

Formation développement noyau et pilotes Linux

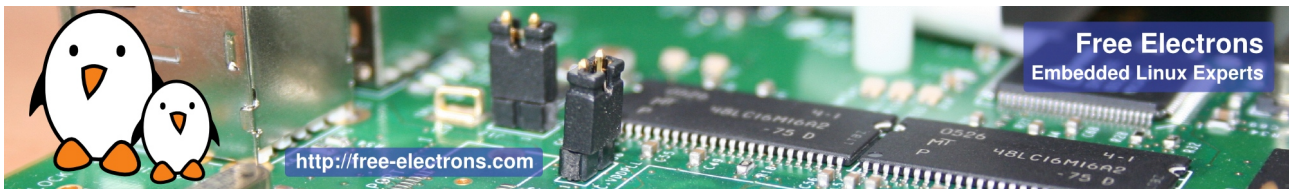
Session de 5 jours

Aperçu

Titre	Formation développement noyau et pilotes Linux pour l'embarqué
Aperçu	Comprendre le noyau Linux Développer des pilotes de périphérique pour le noyau Linux Débogage du noyau Linux Portage du noyau Linux Travailler avec la communauté du noyau Linux Travaux pratiques sur carte ARM et sur machine virtuelle
Durée	5 jours. 50% de présentations et 50% de travaux pratiques.
Formateur	Grégory Clément, Thomas Petazzoni ou Michael Opdenacker (voir http://free-electrons.com/company/staff/)
Langue	Présentations: Français Supports: Anglais
Public ciblé	Ingénieurs développant des systèmes reposant sur le noyau Linux. Ingénieurs supportant des développeurs Linux embarqué.
Pré-requis	Connaissance et pratique des commandes Unix ou GNU/Linux Les personnes n'ayant pas ces connaissances doivent se former en utilisant nos supports de formations disponibles en ligne (http://free-electrons.com/docs/command-line/) Connaissance et pratique de la programmation C Être familier de la configuration, compilation et du démarrage du noyau Linux
Équipement nécessaire	Vidéo-projecteur Un ordinateur sur chaque bureau (pour une ou deux personnes), avec 1 au moins 1 Go de RAM et Ubuntu Linux installé dans une partition d'au moins 10 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'une version 32 bits (i386) d'Ubuntu Desktop 11.04 (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 ayant eu pour conséquence des pertes de données..
Supports	Versions électroniques et imprimées des présentations et travaux pratiques. Version électronique des données pour les travaux pratiques.

Formation de développement noyau et pilotes Linux

Voir nos supports sur <http://free-electrons.com/doc/training/linux-kernel>

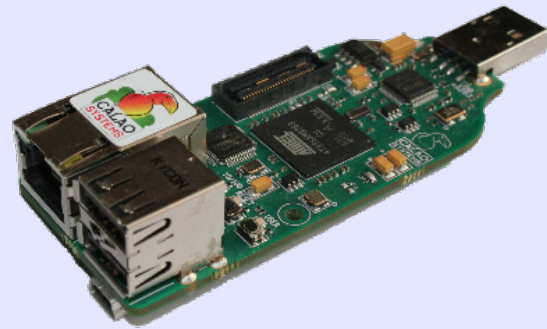


De cette manière, vous pouvez vérifier par vous-même si nos supports correspondent à vos besoins.

Matériel

Utilisation de cartes USB-A9263 de CALAO Systems dans la plupart des travaux pratiques.
Utilisation de systèmes virtuels émulés par QEMU sinon.

CPU ARM AT91SAM9263 d'ATMEL
64 MB RAM, 256 MB flash
2 USB 2.0 host
1 USB device
Port Ethernet 100 Mbit Ethernet
Alimenté par USB!
Port série et JTAG par ce même port USB
De nombreuses cartes d'extension disponibles



Jour 1 - 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)

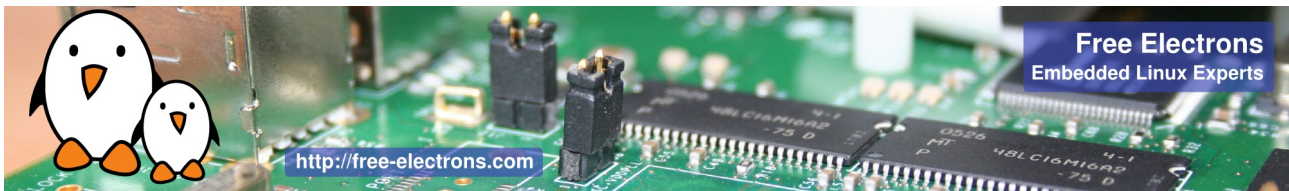
TP - Code source du noyau

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

Jour 1 - Après-midi

Cours - Configuration, compilation et démarrage du noyau Linux

Configuration du noyau
Compilation native. Fichiers générés.
Démarrage du noyau. Paramètres de démarrage.



