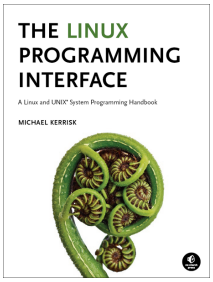
Linux/UNIX Sockets Programming

Course code: M7D-SOCKP01

This course covers network programming using the sockets API on Linux and UNIX systems. Topics covered include: the sockets API; sockets programming in the UNIX and Internet domains; alternative I/O models (*poll()*, *epoll*, nonblocking I/O); TCP/IP fundamentals; TCP in detail; and troubleshooting and monitoring. Detailed presentations coupled with many carefully designed practical exercises provide participants with the knowledge needed to write and understand complex network applications.



Audience and prerequisites

The primary audience for this course is programmers developing network applications for Linux and UNIX systems, or programmers porting such applications from other operating systems (e.g., Windows) to Linux or UNIX.

To get the most out of the course, participants should have:

- Good reading knowledge of the C programming language
- Working knowledge of the fundamental system programming topics covered in the *Linux System Programming Essentials* (M7D-SPESS01) course. This includes file descriptors and file I/O, signals, and the process lifecycle (*fork()*, *exec()*, *wait()*, *exit()*)
- Solid programming experience in a language suitable for completing the course exercises (e.g., C, C++, Go, or Rust)
- Knowledge of basic UNIX/Linux shell commands

Previous network programming experience is *not* required.

Course duration and format

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

Course materials

- A course book (written by the trainer) that includes 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 a large set of example programs 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) 2488 6180 (German landline)

Prices and further details

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

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.
- He has more than two decades of experience as a teacher and trainer, and first 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 Sockets Programming: course contents in detail

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

1. **Course Introduction**
2. **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
3. **Internet Domain Sockets**
   - Internet domain sockets
   - Data-representation issues
   - Presentation-format addresses
   - Loopback and wildcard addresses
   - Internet domain stream sockets example
4. **Internet Domain Sockets: Address Conversion**
   - Host addresses and port numbers
   - Host and service conversion
   - Internet domain sockets example with *getaddrinfo()*
5. **Sockets: Further Details**
   - Socket shutdown (*shutdown()*)
   - Socket options
   - TCP TIME-WAIT state and `SO_REUSEADDR`
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. **Alternative I/O Models**
   - Nonblocking I/O
   - Signal-driven I/O
   - I/O multiplexing: *poll()*
   - Event-loop programming
9. **Alternative I/O Models: *epoll***
   - Problems with *poll()* and *select()*
   - The *epoll* API
   - Creating an *epoll* instance: *epoll_create()*
   - Populating the interest list: *epoll_ctl()*
   - *epoll* events
   - Waiting for events: *epoll_wait()*
   - Performance considerations
   - Edge-triggered notification
   - *epoll* API quirks
10. **Displaying Sockets (*)**
    - *ss*
    - *netstat*

The following are some of the **other courses taught by Michael Kerrisk**. Custom courses are also available upon request. Further details on these and other courses can be found at `http://man7.org/training/`. For course inquiries please email training@man7.org or phone +49 (89) 2488 6180 (German landline).

Linux/UNIX System Programming

Course code: M7D-LUSP01 (5 days)

This course covers the APIs used to build system-level applications on Linux and UNIX systems ranging from embedded processors to enterprise servers. The presentations and practical exercises provide participants with the knowledge needed to write complex system, network, and multithreaded applications. Topics include: file I/O; signals; process creation and termination; program execution; POSIX threads; interprocess communication, and I/O multiplexing (*poll()*, *epoll*).

Linux Security and Isolation APIs

Course code: M7D-SECISOL02 (4 days)

Covering topics including control cgroups (cgroups), namespaces (with a deep dive into user namespaces), capabilities, and seccomp (secure computing), this course provides a deep understanding of the low-level Linux features used to design, build, and troubleshoot container, virtualization, and sandboxing frameworks.