

山东大学 计算机科学与技术 学院

操作系统 课程实验报告

学号：202100130022	姓名：郭家宁	班级：21 数据
实验题目：进程互斥		
实验学时：2	实验日期：2023-5-15	
<p>实验目的：进一步研究和实践操作系统中关于并发进程同步与互斥操作的一些经典问题的解法，加深对于非对称性互斥问题有关概念的理解。</p> <p>观察和体验非对称性互斥问题的并发控制方法。进一步了解 Linux 系统中 IPC 进程同步工具的用法，训练解决对该类问题的实际编程、调试和分析问题的能力。</p>		

2. 独立实验

睡眠理发师问题：假设理发店的理发室中有 3 个理发椅子和 3 个理发师，有一个可容纳 4 个顾客坐等理发的沙发。此外还有一间等候室，可容纳 13 位顾客等候进入理发室。顾客如果发现理发店中顾客已满（超过 20 人），就不进入理发店。在理发店内，理发师一旦有空就为坐在沙发上等待时间最长的顾客理发，同时空的沙发让在等候室中等待时间最长的顾客就坐。顾客理完发后，可向任何一位理发师付款。但理发店只有一本现金登记册，在任一时刻只能记录一个顾客的付款。理发师在没有顾客的时候就坐在理发椅子上睡眠。理发师的时间就用在理发、收款、睡眠上。

请利用 linux 系统提供的 IPC 进程通信机制实验并实现理发店问题的一个解法。

(1) 首先建立两个消息队列，等候队列和沙发队列，然后设置两个信号灯，一个顾客的信号灯和账本的信号灯。

```
//联系一个请求消息队列
wait_quest_flg = IPC_CREAT | 0644;
wait_quest_key = 101;
wait_quest_id = set_msq(wait_quest_key, wait_quest_flg);
//联系一个响应消息队列
wait_respond_flg = IPC_CREAT | 0644;
wait_respond_key = 102;
wait_respond_id = set_msq(wait_respond_key, wait_respond_flg);

//联系一个请求消息队列
sofa_quest_flg = IPC_CREAT | 0644;
sofa_quest_key = 201;
sofa_quest_id = set_msq(sofa_quest_key, sofa_quest_flg);
//联系一个响应消息队列
sofa_respond_flg = IPC_CREAT | 0644;
sofa_respond_key = 202;
sofa_respond_id = set_msq(sofa_respond_key, sofa_respond_flg);

//信号量使用的变量
customer_key = 301; //顾客同步信号灯键值
account_key = 302; //账簿互斥信号灯键值
sem_flg = IPC_CREAT | 0644;
//顾客同步信号灯初值设为0
sem_val = 0;
//获取顾客同步信号灯,引用标识符 customer_sem
customer_sem = set_sem(customer_key, sem_val, sem_flg);
//账簿互斥信号灯初值设为 1
sem_val = 1;
//获取消费者同步信号灯,引用标识符 cons_sem
account_sem = set_sem(account_key, sem_val, sem_flg);
```

(2) 对于理发师实现：

① 使用一个父进程建立三个子进程然后当做三个理发师，然后如果接受到了沙发队列的信号就执行理发并且发送沙发使用完的信息，然后记账，由于只有一个账本，所以使用账本时候阻塞其他理发师使用账本。

```

for (int i = 0; i < 3; i++)
{
    sleep(2);
    if (fork() == 0)
    {
        while (1)
        {
            wait_quest_flg = IPC_NOWAIT;
            sleep(rate);

            if (msgrcv(sofa_quest_id, &msg_arg, sizeof(msg_arg) - sizeof(long), 0, wait_quest_flg) >= 0)
            {
                // printf("temp %d\n",temp);
                msg_arg.mtype = 1;

                msgsnd(sofa_respond_id, &msg_arg, sizeof(msg_arg) - sizeof(long), 0); // 发送服务的信号
                printf("%d 号理发师正在服务 %d 号顾客 \n", (int)getpid(), msg_arg.mid);

                down(account_sem);
                printf("%d 号理发师正在登记 %d 号顾客的钱\n", (int)getpid(), msg_arg.mid);
                up(account_sem);
            }
            else
            {
                printf("%d 号理发师正在睡觉\n", (int)getpid());
            }
        }
    }
}

