

LINUX KERNEL ET DRIVERS (DÉVELOPPEMENT DE MODULES)

4 jours

Réf. : 5136

OBJECTIFS

Cette formation «Linux Kernel et Drivers» permet aux développeurs en informatique de maîtriser les concepts et les outils de développement croisé d'applications et de modules (drivers) GNU/Linux sur système industriel et cible embarquée.

Cette formation aborde les concepts du développement de modules en kernel mode et de drivers sous Linux par l'étude :

- des chaînes de développement GNU/Linux en mode kernel,
- des spécificités du système Linux et de son noyau,
- du développement de modules Kernel et de pilotes de périphériques sous OS Linux.

PRE-REQUIS

La formation Linux Kernel et Drivers est adaptée aux développeurs en informatique et techniciens débutants dans le domaine de l'informatique Open Source, confrontés aux problèmes de portage d'applications de contrôles industriels et de l'embarqué sous OS GNU/Linux. Une maîtrise du langage C est nécessaire.

PROGRAMME

PREMIÈRE JOURNÉE

Linux : Chaîne de développement croisé

- Méthodes, outils et chaîne de développement croisé, binutils, glibc, etc...
- Makefile, Compilateur et débogueur GNU
- Mise au point par port série et TCP/IP

Linux Kernel : Présentation

- Vue d'ensemble du système et rôle du noyau
- Historique, numérotation des versions
- Architectures matérielles supportées
- Support processeurs et File system
- Kernel 2.4, 2.6 et systèmes embarqués

Linux Kernel : Configuration et compilation du noyau

- Structure des sources et modules du noyau
- Optimisation - Patches low-latency, préemptifs
- Fichiers utilisés par les outils de configuration
- Application de patches et utilisation de BSP
- Configuration et compilation du noyau

Travaux pratiques

- Installation et configuration d'une chaîne de développement croisé pour cible ARM 9
- Compilation de noyau optimisé pour le cross développement sur cible ARM 9

DEUXIÈME JOURNÉE

Linux Module : Développement de pilotes

- Pilotes de périphériques sous Linux
- Contraintes de programmation et API Kernel Mode
- Chargement, déchargement de modules
- Un module simple
- Accès aux registres d'E/S et à la mémoire
- Gestion de la mémoire en kernel mode
- Pilotes de type caractère

DEUXIÈME JOURNÉE (suite)

Linux Module : Installation et paramétrage de pilotes

- Paramètres de chargement de modules
- Systèmes de fichiers sysfs et entrées procfs
- Dépendances entre modules
- Intégration de module propriétaire dans la chaîne de configuration et de compilation des sources officielles

Travaux pratiques

- Compilation d'un noyau instrumenté pour le débogage de module
- Ajout de sources d'un module à l'arborescence du noyau
- Création de patches Kernel

TROISIÈME JOURNÉE

Linux Module : Services et configurations pour la mise au point en Kernel Mode

- Console série
- LTT : Linux Trace Toolkit
- Débogueur Kernel Mode : KGDB

Linux Driver : Développement avancé de pilotes sous Linux

- Mise en sommeil, interruptions, mmap, DMA
- Fichiers de périphériques dynamiques avec udevs

Linux Driver : Hardware detection et classe pilotes de périphériques industriels

- Extensions ISA/PC104, PCMCIA, bus de terrain, I²C, CAN,
- Cas particulier du PCI...

Linux Driver : Périphérique de type caractères particuliers

- Driver de port série et support console série...

TROISIÈME JOURNÉE (suite)

Travaux pratiques

- Création et installation d'un pilote de périphérique PC104 de type caractère sur cible GEODE x86
- Traitement d'interruption matérielle en Kernel mode sur port E/S
- Mise en œuvre de LTT
- Mise en œuvre de KGDB

QUATRIÈME JOURNÉE

Linux Driver : Développement avancé de pilotes (suite)

- Architecture des pilotes de périphériques de type bloc
- Architecture des pilotes de périphériques de type réseau
- Frame Buffer vidéo

Linux Driver : Etude de périphériques de type USB

- Standard USB et support Linux Host et Device
- USB Core - Architecture des pilotes USB sous Linux
- Descripteur et classe USB device sous Linux : HID, CDC, Masse storage ...

Travaux pratiques

- Création et installation d'un pilote de périphérique de type bloc
- Accès Direct Frame Buffer
- Mise en œuvre de drivers USB sous Linux avec analyseur Ellisys 2.0

© CenoSYS 2010-01