D0 - Programmation en mode utilisateur Linux

Programmation d'applications Linux embarquées pour Linux

Objectifs

- Découvrir Linux et ses outils de développement
- Connecter un système Linux embarqué dans un réseau
- Passer en revue la séquence de démarrage de Linux
- Monter un système de fichiers distant
- Amorcer un noyau Linux à distance
- Programmer et déboguer des applications Linux
  - Programmation réseau
  - Entrée-sortie synchrone et asynchrone
  - Programmation multithread
  - Communications inter-processus

Les travaux pratiques sont menés sur des cartes cibles, qui peuvent être :
Cartes "STM32MP15-DISCO" à base de double Cortex/A7 de STMicroelectronics
Cartes "SabreLite" à base de Quad Cortex/A9 de NXP
Cartes "imx8q-evk" à base de Quad Cortex/A53 de NXP

Qui devrait suivre ce cours ?

- Les ingénieurs qui doivent créer des applications Linux embarquées

Pré-requis

- Connaissance de base des utilisateurs de Linux
- Bonnes connaissances en programmation C

Environnement du cours

- Cours théorique
  - Support de cours imprimé et au format PDF (en anglais)
  - 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.
  - Un PC pour deux stagiaires pour les activités pratiques
  - Une plateforme cible pour deux stagiaires (sauf en cas d'utilisation de qemu)
  - Exemples de code, exercices et solutions
  - Le formateur assiste les stagiaires pendant les exercices
- Une machine virtuelle préconfigurée téléchargeable pour les activités pratiques après le cours
- Au début de chaque demi-journée 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

Course Outline

Premier jour

Linux overview

- Linux
- The various licenses used by Linux (GPL, LGPL, etc.)
- Linux distributions

Linux for the user

- The Linux filesystem
- The Linux shell and scripts
- The vi editor
- Basic administration of a Linux system

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

Deuxième jour

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

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 : Programming a client-server application
Exercise : Handle several parallel connections using asynchronous I/O

Tracing system calls

- Trace system calls tools
    - Strace
    - Ltrace

Exercise : Understanding Strace

Troisième jour

Time and signal handling

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

Exercise : Manage timeouts using signals and timers

Multitask programming

- Processes
    - The process concept
    - Processes and security
    - Process states
    - Process life-cycle : the "fork" and "'exec" calls
- POSIX Threads
    - User and kernel threads
    - Thread programming
    - Mutexes and condition variables
    - Barriers
    - Thread-specific data

Exercise : Managing several clients in parallel using fork
Exercise : Create a remote server using fork and exec
Exercise : Managing several clients in parallel using threads
Exercise : Manage thread-static data in a library

Quatrième jour

Memory management and Scheduling

- Linux Memory Management
    - Virtual and physical memory
    - Pagination and protection
    - Swapping
    - Memory allocation
    - Caches
- Scheduling in the Linux kernel
    - Context switches
    - The Completely Fair Scheduler
    - Scheduling groups
    - The real-time scheduler
    - Scheduling and SMP (Symmetrical Multi Processors)

Inter Process Communication

- Inter Process Communication
  - File mapping of files and devices
  - Shared memory
  - Message queues
  - Pipes
- Task synchronisation
  - Semaphore
  - Mutex
  - Signals
- The System V IPC (optional, described in appendix)

Exercise : Handle communications between processes in a multi-process client-server system
Exercise : Setup timeouts to close dead connections on a server