```

(3) 对于顾客的实现

① 如果沙发的使用数量小于 4，判断等候室的顾客是否为 0，如果是 0，直接进入沙发，如果不为 0，那么给等候室队列发送 finish 的信号，然后顾客从等候室进入了沙发。然后给沙发队列发送请求信号。

```

if (usedSofa < 4)
{
    if (waitedNum != 0)
    {
        i--;

        // up(costomer_sem);

        msgrcv(wait_quest_id, &msg_arg, sizeof(msg_arg) - sizeof(long), 0, 0);

        msg_arg.mtype = 1;
        int sed = msgsnd(wait_respond_id, &msg_arg, sizeof(msg_arg) - sizeof(long), 0);

        printf("%d 号顾客从等候室来到了沙发\n", msg_arg.mid);
    }
    else
    {
        printf("%d 号新顾客直接坐到了沙发上\n", i);
    }

    sofa_quest_flg = IPC_NOWAIT; // 非阻塞方式接收消息
    msg_arg.mtype = 1;

    // 成功发送坐沙发的请求
    if (mgsnd(sofa_quest_id, &msg_arg, sizeof(msg_arg) - sizeof(long), sofa_quest_flg) >= 0)
    {
        usedSofa++;
    }
    else
    {
        printf("msgsnd fail\n");
    }
}

```

② 如果沙发已满，而且等候室不满，那么就给等候队列发送等候请求，如果等候室已满，那么阻塞顾客，等候沙发队列接收到 finish 的信号。

```

else if (waitedNum < 13)
{
    printf("沙发已经坐满了 %d 号顾客正在等候室等待\n", i);
    wait_quest_flg = IPC_NOWAIT; //非阻塞方式接收消息
    msg_arg.mtype = 1;
    // 发送进入等候室的请求
    msgsnd(wait_quest_id, &msg_arg, sizeof(msg_arg) - sizeof(long), wait_quest_flg);
    waitedNum++;
}
else
{
    printf("等候室已满 %d 号顾客不能进入理发店\n", i);
    down(customer_sem);

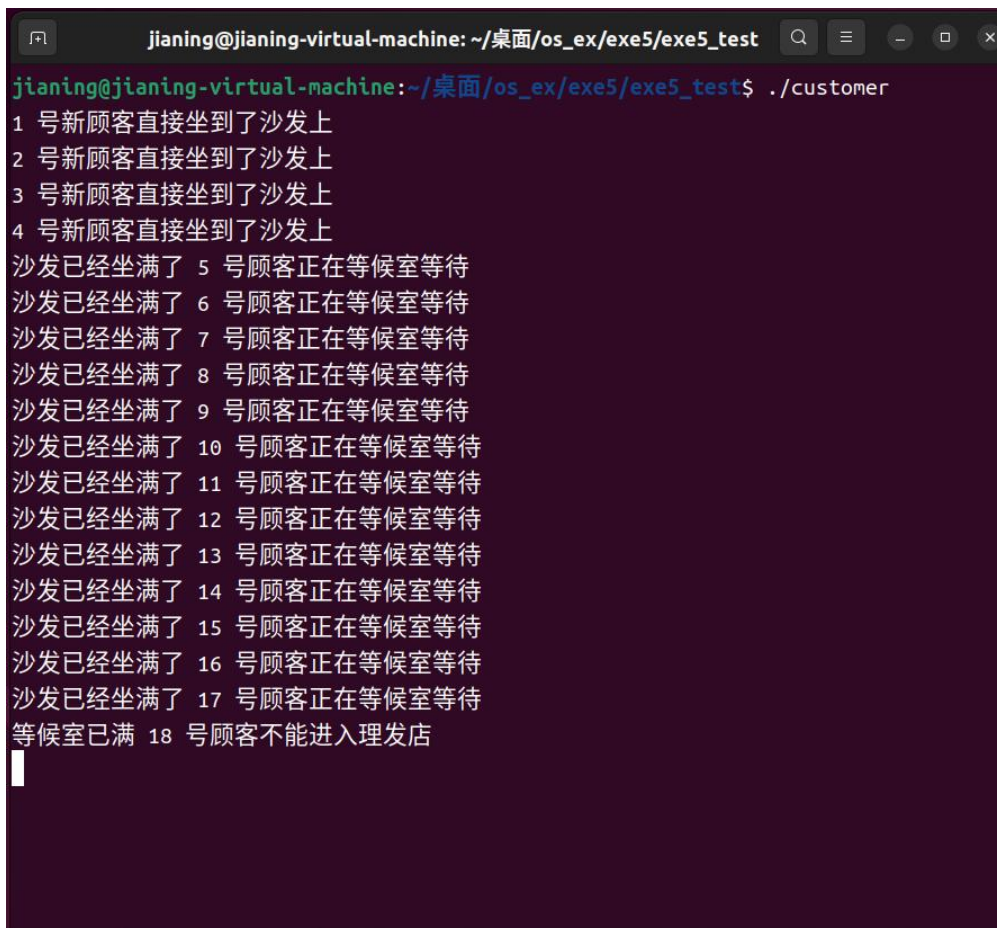
    // 接收到沙发使用结束的信息
    msgrcv(sofa_respond_id, &msg_arg, sizeof(msg_arg) - sizeof(long), 0, 0);
    usedSofa--;
    i--;
}
}

