Écriture de drivers et programmation noyau Linux

Gérer le matériel périphérique

Présentiel ou en classe à distance

Durée : 4 jours (28 h)

Réf. : IXU17

Prix inter : 2.555,00 € HT

Forfait intra : 7.915,00 € HT

Si les ordinateurs sont constitués de périphériques (disques durs, écrans...), ils sont aussi amenés à piloter des périphériques externes de type imprimantes, lecteurs optiques ou lecteurs de cartes à puce. Pour pouvoir les contrôler, les systèmes d'exploitation ont besoin d'interfaces logicielles appelées drivers (ou pilotes en français). Sous Unix et Linux, les drivers sont exécutés dans le noyau du système d'exploitation. Il est donc nécessaire pour le développeur amené à écrire ou à tester des pilotes de périphériques de maîtriser les concepts propres à la programmation noyau. Durant cette formation, les participants seront amenés à écrire des pilotes de différents types pour appréhender les mécanismes parfois complexes (préemptibilité, multiprocesseur, etc.) inhérents au code exécuté en mode noyau.

Les objectifs de la formation

- Comprendre comment programmer pour le noyau Linux
- Connaître les différents types de périphériques et savoir interagir avec eux
- Disposer des compétences nécessaires au développement d'un driver
- Être autonome pour développer des drivers de toute nature

A qui s'adresse cette formation ?

Pour qui

- Toute personne intéressée par le développement systèmes temps réel sur Linux

Prérequis

- Connaissance de Linux (utilisateur) et du langage C

Programme

1 - 1ère partie : Programmer pour le noyau Linux

2 - Noyau Linux et modules

- Modèle
- Versions
- Licence GPL
- Développement du noyau
- Appels-systèmes
- Modules

- Travaux pratiques : observation des appels-système invoqués par des applications et commandes utilisateur et manipulation des modules précompilés

3 - Outils de développement noyau

- Sources
- Compilation du noyau et des modules
- Écriture et compilation de modules
- Cross-compilation
- Messages
- Dépendances
- Travaux pratiques : compilation et installation d'un noyau, écriture de modules simples, intégration dans le noyau, paramètres au boot, cross-compilation sur Raspberry Pi

4 - API du noyau

- Chaînes
- Blocs mémoire
- Fonctions numériques et conversions
- Éléments temporels et actions différées
- Préemptibilité du noyau
- Travaux pratiques : écriture d'un module d'horodatage, chronométrage des phases de boot, mesure de précision d'horloge et mesure de durée d'un appel-système

5 - Environnement du noyau

- Tâches et processus courants
- Espaces d'adressage
- Dialogue avec /proc
- Travaux pratiques : écriture d'un module d'information sur les structures internes des processus, écriture d'un module d'horodatage via /proc et tests sur Raspberry Pi

6 - 2ème partie : Écriture d'un driver

7 - Écriture d'un pilote de périphérique

- Principe
- Numéros majeurs et mineurs
- Classes de périphériques
- Enregistrement du driver
- Fonctions de lecture et écriture
- Travaux pratiques : manipulation des fichiers spéciaux, réservation de numéro majeur, enregistrement de périphérique et écriture d'un driver simple

8 - Appels-système et I/O

- Paramétrage par ioctl
- Synchronisation d'appels-système par mutex
- Accès matériel
- Ports d'entrées-sorties
- GPIO sur carte embarquée
- Travaux pratiques : mise en évidence de la nécessité des mutex, écriture d'un driver d'entrées-sorties sur GPIO du Raspberry Pi

9 - Gestion d'interruption

- Contextes d'exécution
- Installation d'un handler
- Traitement différé (tasklet, workqueue et thread interrupt)
- Travaux pratiques : écriture d'un gestionnaire sur interruption clavier PC et sur GPIO du Raspberry Pi, visualisation des threadeds interrupts

10 - Interactions entre appels-système et interruptions

- Protection des variables globales (spinlock)
- Attentes d'événements (waitqueue)
- Appels-système bloquants
- Travaux pratiques : influence des priorités temps-réel sur les threads d'interruption et mesure de temps de latence des interruptions du Raspberry Pi

11 - 3ème partie : Aspects avancés d'un driver de périphérique

12 - Entrées-sorties avancées

- Multiplexage d'entrée-sorties (select et poll)
- Principes des transferts de données par DMA
- Travaux pratiques : création d'un périphérique "file de messages" virtuel implémentant plusieurs appels-système, implémentation de select sur des entrées GPIO

13 - Gestion de la mémoire

- Allocation et libération de mémoire (kmalloc, vmalloc, get_free_pages, kmem_cache)
- Projections (mmap)
- Travaux pratiques : expériences sur la projection mémoire en espace utilisateur et allocations mémoire maximales

14 - Périphériques blocs et VFS

- Principes
- Enregistrement
- Disque générique
- File de requêtes
- Partitionnement
- Sous-système Block, i/o scheduler
- Virtual File System
- Travaux pratiques : écriture d'un driver de disque virtuel, partitionnement, formatage et montage de disque virtuel, observation des effets des caches-disques du VFS

15 - Périphériques PCI Express

- Principe
- Détection et enregistrement de driver
- Base Address Registers
- Interruptions classiques et MSI
- Travaux pratiques : étude d'un driver PCIe de pilotage d'une carte à FPGA

16 - 4ème partie : Autres types de périphériques

17 - Périphériques réseau

- Interfaces bas-niveau et protocoles réseau
- Périphérique net_device
- Enregistrement
- Activation
- Émission et réception de paquets
- Travaux pratiques : écriture progressive d'un driver pour périphérique virtuel permettant l'utilisation du protocole IPv4

18 - Communications réseau

- Statistiques d'utilisation d'interface
- Principes de la pile IP
- Communications entre protocoles et interface bas-niveau
- Travaux pratiques : examen du trajet des données au sein de la pile IPv4 lors de réception et d'émission de données avec le protocole TCP/IP

19 - Utilisation du bus USB

- Organisation du sous-système USB de Linux
- Implémentation d'un driver Interrupt
- Type de Endpoints
- Communication avec les URB
- Travaux pratiques : écriture d'un driver pour carte d'entrée-sortie Velleman K8055

20 - Aspects avancés

- Écritures successives rapides
- Déconnexions intempestives
- Accès concurrents
- Exemples de drivers Bulk, Control et Isochrones

21 - Conclusion : discussions libres sur l'ensemble des thèmes abordés

22 - Travaux pratiques : expérimentations libres suivant les demandes des participants

Evaluation

- Cette formation fait l'objet d'une évaluation formative.

Les points forts de la formation

- Les apports théoriques sont très largement complétés par des phases de mise en pratique qui amènent les participants à réaliser de nombreuses manipulations.
- Les exercices et démonstrations ont lieu, suivant les sujets, sur plate-forme PC, sur carte ARM Raspberry Pi et sur carte ARM BeagleBone Black.

Dates et villes 2024 - Référence IXU17

A distance

du 20 juin au 23 juin	du 26 août au 29 août

Paris

du 20 juin au 23 juin	du 26 août au 29 août