Producer–Consumer Problem

In Section 6.6.1, we had presented a semaphore-based solution to the producer–consumer problem using a bounded buffer. In this project, we will design a programming solution to the bounded-buffer problem using the producer and consumer processes shown in Figures 6.10 and 6.11. The solution presented in Section 6.6.1 uses three semaphores: empty and full, which count the number of empty and full slots in the buffer, and mutex, which is a binary (or mutual-exclusion) semaphore that protects the actual insertion or removal of items in the buffer. For this project, standard counting semaphores will be used for empty and full, and a mutex lock, rather than a binary semaphore, will be used to represent mutex. The producer and consumer—running as separate threads—will move items to and from a buffer that is synchronized with these empty, full, and mutex structures. You can solve this problem using either Pthreads or the Win32 API.

The Buffer

Internally, the buffer will consist of a fixed-size array of type buffer_item (which will be defined using a typedef). The array of buffer_item objects will be manipulated as a circular queue. The definition of buffer_item, along with the size of the buffer, can be stored in a header file such as the following:

```c
/* buffer.h */
typedef int buffer_item;
#define BUFFER_SIZE 5
```

The buffer will be manipulated with two functions, insert_item() and remove_item(), which are called by the producer and consumer threads, respectively. A skeleton outlining these functions appears in Figure 6.27.