```

③ 最后收尾处理接收其他的沙发信号和等候室送来的信号

(4) 实验结果

① 首先建立顾客进程，可以发现在 18 号顾客后已经不能有顾客进入理发店。



```

jianing@jianing-virtual-machine: ~/桌面/os_ex/exe5/exe5_test
jianing@jianing-virtual-machine:~/桌面/os_ex/exe5/exe5_test$ ./customer
1 号新顾客直接坐到了沙发上
2 号新顾客直接坐到了沙发上
3 号新顾客直接坐到了沙发上
4 号新顾客直接坐到了沙发上
沙发已经坐满了 5 号顾客正在等候室等待
沙发已经坐满了 6 号顾客正在等候室等待
沙发已经坐满了 7 号顾客正在等候室等待
沙发已经坐满了 8 号顾客正在等候室等待
沙发已经坐满了 9 号顾客正在等候室等待
沙发已经坐满了 10 号顾客正在等候室等待
沙发已经坐满了 11 号顾客正在等候室等待
沙发已经坐满了 12 号顾客正在等候室等待
沙发已经坐满了 13 号顾客正在等候室等待
沙发已经坐满了 14 号顾客正在等候室等待
沙发已经坐满了 15 号顾客正在等候室等待
沙发已经坐满了 16 号顾客正在等候室等待
沙发已经坐满了 17 号顾客正在等候室等待
等候室已满 18 号顾客不能进入理发店

```

② 开启理发师进程，可以发现在理发师进程开启后，顾客不再阻塞。


```
jianing@jianing-virtual-machine: ~/桌面/os_exe5/exe5_test
3 号新顾客直接坐到了沙发上
4 号新顾客直接坐到了沙发上
沙发已经坐满了 5 号顾客正在等候室等待
沙发已经坐满了 6 号顾客正在等候室等待
沙发已经坐满了 7 号顾客正在等候室等待
沙发已经坐满了 8 号顾客正在等候室等待
沙发已经坐满了 9 号顾客正在等候室等待
沙发已经坐满了 10 号顾客正在等候室等待
沙发已经坐满了 11 号顾客正在等候室等待
沙发已经坐满了 12 号顾客正在等候室等待
沙发已经坐满了 13 号顾客正在等候室等待
沙发已经坐满了 14 号顾客正在等候室等待
沙发已经坐满了 15 号顾客正在等候室等待
沙发已经坐满了 16 号顾客正在等候室等待
沙发已经坐满了 17 号顾客正在等候室等待
等候室已满 18 号顾客不能进入理发店
5 号顾客从等候室来到了沙发
沙发已经坐满了 18 号顾客正在等候室等待
6 号顾客从等候室来到了沙发
7 号顾客从等候室来到了沙发
8 号顾客从等候室来到了沙发
9 号顾客从等候室来到了沙发
10 号顾客从等候室来到了沙发

jianing@jianing-virtual-machine: ~/桌面/os_exe5/exe5_test$ ./barber
2668 号理发师正在服务 1 号顾客
2668 号理发师正在登记 1 号顾客的钱
jianing@jianing-virtual-machine: ~/桌面/os_exe5/exe5_test$ 2669 号理发师正在服
务 2 号顾客
2669 号理发师正在登记 2 号顾客的钱
2668 号理发师正在服务 3 号顾客
2668 号理发师正在登记 3 号顾客的钱
2670 号理发师正在服务 4 号顾客
2670 号理发师正在登记 4 号顾客的钱
2669 号理发师正在服务 5 号顾客
2669 号理发师正在登记 5 号顾客的钱
2668 号理发师正在服务 6 号顾客
2668 号理发师正在登记 6 号顾客的钱
2670 号理发师正在服务 7 号顾客
2670 号理发师正在登记 7 号顾客的钱
```

