

[Formation] Développer des drivers Linux

OBJECTIFS

- Expliquer les principes du noyau Linux (architecture, compilation)
- Présenter l'API de développement du noyau : API de développement module ; drivers en mode caractère, réseau et bloc ; prise en charge des bus PCI, USB et "platform device"
- Présenter les principes du "device tree"
- Déboguer et profiler le noyau et des drivers

PROGRAMME

Introduction

Présentation de Linux

- Présentation du projet Linux
- Licence
- Architecture Linux (UNIX) (espace utilisateur/noyau)
- Présentation de "systemd"
- Utilisation des sources du noyau Linux : obtention du code source, configuration, compilation native (x86) et croisée (ARM)

L'API du module du noyau

- Rédaction d'un simple module "Hello World" : fonctions "init" et "exit", écrire un Makefile, chargement et déchargement du module à l'aide de "kmod"
- Dépendances des modules
- Utilisation des paramètres de module

Drivers en mode caractère (char drivers)

- Module vs. pilote
- Catégories de pilotes (char, bloc, réseau)
- Appels système d'un pilote char (open, release, read, write, ioctl)
- Enregistrement d'un pilote char
- Utilisation des "classes" du noyau
- Utilisation de l'interface CDEV
- Fonctions de verrouillage ("spinlock" et "mutex")

DATES ET LIEUX

Nous contacter pour les sessions à venir

PUBLIC / PREREQUIS

Développeurs Linux, chefs de projets Linux.

Les participants doivent avoir une base technique assez solide en utilisation de Linux comme plateforme de développement, utilisation du "shell" Linux (bash) et avoir un bon niveau en langage C afin de tirer pleinement profit de cette formation.

COORDINATEURS

Guillaume DUC

Enseignant-chercheur à Télécom Paris, spécialiste de la sécurité des systèmes embarqués. Il est responsable de la filière d'enseignement Systèmes Embarqués et titulaire de la chaire de recherche Connected Cars and CyberSecurity. Il intervient dans de nombreux cours autour des thématiques Linux embarqué (création de distributions embarquées, pilotes de périphériques, etc.).

Pierre FICHEUX

CTO de la division Smile ECS (Embedded & Connected Systems). Auteur de 5 livres sur Linux embarqué (éditions Eyrolles, de 2002 à 2017) et de

Gestion du matériel

- Allocation de mémoire
- Gestion des interruptions : ports d'E/S et accès mappé en mémoire, mappage mémoire (l'appel système "mmap"), DMA
- Écriture d'un pilote PCI générique
- Écriture d'un pilote USB simple (HID)
- Utilisation de l'API de périphérique/pilote de la plateforme pour les systèmes embarqués : présentation du "device tree" - DT (avec la plateforme QEMU/ARM), syntaxe et exemples DT, utilisation de "configs" pour charger une superposition DT

Drivers en mode réseau (net drivers)

- Utilisation du réseau Linux
- Présentation des pilotes réseau
- Structures « net_device » et « net_device_ops »
- Structure "socket buffer" (SKB)
- Écriture et test du pilote réseau factice "faketh"

Drivers en mode bloc (block drivers)

- Driver en mode bloc vs. drivers en mode caractère
- Nouveau framework « blk-mq » (noyau Linux 5.x)
- Écriture d'un simple pilote de bloc (en utilisant un disque virtuel)

Interagir avec l'espace utilisateur

- Limitation de l'approche du "device node"
- Utilisation de "sysfs"
- Utilisation de "udev"
- Écriture de règle udev
- Utilisation de "udevadm"

Débogage des pilotes

- Utilisation de GDB dans l'espace noyau (QEMU/ARM)
- Débogage du noyau statique et d'un driver
- Profilage du noyau et des pilotes avec Ftrace (debugfs) -> trace-cmd

Synthèse et conclusion

livres blancs édités par Smile, consacrés aux logiciels open source pour l'embarqué et l'IoT (Linux embarqué, Android, Linux RT). Il enseigne également le développement des drivers Linux, Linux embarqué et temps réel dans plusieurs écoles d'ingénieurs.

MODALITES PEDAGOGIQUES

Exposés théoriques et nombreux travaux pratiques, avec retours d'expérience d'experts dans l'industrie.

La formation a le gros avantage d'associer un travail pratique à chaque concept présenté.

La formation est basée sur une machine virtuelle (VirtualBox) ce qui garantit l'installation sur des PC Linux ou Windows. Elle peut être réalisée sur place, à distance ou de manière hybride.

La plupart des TP sont réalisés sur architecture x86. Les travaux pratiques liés à l'architecture ARM sont réalisés sur QEMU.