



D3 - Linux drivers

Writing Linux drivers

Objectifs

- Maîtriser les outils kernel de développement et de mise au point
- Découvrir la gestion du multi-core dans le noyau Linux
- Programmer les IO, les interruptions, les timers, le DMA
- Adapter les sources des drivers du marché
- Installer et intégrer les drivers dans un kernel linux
- Gérer les interfaces standard synchrones, asynchrones et ioctl
- Développer les structures des drivers caractères, blocs et réseaux
- Comprendre les spécificités de la version 2.6
- Connaître les évolutions qui ont eu lieu jusqu'au noyau 2.6.23
- Maîtriser les techniques de debugging noyau avec les sondes jtag Lauterbach.

Les exercices se font sur des cartes cibles :

Carte "SnowBall" de ST-Ericsson, basée sur un ARM Cortex/A9 double cœur, avec sonde JTAG Lauterbach.

Carte à base de processeur ARM9 d'Atmel, avec sonde JTAG Lauterbach.

Nous utilisons le dernier noyau disponible sur www.kernel.org

Matériel

- Un PC Linux par binôme
- Une carte cible par binôme
- Une sonde jtag Lauterbach par binôme
- Support de cours
- CDROM avec documentation et exercices corrigés

Pré-requis

- Connaissance de la programmation Linux utilisateur et système (niveau cours D0)

Plan

1er jour

Présentation & Architecture

- Historique
- Licences GPL et open source
- Distributions et versions de Linux
- Architecture du noyau Linux

Programmation Linux Noyau

- Outils de développement
 - outils de compilation
 - sources de documentation
 - makefile
- Développement des modules noyaux
 - différences entre applications utilisateur et noyau
 - écriture d'un module noyau (licence, paramètres, exportation de symboles)
 - chargement et gestion des versions des modules noyau

Exercice : Réalisation d'un système simple de génération sélective dynamique de traces noyau

- Présentation des structures objets du noyau Linux
 - kobject, kset
- Debug noyau
 - traces dans le code noyau (printk)
 - messages Oops!
 - patches permettant le debug noyau (kgdb, ikd)

Exercice : Mise en oeuvre de la sonde jtag Lauterbach

- Allocation mémoire
 - algorithme d'allocation mémoire buddy et slab/slob/slub
 - allocations bloquantes ou "atomiques"
 - limitations de l'allocation dynamique et intérêt de l'allocation statique au boot
- Boot des drivers
 - zones mémoire d'allocations des drivers
 - allocation dynamique et au boot

2ème jour

Multi-tâches Noyau

- Gestion des tâches
 - task struct
 - pile noyau et détermination de la tâche courante (macro "current")
 - quotas d'utilisation des ressources (temps CPU, occupation mémoire, fichiers ouverts...)
- Programmation concurrente
 - Multi-tâche, Multi-coeur et Hyperthreading
 - Quand et comment masquer la préemption
 - spinlocks (simples et "lecture/écriture")
 - seqlocks (nouveaux en 2.6)
 - sémaphores
 - RCU (nouveaux en 2.6)

Exercice : Protection des méthodes d'un objet noyau contre une exécution parallèle

- Timers
 - jiffies
 - timers haute résolution

- Envoi de signaux

Exercice : Mise en évidence des problèmes possibles à l'aide de timers

Introduction aux drivers Linux

- Intégration d'un pilote dans les sources de Linux
 - fichiers Makefile du noyau
 - fichiers Kconfig (description de l'interface de configuration du noyau)
- Concepts des pilotes Linux
 - Pilote en mode utilisateur/pilote en mode noyau
 - numéros majeur/mineur et devnums (nouveau en 2.6). Fichiers spéciaux
 - installation d'un pilote
 - structures associées aux pilotes (struct inode, struct file, file descriptor)
- Transferts de mémoire entre espace noyau et espace utilisateur

Exercice : Ecriture du code d'initialisation d'un pilote

3ème jour

Pilotes caractère

- Ouverture/fermeture
 - Restriction à une seule ouverture/un seul utilisateur
 - différence entre "close" et "release"

Exercice : Ecriture du code d'initialisation d'un driver logiciel

- Transferts de mémoire entre espace noyau et espace utilisateur
 - différences entre adresses virtuelles, logiques, physiques et bus
 - espaces d'adressage des processus. Swap et pagination
 - fonctions de copie entre espaces
 - fonctionnement en "zéro copie" grâce au mapping d'adresses utilisateur dans le noyau
- Lecture et écriture
 - fonctions de base (read/write)
 - lecture/écriture combinées (readv/writev)
 - fonctions asynchrones en mode synchrone (aio_read/aio_write)

