

Hands-on course , 4
day(s)
Ref : LIS

Pre-requisites

Good knowledge of UNIX/
Linux system and C
programming language.

Next sessions

Unix/Linux system developer

OBJECTIVES

This training session will teach you how to make use of the wide and precise UNIX system interface to develop performant applications. You will learn to manage processes and threads, file systems, and memory allocation. At the end of this session, you will be familiar with network communication, signals, Posix and System V IPC.

1) Development method and tools

2) Processes

3) Posix Threads

4) Files and filesystems

5) Inter Process Communication

6) Network

7) Memory management and Time

8) Advanced linker control

Workshop

Progressive practicals and case studies will enable you to fully understand the presentation of the UNIX/Linux system programming interface.

1) Development method and tools

- The linux distributions, open source software, licenses.
- Compilers and associated tools, profilers and debuggers.

Workshop

Use of GDB, cscope and profiling with a simple application.

2) Processes

- The UNIX processes, scheduling processes, priorities and processor affinity.
- Live and death of a process. Fork, exec, exit and wait.
- Security issues. Root and standard user, user ids.
- Namespaces and application mobility.

Workshop

Creation of a simple multi-process application. Tests of some security issues like process running wild and handling with `setrlimit(2)`.

3) Posix Threads

- Programing with threads. Overview of Posix 1c threads.
- Thread creation and termination. Thread scheduling.
- Synchronizing Threads. Mutex and data protection, priority inversion.
- Condition variable and flow control. Using signals and threads.

Workshop

A simple multi-thread application using mutexes and condition variables.

4) Files and filesystems

- File handling.
- Filesystems. Accessing metadata. Accessing directories.
- I/O Multiplexing. Using poll and select.
- Signals and events with multiplexed I/O

Workshop

A small program using `fcntl(2)` for file locking and accessing a directory.

5) Inter Process Communication

- Message queues.
- Shared memory.
- Semaphores. Handling multiple semaphore sets.
- Pipes. Standard I/O redirections.
- Signals. UNIX signals implementation.

Workshop

Implementation of a client/server use case with the help of various technologies: message queues, shared memory and semaphores, pipes and signals.

6) Network

- Socket Interface.
- Address and protocol management. TCP/IP interface.
- Network daemons.
- mplementation of network servers and super servers.

Workshop

Implementation of our client/server use case with the network interface.

7) Memory management and Time

- Virtual memory.
- Memory allocation.
- Advanced use
- Date and time. Timers and timeout.
- Latencies and determinism.

Workshop

Test of various allocations schemes using malloc(3), brk(2) or mmap(2). Implementing good practice for real-time applications. Using the UNIX time interface to measure scheduling latency.

8) Advanced linker control

- Security, real-time and multithreading.
- Shared libraries.

Workshop

A multi-thread real-time application with a time share thread. Implementation of a memory allocation tracer with dlopen.