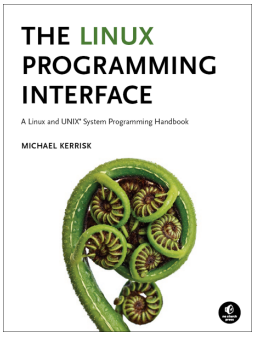
man7.org
Training and Consulting

Linux/UNIX Network Programming

Course code: M7D-NWP02

This course covers network programming using the sockets API on Linux and UNIX systems. Topics covered include: the sockets API; TCP/IP fundamentals; sockets programming in the UNIX and Internet domains; TCP in detail; troubleshooting and monitoring; and alternative I/O models (*poll()*, *epoll*, *select()*). Detailed presentations coupled with many carefully designed practical exercises provide participants with the knowledge needed to write complex network applications. The course touches on some topics specific to Linux, but most of the material is applicable on all UNIX implementations, so that it is also valuable to developers working on other systems.



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. By the completion of the course, participants will have the understanding needed to write advanced network applications on a Linux or UNIX system.

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

Previous network programming experience is *not* required.

Course duration and format

Three 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

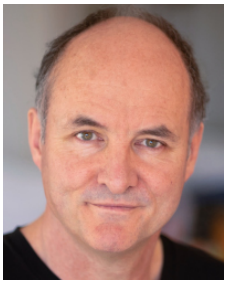
For inquiries about courses and consulting, you can contact us in the following ways:

- Email: training@man7.org
- Phone: +49 (89) 2155 2990 (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/nw_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 Network Programming: course contents in detail

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

1. **Course Introduction**
2. **Fundamental Concepts**
   - System calls and library functions
   - Error handling
   - System data types
   - Notes on code examples
3. **File I/O**
   - File I/O overview
   - *open()*, *read()*, *write()*, and *close()*
4. **Open File Descriptions and Descriptor Duplication**
   - Relationship between file descriptors and open files
   - Duplicating file descriptors
5. **Signals**
   - Overview of signals
   - Signal dispositions
   - Useful signal-related functions
   - Signal handlers
   - Signal sets, the signal mask, and pending signals
   - Designing signal handlers
6. **Process Lifecycle**
   - Creating a new process: *fork()*
   - Process termination
   - Monitoring child processes
   - Orphans and zombies
   - The `SIGCHLD` signal
   - Executing programs: *execve()*
7. **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
8. **UNIX Domain Sockets: Ancillary Data**
   - Ancillary data
   - Ancillary message types
   - Example: passing a file descriptor over a socket
   - Further details
9. **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
10. **Alternative I/O Models**
    - Nonblocking I/O
    - Signal-driven I/O
    - I/O multiplexing: *poll()*
11. **Alternative I/O Models: *epoll***
    - Problems with *poll()* and *select()*
    - The *epoll* API
    - *epoll* events
    - *epoll*: edge-triggered notification
    - *epoll*: API quirks
    - Event-loop programming
12. **TCP/IP Overview (*)**
    - The TCP/IP protocol stack
    - The link layer
    - The network layer: IP
    - The transport layer
    - Port numbers
    - User Datagram Protocol (UDP)
    - Transmission Control Protocol (TCP)
13. **Useful Tools (*)**
    - Displaying devices and addresses
    - *nestat* and *ss*
    - *tcpdump* and *wireshark*
14. **Raw Sockets (*)**
    - Overview of creating and using raw sockets
    - Raw sockets example
15. **Daemons (*)**
    - Creating a daemon
    - Reinitializing a daemon