复旦理发师问题 实验报告

陈家豪 19307130210

回答问题

Question 1：

典型的生产者-消费者模型

互斥：

1. 等待椅子，各顾客线程之间是互斥关系。
2. 理发椅子，各顾客线程之间是互斥关系。

同步：

1. 理发师和顾客是同步关系，理发师理发时顾客可以同时进入理发店

Question 2:

1. 顾客进程：
    进程开始，获取锁mutex以查看空余等待椅子数量waiting
        若有空位，进入理发店空位
            将waiting减1，并释放锁mutex
            sem_post(&customers)，增加店内顾客数，唤醒理发师
            请求获取坐上理发椅子的锁
                若获取，则将waiting -1，等待椅子多出1个
                sem_post(&mutex2)， 表示坐上理发椅子，让理发师可以开始理发
        若无空位，则离开，并释放mutex
2. 理发师进程：
        进程开始，进入循环：
            sem_wait(&customers)，等待客人来，否则就休息
            若有客人来，下一步是sem_wait(&mutex2)，等客人坐上理发椅
            理发，等待一段时间
            获取锁mutex，看是否有等待的顾客
                若没有，则理发师可以休息
                若有，则不休息
            释放锁mutex，
            释放锁barbers，下一位等待的客人可以来理发椅

代码解释

用到的4个sem_t变量：

```
sem_init(&customers,0,0);    //店内顾客数量初值0，影响理发师是否工作
sem_init(&barbers,0,1);      //表示理发椅是否空，初值1，可被占用
sem_init(&mutex,0,1);        //对waiting的锁
sem_init(&mutex2,0,0);       //理发师需等顾客坐上椅子才开始理发的锁
```

自定义函数，打印当前时间

```
void getDateTime()
```

宏定义椅子数量

```
#define N 4     //等待区椅子数量
```

主线程部分：

开启barber线程，理发师开始等待；

每次隔随机a秒开启一个customer线程，表示隔了a秒来了一个客人

```
    pthread_create(&p2,NULL,(void *)barber,NULL);   //开启barber线程
    int i=0;
    int cus_num =8;    //顾客数量

    srand((unsigned)time(NULL));
    int a;
    for(i=0;i<cus_num;i++){
        a=rand()%6+1;       //1~6
        sleep(a);
        pthread_create(&(p1[i]),NULL,(void *)customer,(void *)&i);
    }


    pthread_join(p2,NULL);
    for(i=0;i<cus_num;i++){

        pthread_join(p1[i],NULL);
    }
```

barber线程：

```
void barber(){
    getDateTime();
    printf("barber线程启动:\n");
    getDateTime();
    printf("理发师在睡觉\n");
    while(1){           //循环
```

```
        sem_wait(&customers);//等待有客人来才开始工作，customers初值为0，来一个客人+1

        sem_wait(&mutex2);  //等客人坐上理发椅才开始理发，需加锁，不然会出现先理发,客人再坐上理发椅
        getDateTime();
        printf("为顾客%d理发\n",now);
        sleep(5);      //理发需要的时间（s）

        sem_wait(&mutex);//检查waiting，是否还有等待的客人
        if(waiting==0){ //没有在等待的客人，理发师就可以休息了
            getDateTime();
            printf("无等待顾客，理发师睡觉\n");
        }
        sem_post(&mutex);

        sem_post(&barbers); //客人离开,理发椅空出，mutex2解锁
        getDateTime();
        printf("顾客%d离开\n",now);
    }
    return NULL;
}
```

最后有一个细节：

理发师先判断没有等待的人了，自己可以休息，再让客人走

如果反过来，有可能出现：客人先走，等待位的客人坐上理发椅，理发师再检查等待位发现没人，休息，则会错过这个客人。

customer线程：

```
void customer(void *arg){
    int num = *((int*)arg);      //传入参数为客人编号
    sem_wait(&mutex);        //获取waiting的锁
    if(waiting < N){    //有空位，可以进入
        getDateTime();
        printf("顾客%d进入理发店\n",num);
        waiting = waiting + 1;  //等待的人数增加
        sem_post(&mutex);
        sem_post(&customers);   //顾客数增加，第一个客人可以唤醒理发师

        sem_wait(&barbers);     //客人等待坐上理发椅子

        sem_wait(&mutex);
        waiting = waiting -1;   //客人坐上理发椅,等待列表-1
        sem_post(&mutex);
        getDateTime();
        printf("顾客%d坐上理发椅\n",num);

        now = num;//现在在理发的是几号客人
        sem_post(&mutex2);      //表示客人坐上位置了，理发师可以开始理发
    }
    else{       //没有空位，就离开
        getDateTime();
        printf("由于人满，顾客%d离开\n",num);
```

```
        sem_post(&mutex);

    }
    return NULL;
}
```

