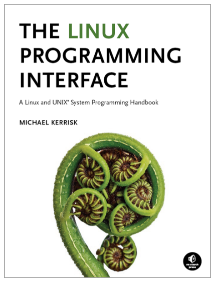
man7.org
Training and Consulting

Building and Using Shared Libraries on Linux

Course code: M7D-SHLIB01

This course provides a thorough understanding of the process of designing, building, and using shared libraries on Linux. Detailed presentations coupled with carefully designed practical exercises provide participants with the knowledge needed to understand, design, create, and deploy shared libraries.


THE LINUX PROGRAMMING INTERFACE
A Linux and UNIX System Programming Handbook
MICHAEL KERRISK

Audience and prerequisites

The primary audience comprises designers and programmers building and deploying shared libraries on Linux. Systems administrators are likely to also find the course of benefit for the purpose of troubleshooting problems with shared libraries.

Participants should have a good reading knowledge of the C programming language and some programming experience in a language suitable for completing the course exercises (e.g., C, C++). No previous experience of working with shared libraries is required.

Course materials

- A course book (written by the trainer) that includes all course slides and exercises
- An electronic copy of the trainer's book, *The Linux Programming Interface*
- A source code tarball containing all of the example programs written by the trainer to accompany the presentation

Course duration and format

One day, with around 40% devoted to practical sessions.

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, upcoming course dates, and further information about the course, please visit the course web page, *http://man7.org/training/shlib/*.

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.

Building and Using Shared Libraries on Linux: course contents in detail

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

1. **Course Introduction**
2. **Fundamentals of Shared Libraries**
   - Background
   - Basics of shared library creation and use
   - The shared library soname
   - Summary: shared library creation, linking, and loading
3. **Versioning and Installation**
   - Shared library versioning
   - Shared library real names, sonames, and linker names
   - Installing shared libraries
   - *ldconfig*
   - Further details
4. **The Dynamic Linker**
   - Rpath: specifying library search paths in an object
   - Finding shared libraries at run time
   - Symbol resolution, library load order, and link maps
   - Debugging the operation of the dynamic linker
5. **Shared Libraries and the Static Linker**
   - How the static linker finds shared libraries
   - Correctly handling secondary dependencies at link time
6. **ELF and Program Execution**
   - Introduction
   - ELF file layout
   - ELF sections
   - Useful commands: *readelf* and *objdump*
   - How programs get run
7. **Dynamically Loaded Libraries (dlopen)**
   - Overview
   - Opening a shared library: *dlopen()*
   - Obtaining the address of a symbol: *dlsym()*
   - The dlopen API: example
   - The dlopen API: further details
8. **Symbol Visibility**
   - Controlling symbol visibility
   - Using version scripts to control symbol visibility
   - Preloading shared libraries
   - Weak symbols (*)
9. **Symbol Versioning**
   - Introduction
   - Creating a symbol-versioned library
   - Transitioning an existing library to symbol versioning
   - Advantages of symbol versioning
   - Symbol versioning: further details
10. **GOT, PLT, and Lazy Binding (*)**
    - The GOT and PLT
    - Lazy binding
    - Lazy binding and the PLT: in pictures
    - Lazy binding and the PLT: code
    - Performance considerations