- Historically, System V shared memory was more widely available than *mmap()* and POSIX shared memory, although most UNIX implementations now provide all of these techniques.

With the exception of the final point regarding portability, the differences listed above are advantages in favor of shared file mappings and POSIX shared memory objects. Thus, in new applications, one of these interfaces may be preferable to System V shared memory. Which one we choose depends on whether or not we require a persistent backing store. Shared file mappings provide such a store; POSIX shared memory objects allow us to avoid the overhead of using a disk file when a backing store is not required.

54.6 Summary

A POSIX shared memory object is used to share a region of memory between unrelated processes without creating an underlying disk file. To do this, we replace the call to *open()* that normally precedes *mmap()* with a call to *shm_open()*. The *shm_open()* call creates a file in a memory-based file system, and we can employ traditional file descriptor system calls to perform various operations on this virtual file. In particular, *ftruncate()* must be used to set the size of the shared memory object, since initially it has a length of zero.

We have now described three techniques for sharing memory regions between unrelated processes: System V shared memory, shared file mappings, and POSIX shared memory objects. There are several similarities between the three techniques. There are also some important differences, and, except for the issue of portability, these differences favor shared file mappings and POSIX shared memory objects.

54.7 Exercise

54-1. Rewrite the programs in Listing 48-2 (`svshm_xfr_writer.c`) and Listing 48-3 (`svshm_xfr_reader.c`) to use POSIX shared memory objects instead of System V shared memory.