Cours - Démarrage par NFS et compilation croisée	TP - Configuration, compilation croisée et démarrage via NFS
<p>Montage du système de fichiers racine par NFS. Compilation croisée du noyau.</p>	<p><i>En utilisant la carte CALAO</i> Configuration, compilation croisée et démarrage du noyau Linux avec support du démarrage via NFS.</p>

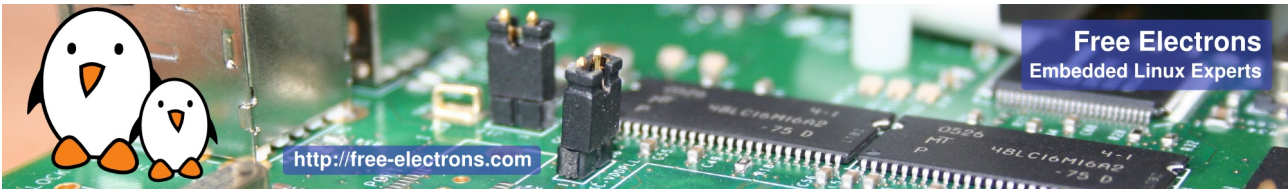
Jour 2 - Matin

Cours - Modules noyau Linux	TP - Développement de module
<p>Pilotes de périphériques Linux Un module simple Contraintes de programmation Chargement et déchargement de modules Paramètres des modules Dépendances entre modules Ajouter du code source à l'arbre du noyau Générer des patches pour les partager avec d'autres</p>	<p><i>Suite du précédent TP</i> Écriture d'un module noyau offrant quelques fonctionnalités, y compris l'utilisation de paramètres. Accès aux informations internes du noyau depuis le module. Mise en place de l'environnement de compilation et de test.</p>

Cours - Gestion de la mémoire
<p>Linux: gestion de la mémoire. Espaces d'adressages physique et virtuel, séparation noyau et espace utilisateur. Allocation avec <code>kmalloc()</code>. Allocation par pages. Allocation avec <code>vmalloc()</code>.</p>

Jour 2 - Après-midi

Cours - Entrées-sorties avec le matériel	TP - Entrées-sorties avec le matériel
<p>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.</p>	<p>Établissement d'une connexion à distance avec votre carte via ssh. Accès à la console système via le réseau. Réservation des plages d'adresses d'E/S utilisées pour le port série. Lecture et écriture de registres du port série, afin d'écrire des caractères sur le port série.</p>



Cours - Pilotes de périphérique caractère	TP - Pilotes de périphérique caractère
<p>Identifiants de périphériques. Allocation d'identifiants disponibles. Implémentation des opérations du pilote: read, write, open, close, ioctl... Échange de données entre l'espace noyau et l'espace utilisateur. <i>Enregistrement de pilotes de périphérique caractère.</i></p>	<p><i>En utilisant la carte ARM Calao</i> Écriture d'un pilote de périphérique caractère simple, pour écrire des données sur le port série Sur votre station de travail, vérifier que les données émises sont bien reçues. Échange de données entre l'espace noyau et l'espace utilisateur. Pratique de l'API pour les pilotes de périphérique caractère. Utilisation des codes d'erreur standard du noyau.</p>

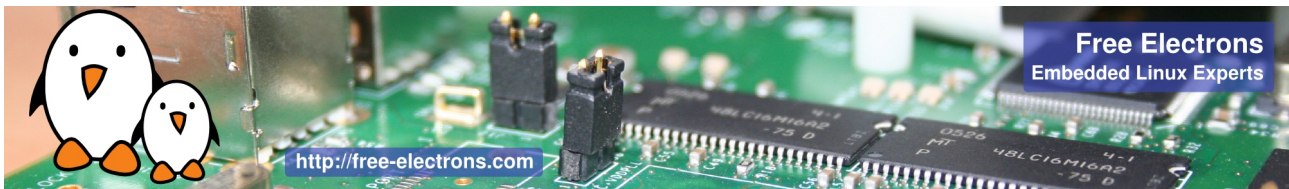
Jour 3 - Matin

Cours - Processus, ordonnancement, sommeil et interruptions
<p>Gestion des processus dans le noyau Linux. L'ordonnanceur du noyau Linux et le sommeil des processus. Gestion des interruptions dans les pilotes de périphérique: enregistrement et développement des gestionnaires d'interruption, utilisation des mécanismes de tasklet.</p>

TP - Sommeil et gestion d'interruption dans un pilote de périphérique
<p><i>En utilisant la carte ARM Calao</i> Ajout de la réception 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.</p>

Jour 3 - Après-midi

Cours - Verrouillage	TP - Verrouillage
<p>Problématique de l'accès concurrent aux 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.</p>	<p><i>Suite du TP précédent</i> Ajout du verrouillage au pilote précédemment développé.</p>



