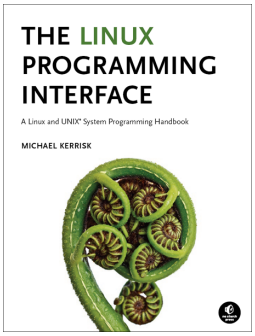
man7.org
Training and Consulting

Linux/UNIX IPC Programming

Course code: M7D-IPC02

This three-day course provides a thorough introduction to the interprocess (IPC) techniques that Linux and UNIX systems provide for use by user-space programs. Using these features allows the creation of complex multiprocess applications that coordinate their actions and exchange information with each other. Detailed presentations coupled with many carefully designed practical exercises provide participants with the knowledge to write such applications. The course dives into some specifics of the Linux system, but makes careful and frequent reference to the POSIX standard, so that it is also valuable to developers working on other UNIX systems.



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++, Go, Rust, or Python)
- Knowledge of basic UNIX/Linux shell commands
- Working knowledge of the fundamental system programming topics covered in the *Linux System Programming Fundamentals* course (M7D-SPINTRO01). In particular, participants should have a good understanding of concepts such as: file I/O using system calls, process lifecycle (*fork()*, *exec()*, *wait()*), and signals.

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*
- A source code tarball containing more than 35,000 lines of example code written by the 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 and further details

For course prices and further information, please visit the course web page, *http://man7.org/training/ipc_prog/*.

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 widely acclaimed as the definitive work on Linux system programming.
- He is actively involved in Linux development, working with kernel developers on testing, review, and design of new Linux kernel–user-space APIs.
- Since 2004, he has been the maintainer of the Linux *man-pages* project, which provides the manual pages documenting the Linux kernel–user-space and GNU C library APIs.

Linux/UNIX IPC Programming: course contents in detail

Topics marked with an asterisk (*) are optional, and will be covered on an as-needed basis and/or as time permits

1. **Course Introduction**
2. **IPC: Introduction and Overview**
   - Categorizing IPC
   - Choosing an IPC mechanism
3. **Pipes and FIFOs**
   - Creating and using pipes
   - Connecting filters with pipes
   - FIFOs
4. **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
5. **UNIX Domain Sockets: Ancillary Data (*)**
   - Ancillary data
   - Ancillary message types
   - Example: passing a file descriptor over a socket
   - Further details
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. **eventfd**
   - Semantics of eventfd
   - eventfd operations
   - Semaphore semantics for eventfd
8. **Alternative I/O Models**
   - Nonblocking I/O
   - Signal-driven I/O
   - I/O multiplexing: *poll()*
   - Problems with *poll()* and *select()*
   - The *epoll* API
   - *epoll* events
   - *epoll*: edge-triggered notification
   - *epoll*: API quirks
   - Event-loop programming
9. **POSIX IPC**
10. **POSIX Semaphores**
    - Overview
    - Named semaphores
    - Semaphore operations
    - Synchronizing access to a shared resource
    - Unnamed semaphores
11. **POSIX Shared Memory**
    - Creating and opening shared memory objects
    - Using shared memory objects
    - Synchronizing access to shared memory
12. **POSIX Message Queues**
    - Overview
    - Opening, closing, and unlinking a message queue
    - Message queue attributes
    - Sending and receiving messages
    - The *mqueue* filesystem
    - Message queue limits and defaults
    - Message notification
    - Message notification via a signal
    - Message notification via a thread
13. **Other IPC methods (*)**
    - Pseudoterminals
    - File locks
    - Cross-memory attach
    - Shared file mappings