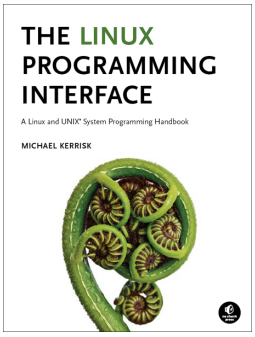
man7.org
Training and Consulting

Linux/UNIX Network Programming

Course code: M7D-NWP03

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*, non-blocking 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 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. 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

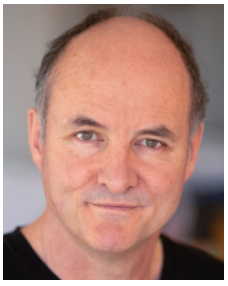
For inquiries about courses and consulting, you can contact us in the following ways:

- Email: training@man7.org
- Phone: +49 (89) 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/nwp/`.

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 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. **TCP/IP Overview**
    - The TCP/IP protocol stack
    - The link layer
    - The network layer: IP
    - The transport layer
    - Port numbers
    - User Datagram Protocol (UDP)
    - Displaying sockets and capturing packets
11. **Transmission Control Protocol (TCP)**
    - Overview of TCP
    - Segments
    - Sequence numbers
    - Acknowledgements and retransmissions
    - TCP header
    - TCP state machine
    - TCP connection establishment and termination
12. **Transmission Control Protocol: Further Details (*)**
    - Flow control and congestion control
    - TCP sliding window
13. **Displaying Sockets**
    - *ss*
    - *netstat*
14. **Capturing network packets**
    - *tcpdump*
    - *wireshark*
    - Packet filtering
15. **Packet filtering: capture filters**
    - Capture filters
    - Capture filters and BPF (*)
16. **Packet filtering: display filters**
    - *wireshark* display filters
17. **Other Networking Tools (*)**
    - Displaying devices and addresses
    - Testing connectivity and routes
18. **Raw Sockets (*)**
    - Overview of creating and using raw sockets
    - Raw sockets example
19. **Open File Descriptions and Descriptor Duplication (*)**
    - Relationship between file descriptors and open files
    - Duplicating file descriptors