Cours - Techniques de débogage noyau	TP - Investigation de bugs noyau
<p>Débogage avec printk Entrées proc et debugfs Analyse d'un oops noyau Utilisation de kgdb, le débogueur noyau Utilisation des commandes SysRq Débogage en utilisant une sonde JTAG Outils de tracing: ftrace, SystemTap, LTTng</p>	<p><i>En utilisant la carte ARM Calao</i> Étude d'un pilote incorrect. Analyse de l'erreur noyau et recherche du problème dans le code source. Utilisation de l'interface JTAG, au travers d'OpenOCD.</p>

Jour 4 - Matin

Cours - mmap
<p>Les zones de mémoire virtuelle des processus. Amélioration des performances avec mmap, permettant aux applications d'accéder directement au matériel. Implémentation de mmap dans les pilotes de périphérique.</p>

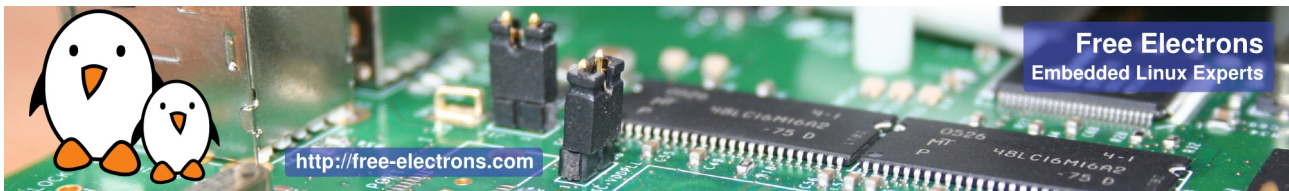
Cours - Architecture du noyau pour les pilotes de périphérique	Cours - Utilisation de DMA
<p>Comprendre comment le noyau est conçu pour supporter les pilotes de périphériques Les « framework » noyau pour quelques types courants de périphériques Le « device model » Connexion entre périphériques et pilotes. Périphériques « platform » Interface en espace utilisateur avec /sys</p>	<p>L'API noyau pour le DMA et son utilisation dans un pilote</p>

Cours - Pilotes PCI	TP - Pilotes PCI
<p><i>L'API noyau pour le PCI et son utilisation dans un pilote</i></p>	<p><i>En utilisant un système x86 virtuel</i> Implémentation de parties d'un pilote PCI</p>

Jour 4 - Après-midi

Cours - Détails de la phase de démarrage du noyau
<p>Description détaillée de la phase de démarrage du noyau, de l'exécution par le chargeur de démarrage jusqu'à l'exécution du premier programme en espace utilisateur. Initcalls: l'enregistrement des routines d'initialisation.</p>

Cours - Pilotes série	TP - Implémentation d'un pilote série
<p>Étude du « framework » noyau pour le développement d'un pilote de type série</p>	<p><i>Implémentation sur la carte ARM.</i> Implémentation de parties du pilote.</p>



Jour 5 - Matin

Cours - Portage du noyau Linux	Cours - Introduction à la gestion de l'énergie
<p>Portage du noyau Linux. Création du code dépendant de la carte. Étude détaillée du code pour une carte ARM.</p>	<p>Support de la modulation dynamique de la fréquence Gestion d'énergie spécifique au CPU et à la carte Gestion de l'énergie dans les pilotes de périphériques. Contrôle depuis l'espace utilisateur. Gestion de l'énergie dans la boucle idle Le framework pour régulateurs de courant et de tension. Étude d'implémentation de la gestion d'énergie dans le noyau Linux.</p>

TP - Gestion de l'énergie
<p><i>En utilisant la station de travail et la carte ARM Calao</i> Utilisation des interfaces standard de gestion de l'énergie : suspend et resume, modulation dynamique de la fréquence. Identification des sources de consommation avec PowerTop.</p>

Jour 5 - Après-midi

Cours - Travail avec la communauté
<p>Comment avoir l'aide de la communauté. Reporter des anomalies. Générer et envoyer des patches. Ressources utiles sur le noyau.</p>

Cours - Gestion des sources noyau avec git	TP - Utilisation de git
<p><i>Très utile pour maintenir vos propres modifications au code de Linux (pilotes, support de nouveau matériel), et pour pouvoir suivre les mises à jour officielles.</i> Récupération d'un arbre git existant. Création de votre branche avec vos changements. Génération de patches. Aperçu des commandes git utiles. Comprendre la méthode de développement utilisée par les développeurs noyau, par l'étude de scénarios typiques.</p>	<p>Créer votre propre branche git à partir de l'arbre officiel. Ajouter des changements provenant d'autres arbres et générer votre propre jeu de patches. Conserver votre branche à jour avec les changements dans l'arbre de référence.</p>