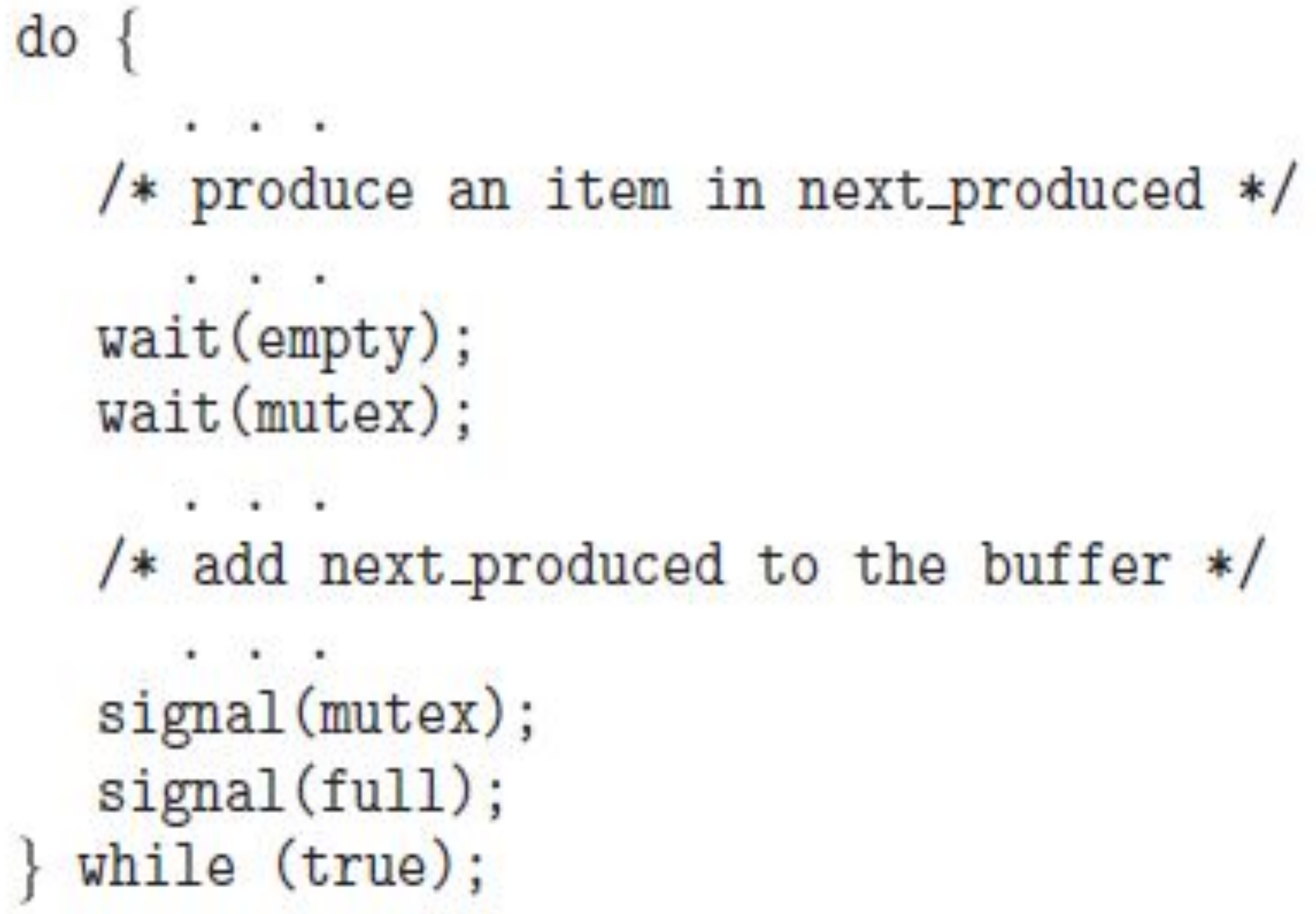
Classical problems of synchronization

The Bounded-Buffer Problem

- The producer and consumer processes share the following data structures:

  ```
  int  n;
  semaphore mutex = 1;
  semaphore empty = n;
  semaphore full = 0
  ```
- We assume that the pool consists of **n buffers**, each capable of holding one item.
- The **mutex semaphore** provides mutual exclusion for accesses to the buffer pool and is initialized to the value **1**.
- The empty and full semaphores count the number of empty and full buffers.
- The semaphore **empty** is initialized to the value **n**; the semaphore **full** is initialized to the value **0**.

