

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：进程同步		
实验学时：2	实验日期：2023-5-8	
<p>实验目的：</p> <p>加深对并发协作进程同步与互斥概念的理解，观察和体验并发进程同步与互斥操作的效果，分析与研究经典进程同步与互斥问题的实际解决方案。了解 Linux 系统中 IPC 进程同步工具的用法，练习并发协作进程的同步与互斥操作的编程与调试技术。</p>		

实验结果：

1. 对于样例实验, 生产者 1 在缓冲区满后, 停止, 生产者 2 在缓冲区满后, 停止生产。

```
gcc producer.o ipc.o -o producer
gcc consumer.o ipc.o -o consumer
jianing@jianing-virtual-machine:~/桌面/os_ex/exe4/exe4_example$ ./producer
3105 producer put: A to Buffer[0]
3105 producer put: B to Buffer[1]
3105 producer put: C to Buffer[2]
3105 producer put: D to Buffer[3]
3105 producer put: E to Buffer[4]
3105 producer put: F to Buffer[5]
3105 producer put: G to Buffer[6]
3105 producer put: H to Buffer[7]
```

2. 对于两个消费者, 使用不同的时间间隔, 一个只读取偶数缓冲区, 另一个只读奇数缓冲区。

```
jianing@jianing-virtual-machine:~/桌面/os_ex/exe4/exe4_example$ ./consumer 6
3125 consumer get: E from Buffer[4]
3125 consumer get: G from Buffer[6]
3125 consumer get: A from Buffer[0]
[...]
```

以生产者不断放产品

```
jianing@jianing-virtual-machine:~/桌面/os_ex/exe4/exe4_example$ ./consumer 3
3117 consumer get: A from Buffer[0]
3117 consumer get: B from Buffer[1]
3117 consumer get: C from Buffer[2]
3117 consumer get: D from Buffer[3]
3117 consumer get: F from Buffer[5]
3117 consumer get: H from Buffer[7]
```

3. 最后总的结果：

```
jianing@jianing-virtual-machine: ~/桌面/os_exe4/exe4_example
3105 producer put: E to Buffer[4]
3105 producer put: F to Buffer[5]
3105 producer put: G to Buffer[6]
3105 producer put: H to Buffer[7]
3105 producer put: A to Buffer[0]
3105 producer put: B to Buffer[1]
3105 producer put: C to Buffer[2]
3105 producer put: D to Buffer[3]
3105 producer put: E to Buffer[4]
3105 producer put: F to Buffer[5]
3105 producer put: G to Buffer[6]
3105 producer put: H to Buffer[7]
3105 producer put: A to Buffer[0]
3105 producer put: B to Buffer[1]
3105 producer put: C to Buffer[2]
3105 producer put: D to Buffer[3]
3105 producer put: E to Buffer[4]
3105 producer put: F to Buffer[5]
3105 producer put: G to Buffer[6]
3105 producer put: H to Buffer[7]
3105 producer put: A to Buffer[0]
3105 producer put: B to Buffer[1]
3105 producer put: C to Buffer[2]

jianing@jianing-virtual-machine: ~/桌面/os_exe4/exe4_example
3125 consumer get: E from Buffer[4]
3125 consumer get: G from Buffer[6]
3125 consumer get: A from Buffer[0]
3125 consumer get: C from Buffer[2]
3125 consumer get: G from Buffer[6]
3125 consumer get: A from Buffer[0]
3125 consumer get: E from Buffer[4]
3125 consumer get: C from Buffer[2]
3125 consumer get: G from Buffer[6]
3125 consumer get: A from Buffer[0]
3125 consumer get: E from Buffer[4]
3125 consumer get: C from Buffer[2]
3125 consumer get: G from Buffer[6]
3125 consumer get: A from Buffer[0]

jianing@jianing-virtual-machine: ~/桌面/os_exe4/exe4_example
3117 consumer get: A from Buffer[0]
3117 consumer get: B from Buffer[1]
3117 consumer get: C from Buffer[2]
3117 consumer get: D from Buffer[3]
3117 consumer get: F from Buffer[5]
3117 consumer get: H from Buffer[7]
3117 consumer get: B from Buffer[1]
3117 consumer get: D from Buffer[3]
3117 consumer get: F from Buffer[5]
3117 consumer get: H from Buffer[7]
3117 consumer get: B from Buffer[1]
3117 consumer get: D from Buffer[3]
3117 consumer get: F from Buffer[5]
3117 consumer get: H from Buffer[7]
```

独立实验：

抽烟者问题。假设一个系统中有三个抽烟者进程，每个抽烟者不断地卷烟并抽烟。抽烟者卷起并抽掉一颗烟需要有三种材料：烟草、纸和胶水。一个抽烟者有烟草，一个有纸，另一个有胶水。系统中还有两个供应者进程，它们无限地供应所有三种材料，但每次仅轮流提供三种材料中的两种。得到缺失的两种材料的抽烟者在卷起并抽掉一颗烟后会发信号通知供应者，让它继续提供另外的两种材料。这一过程重复进行。请用以上介绍的 IPC 同步机制编程，实现该问题要求的功能。

