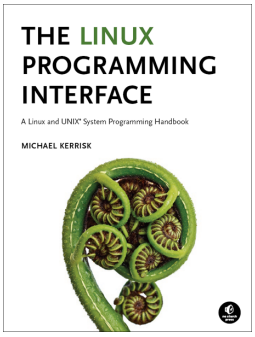
man7.org
Training and Consulting

Linux/UNIX System Programming

Course code: M7D-SP01

This course provides a deep understanding of the operating system architecture and low-level interfaces required to build system-level applications on Linux and UNIX systems ranging from embedded processors to enterprise servers. Detailed presentations coupled with many carefully designed practical exercises provide participants with the knowledge needed to write complex system, network, and multithreaded applications. The course dives into many 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++, D, Go, Rust, or Python)
- Knowledge of basic UNIX/Linux shell commands

Previous system programming experience is *not* required.

Course duration and format

Five days, with up to 50% devoted to practical sessions.

Course materials

- Two course books (written by the trainer) that include all slides and exercises presented in the course
- A copy of the trainer's book, *The Linux Programming Interface*
- A source code tarball containing around 30,000 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, dates, and further details

For course prices, upcoming course dates, and further information about the course, please visit the course web page, *http://man7.org/training/sys_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 System Programming: course contents in detail

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

1. **Fundamental concepts**
   - System calls & library functions
   - Error handling
   - System data types
   - Notes on code examples
2. **File I/O**
   - File descriptors
   - *open()*, *close()*, *read()*, *write()*
   - Seeking to a file offset: *lseek()*
   - Atomicity and races
   - Relationship between file descriptors and open files
   - Duplicating file descriptors
   - Open file status flags
3. **File I/O buffering**
   - Buffering in *stdio* and kernel
   - Controlling buffering
4. **File attributes**
   - Filesystems and i-nodes
   - *stat()*
   - File timestamps
   - Ownership and permissions
   - Changing file attributes
5. **Directories and links**
   - Hard and soft links
   - Directories
   - Current working directory
   - Operating relative to a directory file descriptor
6. **Processes**
   - Process IDs
   - Memory layout of a process
   - Command-line arguments
   - Environment list
   - Process groups and sessions (*)
   - Nonlocal gotos (*)
7. **Process credentials**
   - Password and group file
   - User and group IDs
   - Real, effective, saved set IDs
   - Supplementary group IDs
   - Retrieving process credentials
8. **Signals**
   - Signal types; default actions
   - Setting signal dispositions
   - Signal handlers
   - Signal sets
   - Blocked and pending signals
   - Designing signal handlers
   - Reentrancy; async-signal-safety
   - Restarting system calls
   - Sending signals (*)
   - The `SA_SIGINFO` flag (*)
   - Realtime signals (*)
   - Advanced features (*signalfd()*, *sigsuspend()*, *sigwaitinfo()*) (*)
9. **Process lifecycle**
   - Process creation: *fork()*
   - File descriptors and *fork()*
   - Process termination
   - Exit handlers (*)
   - Monitoring child processes
   - Orphans and zombies
   - The `SIGCHLD` signal
10. **Executing programs**
    - *execve()*
    - *exec()* library functions
    - File descriptors and *exec()*
11. **Privileged programs**
    - Set-UID and set-GID programs
    - Changing process credentials
    - Guidelines for writing privileged programs
12. **Threads**
    - Thread creation
    - Thread termination
    - Joining and detaching
    - Threads versus processes
    - Critical sections
    - Mutexes
    - Condition variables
    - Overview of barriers, read-write locks, and spin locks
13. **Interprocess communication overview**
    - Taxonomy of IPC facilities
    - Comparison of IPC facilities
14. **Pipes and FIFOs**
    - Creating and using pipes
    - Using pipes to connect filters
    - FIFOs
    - I/O semantics
15. **Introduction to sockets**
    - Socket types and domains
    - Sockets system calls
    - Stream sockets
    - Datagram sockets
16. **UNIX domain sockets**
    - UNIX domain stream and datagram sockets
    - Socket permissions
17. **Internet domain sockets**
    - Internet socket addresses
    - Data representation issues
    - *getaddrinfo()*, *getnameinfo()*
    - Socket options
    - Other sockets-specific APIs
18. **Alternative I/O models**
    - Nonblocking I/O
    - Signal-driven I/O
    - *poll()* and *select()*
    - The *epoll* API
19. **Introduction to POSIX IPC**
    - Common features of POSIX IPC
    - Contrast with System V IPC
20. **POSIX semaphores**
    - Named semaphores
    - Semaphore operations
    - Synchronizing access to a shared resource
    - Unnamed semaphores
21. **POSIX shared memory**
    - Opening SHM objects
    - Using SHM objects
    - Synchronizing access to SHM
    - Unmapping and removing SHM objects
22. **POSIX message queues (*)**
    - Open and closing MQs
    - Message queue attributes
    - Sending + receiving messages
    - The *mqueue* filesystem
    - Message queue limits
    - Message notification
23. **Daemons (*)**
    - Steps in creating a daemon
    - Reinitializing a daemon
24. **Time (*)**
    - Calendar time, process time
    - POSIX clocks
25. **Timers and sleeping (*)**
    - Traditional timer APIs
    - POSIX interval timers
    - *timerfd*
    - Sleep APIs