However, POSIX message queues also have some disadvantages compared to System V message queues:

- POSIX message queues are less portable. This problem applies even across Linux systems, since message queue support is available only since kernel 2.6.6.
- The facility to select System V messages by type provides slightly greater flexibility than the strict priority ordering of POSIX messages.

> There is a wide variation in the manner in which POSIX message queues are implemented on UNIX systems. Some systems provide implementations in user space, and on at least one such implementation (Solaris 10), the *mq_open()* manual page explicitly notes that the implementation can't be considered secure. On Linux, one of the motives for selecting a kernel implementation of message queues was that it was not deemed possible to provide a secure user-space implementation.

52.10 Summary

POSIX message queues allow processes to exchange data in the form of messages. Each message has an associated integer priority, and messages are queued (and thus received) in order of priority.

POSIX message queues have some advantages over System V message queues, notably that they are reference counted and that a process can be asynchronously notified of the arrival of a message on an empty queue. However, POSIX message queues are less portable than System V message queues.

Further information

[Stevens, 1999] provides an alternative presentation of POSIX message queues and shows a user-space implementation using memory-mapped files. POSIX message queues are also described in some detail in [Gallmeister, 1995].

52.11 Exercises

52-1. Modify the program in Listing 52-5 (`pmsg_receive.c`) to accept a timeout (a relative number of seconds) on the command line, and use *mq_timedreceive()* instead of *mq_receive()*.

52-2. Recode the sequence-number client-server application of Section 44.8 to use POSIX message queues.

52-3. Rewrite the file-server application of Section 46.8 to use POSIX message queues instead of System V message queues.

52-4. Write a simple chat program (similar to *talk(1)*, but without the *curses* interface) using POSIX messages queues.

52-5. Modify the program in Listing 52-6 (`mq_notify_sig.c`) to demonstrate that message notification established by *mq_notify()* occurs just once. This can be done by removing the *mq_notify()* call inside the `for` loop.