同步互斥 lab 实验报告

2018 级 信息安全 管箫 18307130012

一、复旦早餐王问题

1、关系分析

互斥关系：对窗口的访问必须互斥的进行
同步关系：老板把煎饼果子放进篮子后，一队同学才能拿煎饼果子；老板娘把鸡蛋灌饼放进篮子后，另一队同学才能拿鸡蛋灌饼；当篮子里的早餐被拿走后，老板或者老板娘才能往里面放早餐。

2、伪代码描述

可以抽象为四个进程。
伪代码见 Exercise1.txt

3、问题分析

这道题是个典型的多生产者多消费者问题，并且生产者和消费者涉及到的是不同的资源类型。我们可以注意到，由于窗口的大小为 1，窗口互斥锁的存在并不一定是必须的。

二、小小打印店问题

1、关系分析

互斥关系：最多有 n 个顾客，打印机同时只能打印一份材料。
同步关系：只要有顾客存在，打印机就不会是空闲的。

2、伪代码描述

可以抽象为两个进程。
伪代码见 Exercise2.txt

3、问题分析

这道题是个典型的理发师睡觉问题，可以看作多生产者单消费者问题，其中打印机是消费者，消费理发者这个资源。

三、哲学家就餐问题

1、使用方法

本题实现中，我们采用限制同时就餐的哲学家数量为 4 个的方式，保证总有一位哲学家可以同时取到两只筷子（可以通过抽屉原理简单证明），并且就餐。而限制人数的方式是引入一个计数信号量。而筷子的归属通过定义好的互斥锁数组进行保证。

2、代码实现

引入一个计数信号量 Limit，初始化数值为 4，保证最大尝试就餐人数为 4。

```
sem_init(&Limit, 0, 4);
```

在取筷子和放筷子的函数部分，分别实现了对左手筷子和右手筷子互斥锁的请求和释放。

```c
void pickUp(int philosopherNumber) {
    // request chopsticks
    /*Your code here*/
    int left = (philosopherNumber + 1) % 5;
    int right = philosopherNumber;
    pthread_mutex_lock(&chopsticks[left]);
    printf("Philosopher %d pick up the chopstick %d\n", philosopherNumber, left);
    pthread_mutex_lock(&chopsticks[right]);
    printf("Philosopher %d pick up the chopstick %d\n", philosopherNumber, right);
}
```

```c
void putDown(int philosopherNumber) {
    // release chopsticks
    /*Your code here*/
    int left = (philosopherNumber + 1) % 5;
    int right = philosopherNumber;
    pthread_mutex_unlock(&chopsticks[left]);
    printf("Philosopher %d put down the chopstick %d\n", philosopherNumber, left);
    pthread_mutex_unlock(&chopsticks[right]);
    printf("Philosopher %d put down the chopstick %d\n", philosopherNumber, right);
}
```

将每位哲学家的行动写成相同格式：思考-请求开始就餐-请求筷子-就餐-释放筷子-释放就餐资源。

```c
void *philosopher(void *philosopherNumber) {
    while (1) {
        /*Your code here*/
        think(philosopherNumber);
        sem_wait(&Limit);
        pickUp(philosopherNumber);
        eat(philosopherNumber);
        putDown(philosopherNumber);
        sem_post(&Limit);
    }
}
```

我们这样就实现了简单的以限制同时试图就餐的人数来达成的哲学家就餐问题。