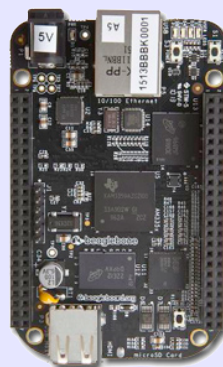


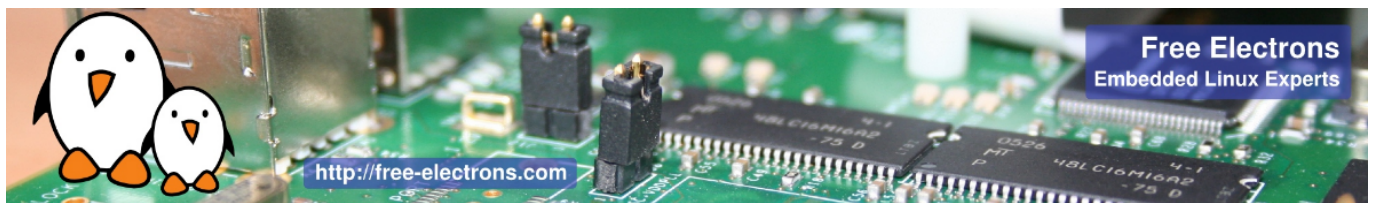
<p>Équipement nécessaire</p>	<p>Pour les sessions sur site uniquement</p> <p>Le matériel est fourni par Free Electrons durant les sessions inter-entreprises</p> <ul style="list-style-type: none"> • Projecteur vidéo • Un ordinateur sur chaque bureau (pour une ou deux personnes), avec au moins 2 Go de RAM et Ubuntu Linux installé dans une partition dédiée d'au moins 20 Go. L'utilisation de Linux dans une machine virtuelle n'est pas supportée, en raison de problèmes avec la connexion au matériel. • Nous avons besoin d'Ubuntu Desktop 16.04 (32 ou 64 bit, Xubuntu et Kubuntu fonctionnent également). Nous ne supportons pas d'autres distributions, car nous ne pouvons tester toutes les versions des paquets. • Connexion à Internet (directe ou par le proxy de l'entreprise). • Les ordinateurs contenant des données importantes doivent être sauvegardés avant d'être utilisés dans nos sessions. Certains participants ont déjà commis des erreurs lors de travaux pratiques avec pour conséquence des pertes de données.
<p>Supports</p>	<p>Versions électroniques et imprimées des présentations et travaux pratiques. Version électronique des données pour les travaux pratiques..</p>

Matériel

La plateforme matérielle utilisée pendant les travaux pratiques 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.





Travaux pratiques

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.

1^{er} jour - Matin

Cours - Introduction au noyau Linux

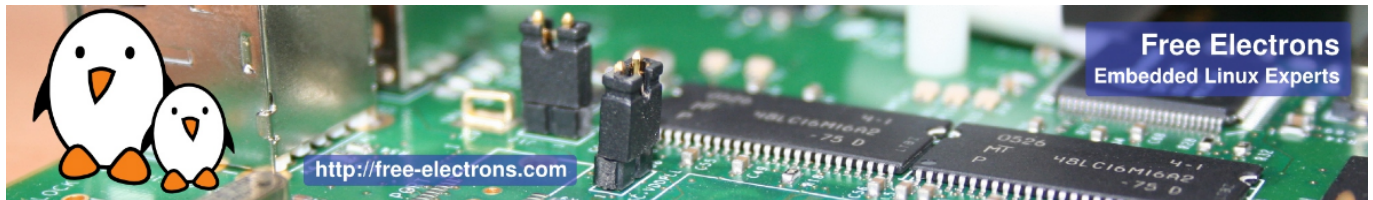
- Fonctionnalités et rôle du noyau.
- Comprendre le processus de développement du noyau.
- Contraintes juridiques liées aux pilotes de périphériques.
- L'interface noyau / espace utilisateur (/proc et /sys).
- Pilotes de périphériques en espace utilisateur.

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, Linux Cross Reference (LXR)

TP - 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.



1^{er} Jour - Après-midi

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.

TP – 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.

