D0 - Linux user mode programming

Programming Embedded Linux Applications for Linux

Objectives

- Discover Linux and its development tools
- Connect an embedded Linux system in a network
- Review the Linux boot sequence
- Mount a remote file system
- Boot a remote Linux kernel
- Program and Debug Linux applications
  - Network programming
  - Synchronous and asynchronous input-output
  - Multi-thread programming
  - Inter-process communications

Labs are conducted on target boards, that can be:
Dual Cortex/A7-based "STM32MP15-DISCO" boards from STMicroelectronics.
Quad Cortex/A9-based "SabreLite" boards from NXP.
Quad Cortex/A53-based "imx8q-evk" boards from NXP.

Who should attend this course?

- Engineers that must create embedded Linux applications

Prerequisite

- Basic Linux user knowledge
- Good C programming skills

Course Environment

- Theoretical course
  - PDF course material (in English) supplemented by a printed version.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
    - One PC for two trainees when there are more than 6 trainees.
  - For onsite trainings:
    - An installation and test manual is provided to allow preinstallation of the needed software.
    - The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

First Day

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*

Second Day

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*

Third Day

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*

Fourth Day

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*

Renseignements pratiques

Inquiry : 4 days