# Linux/UNIX Threads and IPC Programming

Course code: M7D-TIPC01

This course provides a grounding in multithreaded programming using POSIX threads as well as a deep understanding of various IPC (inter-process communication) techniques that can be used to build cooperating multiprocess applications. Detailed presentations coupled with many carefully designed practical exercises provide participants with the knowledge needed to write complex system, network, and multithreaded applications.
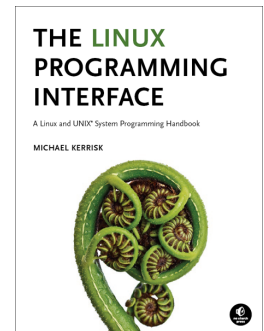
Audience and prerequisites

The audience for this course includes programmers developing and porting system-level and network applications for Linux and UNIX systems, embedded application developers, security engineers, site reliability engineers, and DevOps engineers. To get the most out of the course, participants should have:

- Good reading knowledge of the C programming language
- Solid programming experience in a language suitable for completing the course exercises (e.g., C, C++, D, Go, Rust, or Python)
- Knowledge of basic UNIX/Linux shell commands

Some system programming background is assumed. Where necessary, such background can be gained from either the *Linux System Programming Fundamentals* (M7D-SPINTRO01) or the *Linux System Programming Essentials* (M7D-SPESSO01) course.

Course duration and format

Three days, with up to 40% devoted to practical sessions.

Course materials

- Course books (written by the trainer) that include all slides and exercises presented in the course
- An electronic copy of the trainer's book, *The Linux Programming Interface*
- Numerous example programs written by the course trainer

Course inquiries and bookings

For inquiries about courses and consulting, you can contact us in the following ways:

- Email: training@man7.org
- Phone: +49 (89) 2155 2990 (German landline)

Prices, dates, and further details

For course prices, upcoming course dates, and further information about the course, please visit the course web page, `http://man7.org/training/lusp/`.

About the trainer

Michael Kerrisk has a unique set of qualifications and experience that ensure that course participants receive training of a very high standard:

- He has been programming on UNIX systems since 1987 and began teaching UNIX system programming courses in 1989.
- He is the author of *The Linux Programming Interface*, a 1550-page book acclaimed as the definitive work on Linux system programming.

- He has been actively involved in Linux development, working with kernel developers on testing, review, and design of new Linux kernel–user-space APIs.
- Since 2000, he has been the involved in the Linux *man-pages* project, which provides the manual pages documenting Linux system calls and C library APIs, and was the project maintainer from 2004 to 2021.

Linux/UNIX Threads and IPC Programming: course contents in detail

Topics marked with an asterisk (*) are optional, and will be covered as time permits

1. **Threads: Introduction**
   - Overview of threads
   - Pthreads API basics
   - Thread creation and termination
   - Thread IDs
   - Joining and detaching threads
   - Thread attributes
   - Signals and threads
   - Threads and process control

2. **Threads: Synchronization**
   - Shared resources and critical sections
   - Mutexes
   - Locking and unlocking a mutex
   - Condition variables
   - Signaling and waiting on condition variables
   - Further details on signaling condition variables
   - Dynamically initialized synchronization primitives
   - Other synchronization primitives

3. **IPC: Introduction and Overview (*)**
   - Categorizing IPC
   - Choosing an IPC mechanism

4. **Pipes and FIFOs**
   - Creating and using pipes
   - FIFOs
   - Connecting filters with pipes

5. **Sockets: Concepts and UNIX Domain**
   - Socket types and domains
   - Creating and binding a socket
   - System calls: stream sockets
   - UNIX domain stream sockets
   - System calls: datagram sockets
   - UNIX domain datagram sockets
   - Further details of UNIX domain sockets

6. **Sockets: Internet Domain**
   - Internet domain sockets
   - Data-representation issues
   - Loopback and wildcard addresses
   - Host addresses and port numbers
   - Host and service conversion
   - Internet domain sockets example
   - Additional sockets system calls

7. **Alternative I/O Models**
   - Nonblocking I/O
   - Signal-driven I/O
   - I/O multiplexing: *poll()*
   - Event-loop programming

8. **Alternative I/O Models: *epoll***
   - Problems with *poll()* and *select()*
   - The *epoll* API
   - *epoll* events
   - Performance considerations
   - Edge-triggered notification
   - *epoll* API quirks

9. **POSIX Semaphores**
   - Named semaphores
   - Semaphore operations
   - Unnamed semaphores

10. **POSIX Shared Memory**
    - Creating and opening shared memory objects
    - Using shared memory objects
    - Synchronizing access to shared memory