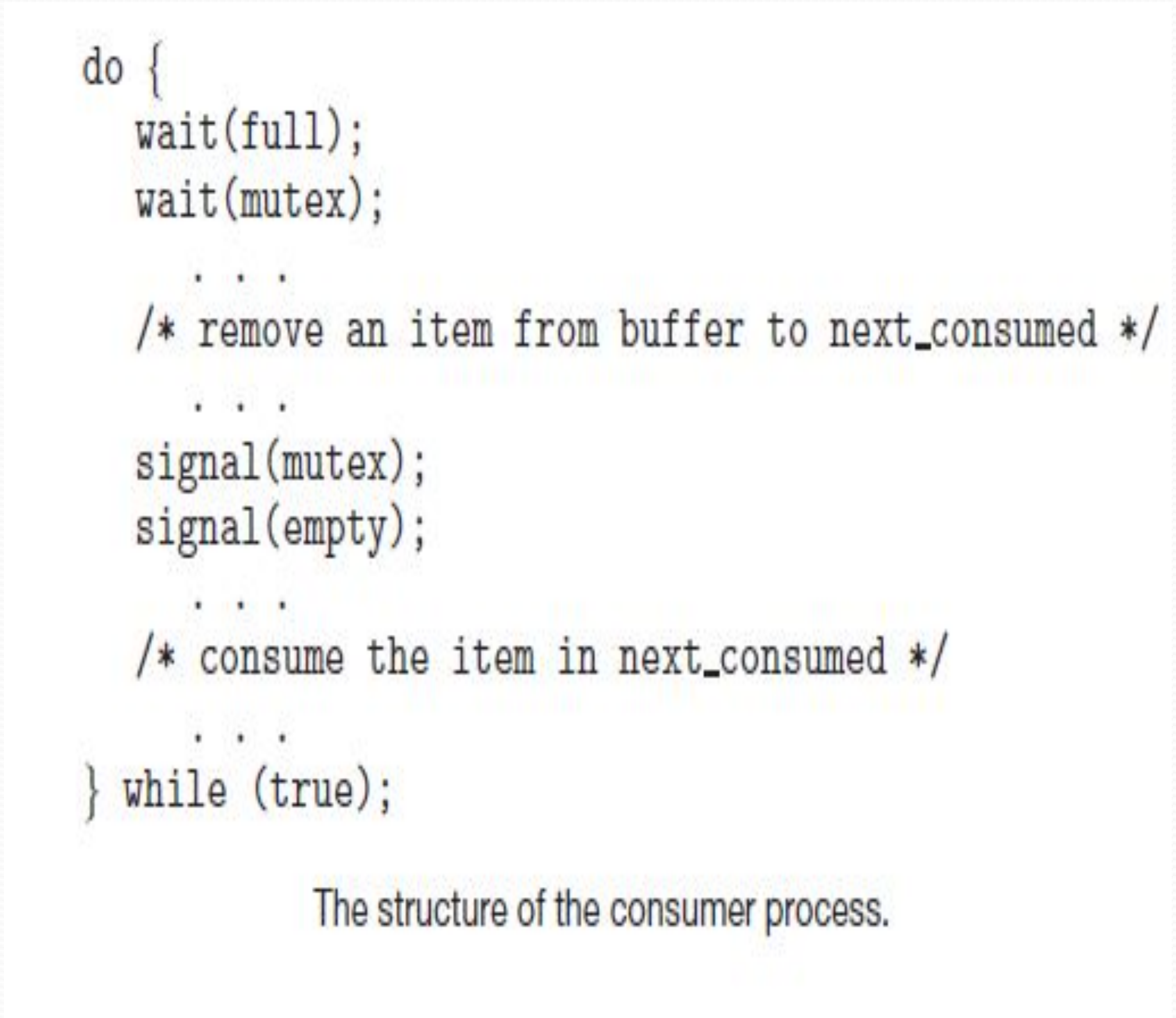
The Bounded-Buffer Problem



```
do {
      . . .
   /* produce an item in next_produced */
      . . .
   wait(empty);
   wait(mutex);
      . . .
   /* add next_produced to the buffer */
      . . .
   signal(mutex);
   signal(full);
} while (true);
```

The structure of the producer process.

The Bounded-Buffer Problem



```
do {
   wait(full);
   wait(mutex);
      . . .
   /* remove an item from buffer to next_consumed */
      . . .
   signal(mutex);
   signal(empty);
      . . .
   /* consume the item in next_consumed */
      . . .
} while (true);
```

The structure of the consumer process.