2^{ème} jour - Matin

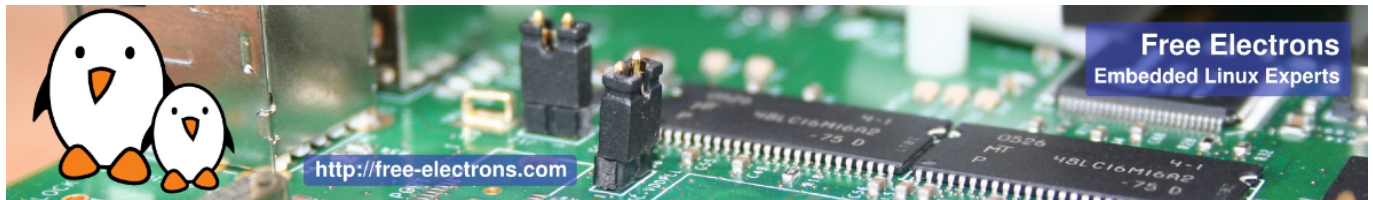
Cours – Modules noyau Linux

- 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

TP - 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



2^{ème} jour - Après-midi

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

TP - 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.

3^{ème} jour - Matin

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.

TP - 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 au TP précédent pour communiquer avec le Nunchuk à travers le bus I2C.



3^{ème} jour - Après-midi

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*

TP - 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 des applications simples en espace utilisateur.

4^{ème} jour - Matin

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.



TP - 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.

4^{ème} jour - Après-midi

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.

TP - Pilote de port série en écriture seule

En utilisant la carte BeagleBone Black

- Extension du pilote commencé dans le TP précédent, 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

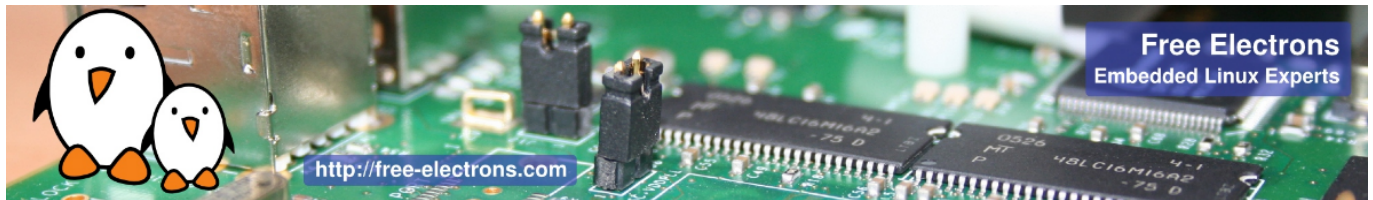
Cours - Processus, ordonnancement, sommeil et interruptions

- 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.

TP - Mise en sommeil et gestion d'interruptions dans un pilote de périphérique

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.



5^{ème} jour - Matin

Cours - 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.

TP - Verrouillage

En utilisant la carte BeagleBone Black

- Observation de problèmes liés à des accès concurrents au périphérique.
- Ajout de mécanismes de verrouillage au pilote pour corriger ces problèmes.

Cours - Techniques de débogage 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

TP - Investigation de bugs noyau

En utilisant la carte BeagleBone Black

- Étude d'un pilote incorrect.
- Analyse du message d'erreur et recherche du problème dans le code source.

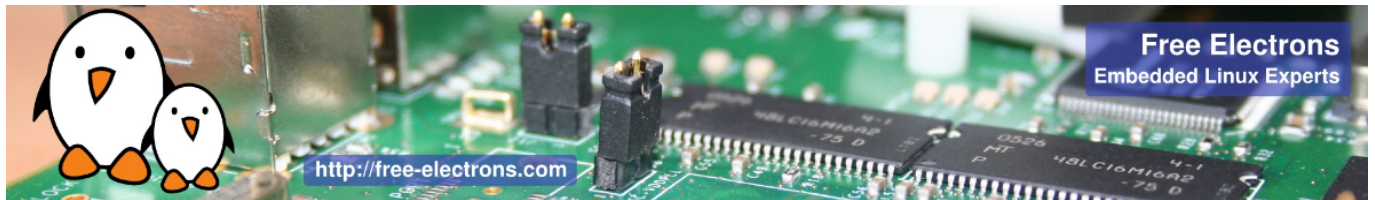
5^{ème} jour - Après-midi

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

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.