1. 给生产者设立的缓冲区间长度为 3。

```
buff_key = 101; // 共享内存使用的变量
buff_num = 3; // 缓冲区任给的键值
pput_key = 102; // 缓冲区任给的长度
pput_num = 1; // 生产者放产品指针的
shm_flg = IPC_CREAT | 0644; // 指针数
// 共享内存读写权限
```

2. 建立三个控制键值，和两个生产者的同步键值

```
ab_key = 201;        //有C的消费者控制键值
ac_key = 202;        //有B的消费者控制键值
bc_key = 203;        //有A的消费者控制键值
all_key = 204;       //对一个缓冲区的控制键值
produce_key = 205;    //对两个生产者的同步键值
sem_flg = IPC_CREAT | 0644;
```

使用一个值对三取余然后循环给缓冲区写内容而且通知三个不同的消费者控制键值。

```
43
44     while (1)
45     {
46         int d = i % 3; // 标记传输的smoker
47         i++;
48         down(all_int); // 如果缓冲区满，则生产者阻塞
49         down(produce_int); // 如果另一个生产者在放置产品，则本生产者阻塞
50         //用写一字符的形式模拟生产者放产品，报告本进程号和放入的字符及存放的位置
51         buff_ptr[*pput_ptr] = 'A' + d;
52         sleep(rate);
53         if (d == 0)
54         {
55             printf("%d producer put: Tobacco and paper to Buffer [%d]\n", getpid(), *pput_ptr);
56         }
57         else
58         {
59             if (d == 1)
60             {
61                 printf("%d producer put: glue and paper to Buffer [%d]\n", getpid(), *pput_ptr);
62             }
63             else
64             {
65                 printf("%d producer put: Tobacco and glue to Buffer [%d]\n", getpid(), *pput_ptr);
66             }
67         }
68     }
69     *pput_ptr = (*pput_ptr + 1) % buff_num;
70
71     up(produce_int); // 唤醒阻塞的生产者
72
73     // 唤醒阻塞的消费者
74     if (d == 0)
75     {
76         up(bc_int);
77     }
78     else
79     {
80         if (d == 1)
81         {
82             up(ac_int);
83         }
84         else
85         {
86             up(ab_int);
87         }
88     }
89 }
90 return EXIT_SUCCESS;
91 }
```

3. 执行代码，首先建立生产者 1，在往缓冲区写满后阻塞。

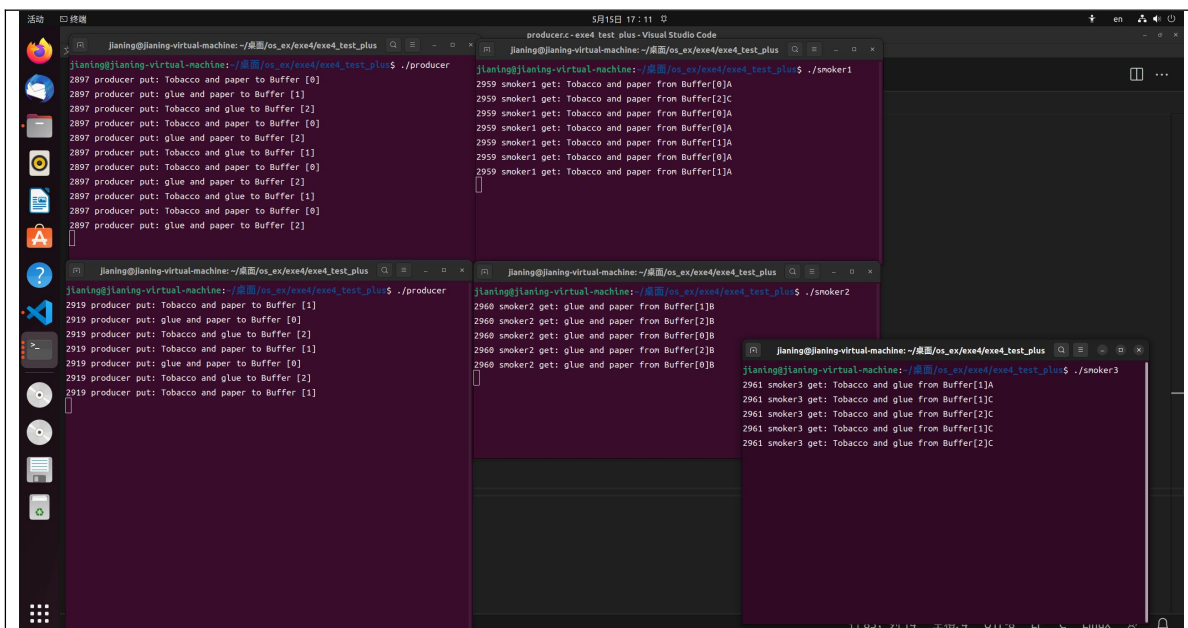
```
jianing@jianing-virtual-machine: ~/桌面/os_ex/exe4/exe4_test_plus
jianing@jianing-virtual-machine:~/桌面/os_ex/exe4/exe4_test_plus$ ./producer
2897 producer put: Tobacco and paper to Buffer [0]
2897 producer put: glue and paper to Buffer [1]
2897 producer put: Tobacco and glue to Buffer [2]
```

