Formation Ecriture de drivers et programmation noyau Linux

Description de la formation linux drivers :

Le support pour les périphériques est assuré sous Linux par des pilotes (drivers) dont le code se déroule 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. Cette formation Linux Drivers propose une approche originale, s'appuyant sur l'écriture progressive de pilotes de différents types, pour appréhender les mécanismes parfois complexes (préemptibilité, multiprocesseur, support d'architectures différentes, etc.) inhérents au code exécuté en mode noyau. Cette formation Linux drivers vous permettra de maîtriser le développement de pilotes de périphériques (drivers) robustes et adaptés aux différentes déclinaisons de Linux. Outre les périphériques classiques (caractère, bloc, réseau), son étudié les sous-systèmes du noyau tels que les systèmes de fichiers ou l'ensemble USB.

Objectifs de la formation

Concrètement cette formation vous apprendra à:

- Maîtriser le développement de pilotes de périphériques (drivers) robustes et adaptés
- Connaitre les outils de développement noyau
- Maitriser les aspects avancés d'un driver de périphérique
- Savoir gérer la mémoire
- Connaitre les Périphériques réseau et USB

À qui s'adresse cette formation ?

Public :

Prérequis :

Connaissance de Linux (utilisateur) et bonne familiarité avec le langage C.

Contenu du cours

Programmer pour le noyau Linux
Noyau Linux et modules modèle, version, licence GPL, développement du kernel, appels-système, modules. Travaux pratiques Observation des appels-système invoqués par des applications et commandes utilisateur. Manipulation des modules précompilés. Outils de développement noyau sources, compilation noyau et modules, écriture et compilation de modules, cross-compilation, messages, dépendances. Travaux pratiques Compilation et installation d'un noyau. Écriture de module simple. Intégration dans le noyau. Paramètres de boot. Cross-compilation sur Raspberry Pi. API du noyau chaines, 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. Mesure de durée d'appel-système. Environnement du noyau tâches et processus curent, espaces d'adressage, dialogue avec /proc. Travaux pratiques Ecriture d'un module d'information sur les structures internes des processus. Écriture d'un module d'horodatage via /proc. Test sur Raspberry Pi.

Écriture d'un driver Linux
Écriture d'un pilote de périphérique principes, 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. Écriture d'un driver simple. Appels-systèmes et I/O Paramétrage par ioctl. Synchronisation d'appels-système par mutex. Accès matériel. Ports d'entrée-sorties. GPIO sur carte embarqué Travaux pratiques Mise en évidence de la nécessité des mutex. Écriture d'un driver d'entrée-sorties sur GPIO du Raspberry Pi. Gestion d'interruptions contextes d'exécution, installation d'un handler, traitement différé (tasklet, workqueue et threaded interrupts). Travaux pratiques Écriture d'un gestionnaire d'interruption sur interruption clavier PC et sur GPIO du Raspberry Pi. Visualisation des threads interrupts. 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. Mesure de temps de latence des interruptions du Raspberry Pi.

Aspects avancés d'un driver de périphérique

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. 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. Allocations mémoire maximales. Périphériques blocs et VFSprincipes, enregistrement, disque générique, file de requêtes, partitionnement, sous-système Block, i/o scheduler, Virtual File System. Travaux pratiques Ecriture d'un driver de disque virtuel. Partitionnement, formatage et montage de disque virtuel. Observation des effets des cache-disques du VFS.Périphériques PCI Express principe, détection et enregistrement de driver, Base Address Registres, interruptions classiques et MSI. Travaux pratiques Etude d'un driver PCIe de pilotage d'une carte à FPGA.

Périphériques réseau et USB

Périphériques réseau interfaces bas-niveau et protocoles réseau, périphérique net_device, enregistrement, activation, émission et réception de paquets. Travaux pratiques Ecriture progressive d'un driver pour périphérique virtuel permettant l'utilisation du protocole IPv4.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. Périphériques USB Organisation du sous-système USB de Linux, implémentation d'un driver Interrupt, type d'end points, Communication avec les URB.Travaux pratiques Ecriture d'un driver pour carte d'entrée-sortie Velleman K8055.Aspects avancés écritures successives rapides, déconnexions intempestives, accès concurrents, Exemples de drivers Bulk, Control et Isochrones.