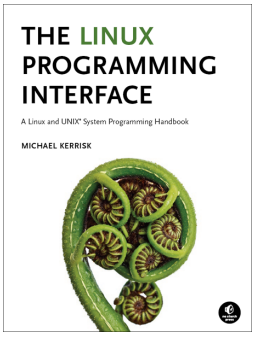
man7.org
Training and Consulting

Linux System Programming Fundamentals

NDC TechTown Workshop, 29-30 August 2022

http://man7.org/training/spintro_ndc/

This two-day workshop provides a solid understanding of the operating system architecture 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. By completion of the workshop, participants will have a good understanding of the construction of many common Linux and UNIX programs (e.g., the shell, *ls(1)*, and *cp(1)*).



Audience and prerequisites

The audience for this workshop 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 workshop, participants should have:

- Good reading knowledge of the C programming language
- Solid programming experience in a language suitable for completing the practical exercises (e.g., C, C++, D, or Rust)
- A familiarity with basic UNIX/Linux shell commands will be helpful

Previous system programming experience is *not* required.

Workshop duration and format

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

Computer set-up

You'll need a laptop with Linux installed–either as a native install or inside a virtual machine (VM). In the latter case, you should ensure that the VM has working Internet access. The system should have a C compiler (e.g., gcc) and "make" installed. (It's likely that both of these are provided as part of the default install, if you are setting up a new system.)

Workshop materials

- A course book (written by the trainer) that includes all slides and exercises presented in the workshop
- An electronic 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

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 System Programming Fundamentals: workshop 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 and Files**
   - File I/O overview
   - *open()*, *read()*, *write()*, *close()*
   - The file offset and *lseek()*
   - Relationship between file descriptors and open files
   - Duplicating file descriptors
   - File status flags (and *fcntl()*)
   - Retrieving file info: *stat()*
4. **Processes**
   - Process IDs
   - Process memory layout
   - Command-line arguments
   - The environment list
   - Process credentials
   - The `/proc` filesystem
5. **Signals**
   - Overview of signals
   - Signal dispositions
   - Signal handlers
   - Useful signal-related functions
   - Signal sets, the signal mask, and pending signals
   - Designing signal handlers
6. **Signals: Signal Handlers**
   - Async-signal-safe functions
   - Interrupted system calls
   - `SA_SIGINFO` handlers (*)
   - The signal trampoline (*)
7. **Process Lifecycle**
   - Creating a new process: *fork()*
   - Process termination
   - Monitoring child processes
   - Orphans and zombies
   - The `SIGCHLD` signal
   - Executing programs: *execve()*
8. **System Call Tracing (*)**
   - *strace* basics
   - Tracing child processes
   - Filtering *strace* output
9. **Pipes and FIFOs**
   - Creating and using pipes
   - FIFOs (*)
   - Connecting filters (*)
10. **Alternative I/O Models**
    - Nonblocking I/O
    - Signal-driven I/O
    - I/O multiplexing: *poll()*
    - Event-loop programming (*)
11. **Alternative I/O Models: *epoll***
    - Problems with *poll()* and *select()*
    - The *epoll* API
    - *epoll* events
    - Performance considerations
    - Edge-triggered notification (*)

The following are a few 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) 2155 2990 (German landline).

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.

Linux/UNIX Network Programming

Course code: M7D-NWP03 (3 days)

This course covers sockets programming (both UNIX and Internet domain sockets), and the use of relevant I/O techniques for working with sockets (*poll()*, *epoll*, nonblocking I/O). In addition, we look at the TCP/IP protocol stack (including details of TCP such as the 3-way handshake and the TCP state machine), the use of monitoring and tracing tools (*ss*, *netstat*, and *tcpdump*/*wireshark)*, and raw sockets.

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*).

Building and Using Shared Libraries on Linux

Course code: M7D-SHLIB04 (2.5 days)

This course describes how to design, build, and use shared libraries on Linux. Topics include: fundamentals of library creation and use; shared library versioning; symbol resolution; library search order; executable and linking format (ELF); dynamically loaded libraries; controlling symbol visibility; and symbol versioning.