



## oD0 - Linux User Mode Programming

*Programming Embedded Applications under 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 QEMU ARM-based board

We use a recent version of Kernel

### 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)
  - Course dispensed using the Teams video-conferencing system
  - The trainer to answer trainees' questions during the training and provide technical and pedagogical assistance through the Teams video-conferencing system
- Practical activities
  - Practical activities represent from 40% to 50% of course duration
  - One Online Linux PC per trainee for the practical activities
  - The trainer has access to trainees' Online PCs for technical and pedagogical assistance
- Downloadable preconfigured virtual machine for post-course practical activities

### Duration

- Total: 24 hours
- 4 sessions, 6 hours each (excluding break time)
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

### Target Audience

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

# Course Outline

## First Session

### 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

### 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

## Second Session

### 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 Session

### 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 Session

### 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