



操作系统原理

Operating Systems Principles

陈鹏飞
计算机学院



作业-6

1、“忙等待”这个词的含义是什么？操作系统中还有什么类型的等待？可以避忙等待吗？如何解决？

2、竞争条件在许多计算机系统中存在。考虑一个银行系统，它通过两个功能来维持账户余额：存款（金额）和取款（金额）。这两个函数传递的是要从银行账户余额中存入或取出的金额。假设一对夫妻共用一个银行账户。同时，丈夫调用 `withdraw()` 函数，妻子调用 `deposit()` 函数。描述竞争状况是如何发生的，以及如何防止竞争状况的发生。



作业-6

3、考虑下图所示的分配和释放进程的代码示例。

```
#define MAX_PROCESSES 255
int number_of_processes = 0;

/* the implementation of fork() calls this function */
int allocate_process() {
    int new_pid;

    if (number_of_processes == MAX_PROCESSES)
        return -1;
    else {
        /* allocate necessary process resources */
        ++number_of_processes;

        return new_pid;
    }
}

/* the implementation of exit() calls this function */
void release_process() {
    /* release process resources */
    --number_of_processes;
}
```

a. 指出竞争条件。

b. 假设有一个名为mutex的互斥锁，其操作为acquire () 和release ()。指出需要将锁放置在何处，以防止出现竞争条件。

c. 能否采用原子整数 `atomic_t number_of_processes = 0` 替代 `int number_of_processes = 0` 以防止竞争?



作业-6

4. 考虑一个版本的面包师算法，这个版本未使用变量 `choosing`，代码如下：

```
1 int number[n];
2 while (true) {
3     number[i] = 1 + getmax(number[], n);
4     for (int j = 0; j < n; j++){
5         while ((number[j] != 0) && (number[j],j) < (number[i],i)) { };
6     }
7     /* 临界区 */;
8     number[i] = 0;
9     /* 其余部分 */;
10 }
```

该版本是否违反了互斥原则？解释原因。



作业-6

5. 下面的问题曾用于一次测验：侏罗纪公园有一个恐龙博物馆和一个公园。有 m 名旅客和 n 辆车，每辆车只能容纳 1 名旅客。旅客在博物馆中逛一会儿后，排队乘坐旅行车。当一辆车可用时，它载入一名旅客，然后绕公园行驶任意长时间。若 n 辆车都已被旅客乘坐游玩，则想坐车的旅客需要等待；若一辆车已就绪，但没有旅客等待，那么这辆车等待。使用信号量同步 m 名旅客进程和 n 辆车进程。下面的代码框架是在教室的地板上发现的。忽略语法错误和丢掉的变量声明，请判定它是否正确。注意，P 和 V 分别对应于 semWait 和 semSignal。

```
resource Jurassic_Park()
sem car_avail := 0, car_taken := 0, car_filled := 0, passenger_released := 0
process passenger(i := 1 to num_passengers)
do true -> nap(int(random(1000*wander_time)))
P(car_avail); V(car_taken); P(car_filled)
P(passenger_released)
od
end passenger
process car(j := 1 to num_cars)
do true -> V(car_avail); P(car_taken); V(car_filled)
nap(int(random(1000*ride_time)))
V(passenger_released)
od
end car
end Jurassic_Park
```



作业-6

6. 下面对一个写者/多个读者问题的解法错在哪里？

```
int readcount;           // 共享, 初值为 0
Semaphore mutex, wrt;    // 共享, 初值为 1;
// 写者:                 // 读者:
                          semWait(mutex);
                          readcount := readcount + 1;
                          if readcount == 1 then semWait(wrt);
                          semSignal(mutex);
                          /*执行写操作*/
                          semSignal(wrt);
                          /*执行读操作*/
                          semWait(mutex);
                          readcount := readcount - 1;
                          if readcount == 0 then Up(wrt);
                          semSignal(mutex);
```



作业-6

7. Consider the following snapshot of a system:

	<u>Allocation</u>	<u>Max</u>	<u>Available</u>
	<i>A B C D</i>	<i>A B C D</i>	<i>A B C D</i>
T_0	3 1 4 1	6 4 7 3	2 2 2 4
T_1	2 1 0 2	4 2 3 2	
T_2	2 4 1 3	2 5 3 3	
T_3	4 1 1 0	6 3 3 2	
T_4	2 2 2 1	5 6 7 5	

Answer the following questions using the banker's algorithm:

- Illustrate that the system is in a safe state by demonstrating an order in which the threads may complete.
- If a request from thread T_4 arrives for (2, 2, 2, 4), can the request be granted immediately?
- If a request from thread T_2 arrives for (0, 1, 1, 0), can the request be granted immediately?
- If a request from thread T_3 arrives for (2, 2, 1, 2), can the request be granted immediately?



作业-6 选做

8. 本题演示了信号量用来协调 3 种进程的方法。睡在北极商店中的 6 名圣诞老人只能被下述情形唤醒：
- (1) 所有 9 头驯鹿都从南太平洋度假归来。(2) 有些小精灵在制作玩具时遇到了麻烦；为了让圣诞老人们多休息一会儿，只能在 3 个小精灵遇到麻烦时才能叫醒圣诞老人。在这 3 个小精灵解决它们的问题时，其他想要找圣诞老人的小精灵只能等这 3 个小精灵返回。如果圣诞老人醒来后发现 3 个小精灵及最后一头从热带度假归来的驯鹿在店门口等着，那么圣诞老人就决定让这些小朋友等到圣诞节以后，因为准备雪橇更加重要（假设驯鹿不想离开热带，因此它们要在那里待到最后可能的时刻）。最后的驯鹿一定要赶回来找圣诞老人，在套上雪橇之前，驯鹿会在温暖的棚子里等着。用信号量解决上述问题。

Deadline: 6月2日