oD10 - Programmation utilisateur et Kernel de Linux

Développement des systèmes Linux embarqués

Objectifs

- Découvrir Linux et ses outils de développement
- Connecter un système Linux embarqué dans un réseau
- Examiner la séquence de démarrage de Linux
- Amorcer un Kernel Linux distant
- Programmer et déboguer des applications Linux
- Maîtriser les outils de développement et de débogage du Kernel
- Programmer les entrées/sorties, les interruptions
- Découvrir les techniques de débogage du Kernel

Les travaux pratiques sont effectués sur une carte ARM QEMU
Nous utilisons une version récente du Kernel

Pré-requis

- Connaissance de base de l'utilisateur de Linux
- Bonnes connaissances en programmation C (voir notre cours L2)

Environnement du cours

- Cours théorique
  - Support de cours au format PDF (en anglais).
  - Cours dispensé via le système de visioconférence Teams.
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique.
- Activités pratiques
  - Les activités pratiques représentent de 40% à 50% de la durée du cours.
  - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
  - Exemples de code, exercices et solutions.
  - Un PC Linux en ligne par stagiaire pour les activités pratiques.
  - Le formateur a accès aux PC en ligne des stagiaires pour l'assistance technique et pédagogique.
  - Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante.
- Une machine virtuelle préconfigurée téléchargeable pour refaire les activités pratiques après le cours
- Au début de chaque session une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus

Durée

- Totale : 24 heures
- 4 sessions, 6 heures chacune (sans compter les pauses)
- Certains travaux pratiques doivent être terminés après la session

Modalités d'évaluation

- Les prérequis indiqués ci-dessus sont évalués avant la formation par l'encadrement technique du stagiaire dans son entreprise, ou par le stagiaire lui-même dans le cas exceptionnel d'un stagiaire individuel.
- Les progrès des stagiaires sont évalués de deux façons différentes, suivant le cours:
  - Pour les cours se prêtant à des exercices pratiques, les résultats des exercices sont vérifiés par le formateur, qui aide si nécessaire les stagiaires à les réaliser en apportant des précisions supplémentaires.
  - Des quizz sont proposés en fin des sections ne comportant pas d'exercices pratiques pour vérifier que les stagiaires ont assimilé les points présentés
- En fin de formation, chaque stagiaire reçoit une attestation et un certificat attestant qu'il a suivi le cours avec succès.
  - En cas de problème dû à un manque de prérequis de la part du stagiaire, constaté lors de la formation, une formation différente ou complémentaire lui est proposée, en général pour conforter ses prérequis, en accord avec son responsable en entreprise le cas échéant.

Plan

Première session

Linux overview

- Linux
  - History
  - Version management
- Linux architecture and modularity
- Linux system components
- The various licenses used by Linux (GPL, LGPL, etc)

Linux application development

- Structure of Linux applications
  - The ELF file format
- Linux development tools
  - Compiling
  - Documentation
  - Makefiles
  - Integrated Development Environments
- Creating Linux libraries
  - Static libraries
  - Dynamic libraries

Exercise : *Writing a simple, static and dynamic, library*

Linux application debugging

- Software Debug tools
  - Gdb
  - Memory management debug using dmalloc and efence
  - Runtime checks using valgrind

Exercise : *Debug an application and its libraries using gdbserver*
Exercise : *Checking memory management using dmalloc and valgrind*

User-Level Input-Output

- Standard input-output
  - disk files
  - devices
- Network programming

  - sockets
  - UDP and TCP protocols
- Asynchronous input-output
  - Non-blocking I/O
  - Multiplexed (the select and poll APIs)
  - Notified I/O
  - Chained I/O (the aio POSIX API)

Exercise : *Understanding Non-blocking Requests*

Deuxième session

Signal handling

- Signal handling
  - Signal types
  - Handling a signal
  - Functions usable in a signal handler
  - Signal masking and synchronous handling

Exercise : *Manage timeouts using signals and timers*

Multitask programming

- POSIX Threads
  - Definition of a thread
  - Thread programming
- Processes
  - The process concept
  - Processes and security
  - Process states
  - Process life-cycle: the "fork" and "'exec" calls
- Inter-Process Communications
  - File Mapping
  - Shared Memory

Exercise : *Fork a process*
Exercise : *Understand mmap system call*

Linux kernel programming

- Development in the Linux kernel
  - Kernel compilation
- Modules compilation
- Memory allocation in the kernel
- Helper libraries
  - Kref and container_of
  - Linked list
  - Fifos

Exercise : *Write the typical "Hello World" kernel module*
Exercise : *Understand kernel parameters and their access through the /sys filesystem*

Troisième session

Linux kernel Debugging

- Oops
- The /proc filesystem
- The DebugFS filesystem
- Traces
- Miscellaneous Kernel Debug Features

- Debugging with KGDB

Exercise : *Debugging a module using KGDB*

Kernel multi-tasking

- Kernel task handling
- Concurrent kernel programming
- Miscellaneous Locks
- Creating kernel threads

Introduction to Linux drivers

- Kernel Architecture
- Device Access
- Functional Driver registration

Exercise : *Understand the driver structure and registration*

Driver IO functions

- Kernel structures used by drivers
- Opening and closing devices
- Data transfers
- Controlling the device
- Mapping device memory

Exercise : *Understand how to transfer data to or from user space*

Exercise : *Understand the basic read and write calls*

Quatrième session

Synchronous and asynchronous requests

- Task synchronization
- Synchronous request
- Asynchronous requests
  - Non-blocking requests
  - Multiplexed I/O
  - Notified I/O

Exercise : *Understand the ioctl system call handling*

Input-Output and Interrupts

- Memory-mapped registers
- Interrupts
- Legacy features

The Linux Device Model

- Linux Device Model Architecture
  - Overview
  - Classes
  - Busses
- The platform bus
  - Platform devices
  - Platform drivers
- Classes
  - The misc class
  - Custom classes
- Writing udev rules

Exercise : *Learning how to implement a platform driver (callbacks, registration, resources)*

Renseignements pratiques

Durée : 24 heures
Prix : 2270 € HT