结果

设理发店有4个等待椅子

以下情况为平常情况：

理发师5秒理完一个人，有10个顾客依次来，间隔时间为1～8s

```
jiahao@jiahao-virtual-machine:~/Documents/OS-lab/lab3$ ./test.out
19:51:11        main线程启动:
19:51:11        barber线程启动:
19:51:11        理发师在睡觉
19:51:12        顾客0进入理发店
19:51:12        顾客0坐上理发椅
19:51:12        为顾客0理发
19:51:16        顾客1进入理发店
19:51:17        顾客0离开
19:51:17        顾客1坐上理发椅
19:51:17        为顾客1理发
19:51:18        顾客2进入理发店
19:51:22        顾客1离开
19:51:22        顾客2坐上理发椅
19:51:22        为顾客2理发
19:51:23        顾客3进入理发店
19:51:27        顾客2离开
19:51:27        顾客3坐上理发椅
19:51:27        为顾客3理发
19:51:30        顾客4进入理发店
19:51:32        顾客3离开
19:51:32        顾客4坐上理发椅
19:51:32        为顾客4理发
19:51:36        顾客5进入理发店
19:51:37        顾客4离开
19:51:37        顾客5坐上理发椅
19:51:37        为顾客5理发
19:51:40        顾客6进入理发店
19:51:42        顾客5离开
19:51:42        顾客6坐上理发椅
19:51:42        为顾客6理发
19:51:44        顾客7进入理发店
19:51:47        顾客6离开
19:51:47        顾客7坐上理发椅
19:51:47        为顾客7理发
19:51:51        顾客8进入理发店
19:51:52        顾客7离开
19:51:52        顾客8坐上理发椅
19:51:52        为顾客8理发
19:51:57        无等待顾客，理发师睡觉
```

```
19:51:57        顾客8离开
19:51:58        顾客9进入理发店
19:51:58        顾客9坐上理发椅
19:51:58        为顾客9理发
19:52: 3        无等待顾客，理发师睡觉
19:52: 3        顾客9离开
```

以下为极端情况，测试同步互斥：

理发师5s处理一个人，8个人同时来：

处理方法为：去掉sleep(),同时开启8个客人线程

```
    for(i=0;i<cus_num;i++){
        //a=rand()%8+1;       //1~8
        //sleep(a);
        pthread_create(&(p1[i]),NULL,customer,(void *)i);
    }
```

结果：

```
jiahao@jiahao-virtual-machine:~/Documents/OS-lab/lab3$ ./test.out
19:58:55        main线程启动:
19:58:55        barber线程启动:
19:58:55        理发师在睡觉
19:58:55        顾客0进入理发店
19:58:55        顾客0坐上理发椅
19:58:55        为顾客0理发
19:58:55        顾客1进入理发店
19:58:55        顾客5进入理发店
19:58:55        顾客3进入理发店
19:58:55        顾客2进入理发店
19:58:55        由于人满，顾客4离开
19:58:55        由于人满，顾客7离开
19:58:55        由于人满，顾客6离开
19:59: 0        顾客0离开
19:59: 0        顾客1坐上理发椅
19:59: 0        为顾客1理发
19:59: 5        顾客1离开
19:59: 5        顾客5坐上理发椅
19:59: 5        为顾客5理发
19:59:10        顾客5离开
19:59:10        顾客3坐上理发椅
19:59:10        为顾客3理发
19:59:15        顾客3离开
19:59:15        顾客2坐上理发椅
19:59:15        为顾客2理发
19:59:20        无等待顾客，理发师睡觉
19:59:20        顾客2离开
```

顾客自由争取空位，顾客0最先坐上理发椅，1，5，3，2坐上等待椅，其余人没有空位，所以离开。