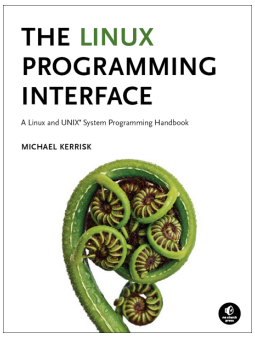
man7.org
Training and Consulting

Linux/UNIX System Programming Fundamentals

Course code: M7D-SPINTRO01

This two-day course provides a sound understanding of the basic operating system features and low-level interfaces (principally, system calls and library functions) that are used 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-level applications. The course provides the fundamental knowledge that is required by the follow-on course *Linux/UNIX IPC Programming* (M7D-IPC01) and is also recommended as preparation for the course *Linux Security and Isolation APIs* (M7D-SECISOL02).



Audience and prerequisites

The audience for this course includes programmers developing and porting system-level 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

Two 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
- A copy of the trainer's book, *The Linux Programming Interface*
- A source code tarball containing more than 35,000 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 and further details

For course prices and further information, please visit the course web page, *http://man7.org/training/spintro/*.

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 Fundamentals: 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()*, *close()*
4. **File I/O Buffering**
   - Kernel buffering
   - User-space (*stdio*) buffering
   - Controlling kernel buffering
5. **File I/O: Further Details**
   - The file offset and *lseek()*
   - Atomicity
   - Relationship between file descriptors and open files
   - Duplicating file descriptors
   - File status flags (and *fcntl()*)
6. **Files**
   - Inodes
   - Retrieving file information: *stat()*
   - File mode
   - Changing file attributes
7. **Directories and Links (*)**
   - Directories and (hard) links
   - Symbolic links
   - Current working directory
   - Operating relative to a directory (*openat()* etc.)
   - Scanning directories
8. **Processes**
   - Process IDs
   - Process memory layout
   - Command-line arguments
   - The environment list
   - Process groups and sessions (*)
   - Nonlocal gotos
9. **Process Credentials**
   - Users and groups
   - Process credentials
   - Retrieving process credentials
10. **Signals**
    - Signal dispositions
    - Signal handlers
    - Useful signal-related functions
    - Signal sets, the signal mask, and pending signals
    - Designing signal handlers
11. **Signals: Signal Handlers**
    - Async-signal-safe functions
    - Interrupted system calls
    - `SA_SIGINFO` signal handlers
    - Addendum: the signal trampoline (*)
12. **Process Creation and Termination**
    - Creating a new process: *fork()*
    - File descriptors and *fork()*
    - Process termination
    - Monitoring child processes
    - Orphans and zombies
    - The `SIGCHLD` signal
13. **Executing Programs**
    - Executing programs: *execve()*
    - The *exec()* library functions
    - File descriptors and *exec()*
14. **Daemons (*)**
    - Creating a daemon
    - Reinitializing a daemon
15. **System Call Tracing with *strace* (*)**
    - Getting started
    - Tracing child processes
    - Filtering *strace* output