第 6 章作业

6.8

6.8 Race conditions are possible in many computer systems. Consider an online auction system where the current highest bid for each item must be maintained. A person who wishes to bid on an item calls the `bid(amount)` function, which compares the amount being bid to the current highest bid. If the amount exceeds the current highest bid, the highest bid is set to the new amount. This is illustrated below:

```
void bid(double amount) {
   if (amount > highestBid)
      highestBid = amount;
}
```

Describe how a race condition is possible in this situation and what might be done to prevent the race condition from occurring.

答:

如果有两个及以上的人同时出价，且都比现在的最高价格高，那么可能会有多条

```
highestBid = amount;
```

并发执行，引起竞争。因为不知道先执行了哪个，所以最后 highestBid 的结果不一定是正确的。

可以规定在同一时间只能有一个人出价，即同一时间只有一个人能调用 bid() 函数，可以把 bid() 函数放入临界区，用Peterson方法或互斥锁或信号量等方法来解决。

6.13

6.13 The first known correct software solution to the critical-section problem for two processes was developed by Dekker. The two processes, P_0 and P_1, share the following variables:

```
boolean flag[2]; /* initially false */
int turn;
```

The structure of process P_i (i == 0 or 1) is shown in Figure 6.18. The other process is P_j (j == 1 or 0). Prove that the algorithm satisfies all three requirements for the critical-section problem.

```
while (true) {
   flag[i] = true;

   while (flag[j]) {
      if (turn == j) {
         flag[i] = false;
         while (turn == j)
            ; /* do nothing */
         flag[i] = true;
      }
   }

      /* critical section */

   turn = j;
   flag[i] = false;

      /* remainder section */
}
```

Figure 6.19 The structure of process P_i in Dekker's algorithm.

答:

只有当 flag[j] == false 时，Pi 才能进入临界区，此时 flag[i] == true，Pj 会进入 while(flag[i]) 循环。如果 turn == i，则 Pj 会执行 flag[j] = false 然后阻塞，直到 Pi 出临界区，执行 turn = j，Pj 会执行flag[j] = true，然后进入临界区，而Pi将阻塞在 while(flag[j]) 循环直到 Pj 出临界区执行 turn = i 。所以，turn 为 i 、j 分别对应了Pi、Pj 能否进入临界区，flag[] 代表 Pi、Pj 是否准备好。

（1）互斥成立：如果flag[i] == flag[j] == true，那么Pi、Pj 谁会被阻塞由 turn 来决定。因为 turn 在同一时间只有一个值，说明同一时间只有一个进程能够进入临界区。

（2）进步要求满足：只有当 flag[j] == false 时，Pi 才能进入临界区。而只要 flag[i] == true，turn == i， flag[j] == false 就会成立，Pi 就会进入临界区。对于 Pj 同理。

（3）有限等待要求满足：只要 Pi 出临界区，执行 turn = j，flag[i] = false，就能解除 Pj 的阻塞，使之进入临界区，即只需要等待一次。对于 Pi 同理。

6.21

6.21 A multithreaded web server wishes to keep track of the number of requests it services (known as *hits*). Consider the two following strategies to prevent a race condition on the variable `hits`. The first strategy is to use a basic mutex lock when updating `hits`:

```
int hits;
mutex_lock hit_lock;

hit_lock.acquire();
hits++;
hit_lock.release();
```

A second strategy is to use an atomic integer:

```
atomic_t hits;
atomic_inc(&hits);
```

Explain which of these two strategies is more efficient.

答:

第二种方法效率更高，因为第一种基于互斥锁的方法存在上下文切换。