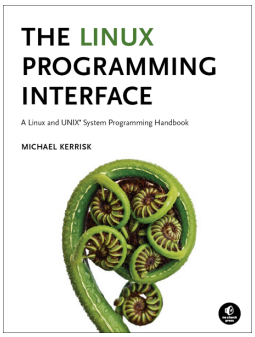
Linux/UNIX IPC Programming

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.



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 including file I/O using system calls, signals and process lifecycle (*fork()*, *exec()*, *wait()*). Such knowledge can be obtained in 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

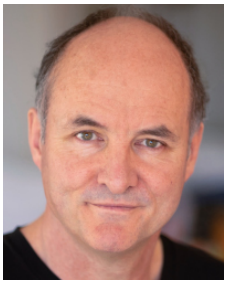
For inquiries about courses and consulting, you can contact us in the following ways:

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

Prices 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 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 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. **Fundamentals**
   - Error handling
   - Notes on code examples
4. **Pipes and FIFOs**
   - Preamble: file descriptors and file descriptor duplication
   - Creating and using pipes
   - FIFOs
   - Connecting filters with pipes
5. **Sockets: Introduction**
   - Socket types and domains
   - Creating and binding a socket (*socket()* and *bind()*)
   - Overview of stream sockets
   - *listen()* and pending connections
   - *accept()* and *connect()*
   - I/O on stream sockets
   - Overview of datagram sockets
   - I/O on datagram sockets
6. **UNIX Domain Sockets**
   - UNIX domain stream sockets
   - UNIX domain datagram sockets
   - Further details of UNIX domain sockets
7. **UNIX Domain Sockets: Ancillary Data (*)**
   - Ancillary message types
   - *sendmsg()*, *recvmsg()*, and *struct msghdr*
   - *struct msghdr* in more detail
   - Ancillary data and *struct cmsghdr*
   - Example: passing a file descriptor over a socket
8. **Internet Domain Sockets**
   - Internet domain sockets
   - Data-representation issues
   - Presentation-format addresses
   - Loopback and wildcard addresses
   - Internet domain stream sockets example
9. **Internet Domain Sockets: Address Conversion**
   - Host addresses and port numbers
   - Host and service conversion
   - Internet domain sockets example with *getaddrinfo()*
10. **Sockets: Further Details**
    - Socket shutdown (*shutdown()*)
    - Socket options
    - TCP TIME-WAIT state and `SO_REUSEADDR`
11. **Alternative I/O Models**
    - Nonblocking I/O
    - Signal-driven I/O
    - I/O multiplexing: *poll()*
    - Event-loop programming
12. **Alternative I/O Models: *epoll***
    - Problems with *poll()* and *select()*
    - The *epoll* API
    - *epoll* events
    - Performance considerations
    - Edge-triggered notification
    - *epoll* API quirks
13. **eventfd**
    - Overview of eventfd
    - eventfd operations
    - Semaphore semantics for eventfd
14. **POSIX IPC**
15. **POSIX Semaphores**
    - Overview
    - Named semaphores
    - Semaphore operations
    - Unnamed semaphores
16. **POSIX Shared Memory**
    - Creating and opening shared memory objects
    - Using shared memory objects
    - Synchronizing access to shared memory
17. **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 (*)
18. **Other IPC methods (*)**
    - Pseudoterminals
    - File locks
    - Cross-memory attach
    - Shared file mappings