Exercice : Ecriture des routines de lecture-écriture d'un pilote logiciel (version sans synchronisation)

- Contrôle des périphériques
 - fonction ioctl
 - choix des codes de commandes

Exercice : Modification des paramètres du périphérique à travers un IOCTL

Entrées sorties synchrones et asynchrones

- Synchronisation des tâches
 - files d'attente
 - mise en attente/réveil d'une tâche
 - attente exclusive
 - attente sur un bit (nouveau en 2.6)
 - évènement de complétion
- Entrées/sorties synchrones

Exercice : Ecriture des fonctions de lecture/écriture synchrones

- Entrées/sorties asynchrones
 - requêtes non bloquantes

- asynchrone multiplexé (select et poll)
- asynchrone notifié (signal SIGIO)
- asynchrone vrai (totalement parallèle)

Exercice : Ajout des fonctions de gestion des E/S asynchrones

4ème jour

Accès au matériel et interruptions

- Accès aux registres des périphériques
 - Espace d'IO et espace de mémoire physique
 - réservation et utilisation des ports d'IO (espace d'IO)
 - réservation et utilisation des plages d'IO en espace physique
 - mapping des registres dans un pilote
 - problèmes d'optimisation (cache, out-of-order, volatile)

Exercice : Ecriture d'un pilote d'entrées/sorties parallèles (GPIO)

- Direct Memory Access (DMA)
 - Slave DMA (ISA)
 - DMA Bus Master (PCI)
 - mapping d'un buffer noyau dans l'espace du périphérique
 - intérêts des mappings permanent ("coherent mapping") et temporaire ("streaming DMA")
- Spécificités des interfaces PCI
 - énumération
 - espace de configuration
 - association dynamique pilote/périphérique (hotplug)
- Interruptions
 - Problèmes spécifique aux Multi-coeurs (SMP)
 - IRQ, ISR, "top half" et "bottom halves" (softirq, tasklet, work_queue)
 - contexte d'exécution des gestionnaires d'interruption (atomique)
 - partage d'interruptions
 - affectation d'interruptions à des processeurs
 - synchronisation entre code noyau, "top half" et "bottom halves"
 - fonctions atomiques

Exercice : Ajout d'un mécanisme d'attente de changement d'état de la GPIO sur interruption

Pilotes bloc

- structures
 - représentation d'un disque (struct gendisk)
 - file des requêtes (struct request queue)
- interface
 - chargement/déchargement (open/release)
 - gestion des media amovibles
 - contrôle (ioctl). Ioctl de description de la géométrie du disque
- routine de stratégie
 - relation entre lecture/écriture et routine de stratégie
 - algorithmes d'ordonnancement des requêtes ("elevators")
 - structures décrivant les requêtes (struct bio et bio_vec)

Exercice : Analyse d'un pilote de Ramdisk

Pilotes réseaux

- structures

- représentation d'une interface réseau (struct net_device)
- paquet réseau (struct sk_buff)
- Cas du "scatter/gather"
- interface
 - réception de paquet
 - envoi de paquet
 - gestion des paquets perdus
 - statistiques de l'interface
- Nouvelles API réseau (NAPI, nouveau en 2.6)
 - "interrupt mitigation" (suppression des IRQ inutiles)
 - "paquet throttling" (désengorgement des couches protocolaires)

Pilotes USB

- La norme USB
 - notion de configuration
 - notion d'interface (rôle d'un périphérique)
 - notion de terminaison (canal de communication)
 - types des terminaisons (contrôle, interruption, bloc, isochrone)
- Drivers USB host
 - requêtes synchrones (directes)

Exercice : Examen d'un pilote USB host

Renseignements pratiques

Duration : 4 days

Cost : 2000 € HT

Prochaines sessions : du 29 May au 1 June 2012
du 19 au 22 June 2012
du 17 au 20 July 2012
du 18 au 21 September 2012
du 23 au 26 October 2012



SARL au capital de 15400€ - SIRET 449 597 103 00026 - RCS Nanterre - NAF 722C - Centre de Formation : 19, rue Pierre Curie - 92400 Courbevoie
Siège social et administration : 21, rue Pierre Curie - 92400 Courbevoie - Tél. 01 41 16 80 10 - Fax. 01 41 16 07 78

Last site update: Tue 22 May 2012 10:50:29 AM CEST

<http://www.ac6-formation.com/>