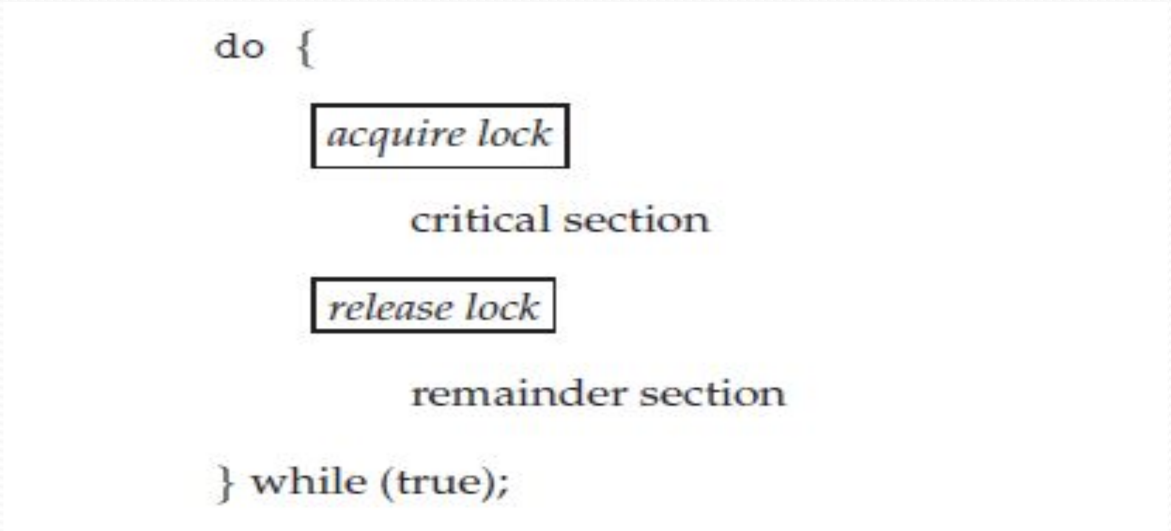
Mutex Locks

Mutex Locks

- Used to protect critical regions and thus prevent race conditions.
- A process must acquire the lock before entering a critical section; it releases the lock when it exits the critical section.
- The acquire()function acquires the lock, and the release() function releases the lock.



```
do {
     acquire lock
          critical section
     release lock
          remainder section
} while (true);
```

Solution to the critical-section problem using mutex locks.

Mutex Locks

- The definition of acquire() is as follows:

```
acquire()
   {
       while (!available); /* busy wait */
        available = false;
    }
```

- The definition of release() is as follows:

```
release()
   {
       available = true;
    }
```

Mutex Locks

- Calls to either acquire() or release() must be performed atomically.

Disadvantage

- Busy waiting.
- While a process is in its critical section, any other process that tries to enter its critical section must loop continuously in the call to acquire().
- This type of mutex lock is also called a **spinlock** because the process "spins" while waiting for the lock to become available.
- Busy waiting wastes CPU cycles that some other process might be able to use productively.