③ 阻塞消费者进程后，可以发现开始理发师睡觉

```
jianing@jianing-virtual-machine: ~/桌面/os_exe5/exe5_test
14 号顾客从等候室来到了沙发
15 号顾客从等候室来到了沙发
16 号顾客从等候室来到了沙发
17 号顾客从等候室来到了沙发
18 号顾客从等候室来到了沙发
19 号新顾客直接坐到了沙发上
20 号新顾客直接坐到了沙发上
21 号新顾客直接坐到了沙发上
22 号新顾客直接坐到了沙发上
23 号新顾客直接坐到了沙发上
24 号新顾客直接坐到了沙发上
25 号新顾客直接坐到了沙发上
26 号新顾客直接坐到了沙发上
27 号新顾客直接坐到了沙发上
28 号新顾客直接坐到了沙发上
29 号新顾客直接坐到了沙发上
30 号新顾客直接坐到了沙发上
31 号新顾客直接坐到了沙发上
32 号新顾客直接坐到了沙发上
33 号新顾客直接坐到了沙发上
34 号新顾客直接坐到了沙发上
35 号新顾客直接坐到了沙发上
^C

2668 号理发师正在登记 27 号顾客的钱
2670 号理发师正在服务 28 号顾客
2670 号理发师正在登记 28 号顾客的钱
2669 号理发师正在服务 29 号顾客
2669 号理发师正在登记 29 号顾客的钱
2668 号理发师正在服务 30 号顾客
2668 号理发师正在登记 30 号顾客的钱
2670 号理发师正在服务 31 号顾客
2670 号理发师正在登记 31 号顾客的钱
2669 号理发师正在服务 32 号顾客
2669 号理发师正在登记 32 号顾客的钱
2668 号理发师正在服务 33 号顾客
2668 号理发师正在登记 33 号顾客的钱
2670 号理发师正在服务 34 号顾客
2670 号理发师正在登记 34 号顾客的钱
2669 号理发师正在服务 35 号顾客
2669 号理发师正在登记 35 号顾客的钱
2668 号理发师正在睡觉
2670 号理发师正在睡觉
2669 号理发师正在睡觉
2668 号理发师正在睡觉
2670 号理发师正在睡觉
2669 号理发师正在睡觉
```

3. 回答问题

(1) 真实操作系统中提供的并发进程同步机制是 怎样实现和解决同步问题的，它们是怎样应用操作系统教材中讲解的进程同步原理的？

操作系统中提供的并发进程同步机制的实现通常依赖于硬件和内核的支持。这些机制包括二进制信号量、计数信号量、事件标志和互斥量等。

在具体实现上，当一个进程需要持有资源或执行某个关键代码段时，它需要主动请求获得对该资源的访问权限。此时，如果该资源已经被其他进程占据，则该请求将进入阻塞状态，直到该资源得到释放并且控制权获得转交。同时，为了保证不会出现无限期等待可能对整个系统造成灾难性后果的情况，在等待一定时间后，对于长时间未获得访问许可的请求将被强制终止。

(2) 对应教材中信号量的定义，说明信号量机制是怎样完成进程的互斥和同步的？其中信号量的初值和其值的变化物理意义是什么？

信号量是一种同步原语，可用于进程之间或线程之间进行同步和通信，以确保共享资源的安全和正确使用。其中 P（passeren）和 V（verhogen）是两种基本的操作。它们通过原子性地增加或减少一个计数器的值来保持进程的互斥和同步。

对于二元信号量（也称为互斥锁），它只能取 0 或 1 两种值，表示临界区域中是否有进程运行。当一个进程进入该区域时，它将 P 操作减少信号量并获得控制权；离开前则通过 V 操作放弃控制权并递增信号量。

对于计数信号量，其值可以大于 1，表示可用共享资源的数量。当一个进程需要使用共享资源时，它将 P 操作减少信号量，并在获得访问权限后执行必要的操作；完成后，通过 V 操作将信号量递增以释放对该共享资源的独占访问。

信号量创建时具有一个初始值，该值指示可用资源的数量。当进程完成其作业并退出时，如果它已经访问了该资源，则将使信号量值递增，从而将资源返回到资源池中。在此过程中，信号量的物理意义是控制共享资源的数量或访问许可的数量，以防止特定资源被访问太多而影响系统性能。

问题及收获：

对进程互斥有了更加深刻的理解与体会。