建立生产者 2，由于缓冲区已满，直接堵塞。

```
jianing@jianing-virtual-machine: ~/桌面/os_ex/exe4/exe4_test_plus
jianing@jianing-virtual-machine:~/桌面/os_ex/exe4/exe4_test_plus$ ./producer
2897 producer put: Tobacco and paper to Buffer [0]
2897 producer put: glue and paper to Buffer [1]
2897 producer put: Tobacco and glue to Buffer [2]
█

jianing@jianing-virtual-machine: ~/桌面/os_ex/exe4/exe4_test_plus
jianing@jianing-virtual-machine:~/桌面/os_ex/exe4/exe4_test_plus$ ./producer
█
```

4. 对三个吸烟者执行代码，然后观察实验结果



```
jianing@jianing-virtual-machine: ~/桌面/os_ex/exe4/exe4_test_plus$ ./producer
2897 producer put: Tobacco and paper to Buffer [0]
2897 producer put: glue and paper to Buffer [1]
2897 producer put: Tobacco and glue to Buffer [2]
2897 producer put: Tobacco and paper to Buffer [0]
2897 producer put: glue and paper to Buffer [2]
2897 producer put: Tobacco and glue to Buffer [1]
2897 producer put: Tobacco and paper to Buffer [0]
2897 producer put: glue and paper to Buffer [2]
2897 producer put: Tobacco and glue to Buffer [1]
2897 producer put: Tobacco and paper to Buffer [0]
2897 producer put: glue and paper to Buffer [2]

jianing@jianing-virtual-machine: ~/桌面/os_ex/exe4/exe4_test_plus$ ./producer
2919 producer put: Tobacco and paper to Buffer [1]
2919 producer put: glue and paper to Buffer [0]
2919 producer put: Tobacco and paper to Buffer [1]
2919 producer put: glue and paper to Buffer [0]
2919 producer put: Tobacco and glue to Buffer [2]
2919 producer put: Tobacco and paper to Buffer [1]

jianing@jianing-virtual-machine: ~/桌面/os_ex/exe4/exe4_test_plus$ ./snoker1
2959 snoker1 get: Tobacco and paper from Buffer[0]A
2959 snoker1 get: Tobacco and paper from Buffer[2]C
2959 snoker1 get: Tobacco and paper from Buffer[0]A
2959 snoker1 get: Tobacco and paper from Buffer[1]A
2959 snoker1 get: Tobacco and paper from Buffer[0]A
2959 snoker1 get: Tobacco and paper from Buffer[1]A

jianing@jianing-virtual-machine: ~/桌面/os_ex/exe4/exe4_test_plus$ ./snoker2
2960 snoker2 get: glue and paper from Buffer[1]B
2960 snoker2 get: glue and paper from Buffer[2]B
2960 snoker2 get: glue and paper from Buffer[0]B
2960 snoker2 get: glue and paper from Buffer[2]B

jianing@jianing-virtual-machine: ~/桌面/os_ex/exe4/exe4_test_plus$ ./snoker3
2961 snoker3 get: Tobacco and glue from Buffer[1]A
2961 snoker3 get: Tobacco and glue from Buffer[2]C
2961 snoker3 get: Tobacco and glue from Buffer[1]C
2961 snoker3 get: Tobacco and glue from Buffer[2]C
```

可以发现,两个生产者轮流生产,然后三个消费者轮流在对应的缓冲区里面获取其所需要的资源,一开始在只有一个生产者时候,消费者在缓冲区内获取的位置不一定,但是后面两个生产者轮流交替生产后,三个消费者在缓冲区内获取资源的位置稳定。

问题及收获：

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？

实际操作系统中提供的并发进程同步机制常用的有互斥锁、条件变量和信号量等。在解决同步问题时，这些机制的主要目的是协调不同进程之间的操作顺序，避免竞争状态和死锁的产生。

具体来说，以信号量为例，它是一种整数值，用于控制临界资源的访问。当临界资源被某一个进程占用时（信号量值为 0），其他进程需要等待，直到该进程释放临界资源（信号量值加 1）后才能进行访问。通过对信号量进行 P 操作（等待）和 V 操作（释放），可以实现进程之间的互斥和同步。

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？

信号量是一种基本的进程间通信机制，用于实现进程间的同步和互斥。通过定义信号量变量和对其进行操作（P 操作和 V 操作），可以确保多个进程之间的访问顺序和互斥性。

3. 信号量的初值和其值的变化物理意义是什么？

信号量的初值表示系统中可用资源的数量，当信号量被初始化为 1 时，它被称为二元信号量（Binary Semaphore），表示只有一个进程能够进入临界区。此外，信号量的值还随着不同操作的进行而改变，P 操作对

信号量减 1（申请资源），V 操作对信号量加 1（释放资源）。在实现进程的互斥和同步问题时，需要对信号量进行加锁和解锁操作，以及判断信号量当前的值来决定是否可以进入临界区或等待。根据信号量的值的变化情况，可以判断各个进程是否处于满足资源需求的状态下执行。

