

LINUX EMBARQUÉ, DRIVERS ET TEMPS RÉEL

5 jours

Réf. : 5158

OBJECTIFS

Ce stage permet aux développeurs, confrontés aux problèmes de portage d'un noyau Linux et des applications temps réel sur plate-forme Linux embarqué, de pouvoir concevoir une distribution optimisée et des drivers linux sur mesure.

Ce stage de formation Linux embarqué aborde les concepts du portage d'un OS Linux sur cible embarquée par l'étude :

- des caractéristiques et architectures des systèmes Linux embarqué,
- de la mise en œuvre d'une chaîne de développement croisé,
- de la compilation d'un noyau et l'installation sur ROM/FLASH NAND et NOR,
- de la préparation d'un BSP et d'un boot-loader Linux pour l'embarqué,
- des packages et de la configuration de l'installation sur différents types d'architectures matérielles x86 et ARM9,
- de la mise au point et la validation de modules et de pilotes de périphériques Linux,
- des API et des extensions temps réel sous Linux embarqué.

PRE-REQUIS

Cette formation est particulièrement adaptée aux techniciens et ingénieurs, confrontés aux problèmes de portage de solutions Linux sur systèmes embarqués, drivers et temps réel.

PROGRAMME

PREMIÈRE JOURNÉE

Distributions Linux Embarqué

- Projets existants: MontaVista, TimeSys, WindRiver, TimeSys, uClinux, eldk,...

Présentation Noyau Linux pour l'Embarqué

- Vue d'ensemble du système et rôle du noyau
- Architectures matérielles supportées - processeurs et File system
- Structure des sources du noyau, fichiers utilisés par les outils de configuration
- Configuration, optimisation et compilation d'un noyau,
- BSP Linux embarqué : application de patch au Kernel Linux officiel

Méthodes et outils de validation

- Compilation croisée - Méthodes, outils GNU Linux,
- Débogueur distant via port série ou par réseau : gdbserver
- IDE Eclipse

Travaux pratiques

- Préparation d'une chaîne de développement croisé,
- Configuration et compilation d'un noyau 2.6 « patché » ARM 9

DEUXIÈME JOURNÉE

Services et configurations Linux embarqué

- Personnalisation du système : Scripts de démarrage et fichiers de configuration Linux embarqué
- Installation des services réseaux : Console série Inetd, Rsh, telnet, Nfs

DEUXIÈME JOURNÉE (suite)

Environnement utilisateur et Processus sous Linux embarqué

- Choix de bibliothèques : LibC : glibc, uClibc, NewLibC
- Gestion de la mémoire virtuelle : page stack, swap et overcommit memory
- Démons Unix et services réseau TCP/IP Client Serveur
- Environnements graphiques X: nanoX, XFree, Qt Embedded,

TROISIÈME JOURNÉE

Mise au point du Boot Loader et du kernel Linux embarqué

- Préparation du boot loader, Setup de l'architecture et commandes U-BOOT
- BSP Linux embarqué : mapping E/S physique mémoire RAM/FLASH

Travaux pratiques

- Mise en œuvre d'une solution de débogage boot loader et noyau pour ARM 9 avec sonde JTAG Abatron BDI

Portage d'une image système Linux embarqué

- Technologies MTD : Flash Chip NOR, NAND, Disk Flash: CompactFlash,...
- Systèmes de fichiers, génération d'image file system Linux CRAMFS, JFFS2 et iniramfs
- Shell et d'outils d'administration (BusyBox ...)

Travaux pratiques

- Installation d'un système bootable via réseau et montage nfs avec U-BOOT sur cible ARM 926

Introduction au développement Kernel Mode Linux

- Introduction à la programmation en mode noyau, architecture d'un module linux simple
- Intégration de codes sources personnels au kernel Linux
- Gestion de paramètres de modules, communication avec les systèmes de fichiers sysfs et procfs

QUATRIÈME JOURNÉE

Introduction au développement de pilotes Linux

- Programmation de pilotes de périphériques : rôle de File Operations
- API noyau Linux et gestion mémoire en Kernel Mode
- Driver bloquant, gestion d'interruption sous Linux, Signaux et Timer en kernel mode
- Mise au point des drivers linux : KGDB, Kprobe, Linux Trace Tools Kit,

Travaux pratiques

- Construction et compilation de pilotes de périphériques simples et bloquants

CINQUIÈME JOURNÉE

Introduction aux applications embarquées temps réel en mode utilisateur

- API POSIX temps réel souple: ordonnancement des processus et signaux UNIX sous Linux
- Programmation multi-thread et extension API POSIX Thread, ...
- Timers Linux et choix du Tick system pour l'embarqué

Extensions temps réel embarquées en mode noyau sous Linux

- Temps réel dur sous Linux : patches low-latency, temps réel RTAI/RTLinux
- Temps réel strict : Tâche et ordonnanceur temps réel RTAI
- Communication et synchronisation de tâches temps réel RTAI
- Timer temps réel RTAI et gestion d'IT,
- FIFO temps réel RTAI entre module RTAI et processus Linux
- LXRT : tâches temps réel RTAI et Thread Linux

Travaux pratiques

- Mise en œuvre d'un Linux RTAI sur cible PC104

© CenoSYS 2010-01