操作系统第 4 次作业

张金源/76066001

1.读者写者问题（写者优先）: 1）共享读; 2）互斥写、读写互斥; 3）写者优先于读者（一 旦有写者，则后续读者必须等待，唤醒时优先考虑写者）。

1）共享读

/*初始化读者队列为 0，文件资源的初始值为 1*/

```
int readCount = 0;
semaphore readCountSignal = 1;

reader()
{
     while(true)
    {

       wait(readCountSignal); //申请读者队列计数器

       if(!readCount)        //如果读者队列为空，申请文件资源

          wait(fileSrc);

       readCount++;

       signal(readCountSignal); //释放读者计数器资源

       ...

       perform read operation //执行临界区代码
       ...


       wait(readCountSignal);   //申请读者计数器资源

       readCount--;

       if(!readCount)         //如果读者队列为空，释放文件资源

         signal(fileSrc);

       signal(readCountSignal); //释放读者计数器资源

     }
```

```
}

writer()
{
    while(true)
    {
     wait(file);              //申请文件资源

     ...

     perform write operation   //执行临界区代码
     ...

     signal(fileSrc);         //释放文件资源

    }
}
```

2）互斥写，读写互斥

```
/* 读者队列初始值为 0，其他资源初始值为 1*/
int readCount = 0;
semaphore keySignal = 1;
semaphore OneSignal = 1;
semaphore readCountSignal = 1;

reader()
{
   while(true)
   {
    wait(keySignal);        //申请令牌
    wait(readCountSignal);  //申请计数器资源
    if(!readCount)          //为零则申请文件资源
      wait(fileSrc);
    readCount++;
    signal(readCountSignal); //释放计数器资源
    signal(keySignale);      //释放令牌

    ...
    perform read operation  //执行临界区代码
    ...
```

操作系统第 4 次作业

张金源/76066001

```
     wait(readCountSignal);  //申请计数器资源
     readCount--;
     if(!readCount)               //为零则释放文件资源
        signal(fileSrc);
    signal(readCountSignal); //释放读者计数器资源
    }
}

writer()
{
    while(true)
    {
       wait(OneSignal);     //申请令牌资源
       wait(keySignal);     //申请令牌
       wait(fileSrc);       //申请文件资源

       ...
       perform write operation //执行临界区代码
       ...

       signal(fileSrc);   //释放文件资源
       signal(keysignal); //释放令牌
       signal(OneSignal); //释放令牌资源
    }
}
```

操作系统第 4 次作业

张金源/76066001

3）写者优先

```
/*初始化读者、写者队列为 0，初始化令牌资源、读写计数器资源的初始值为 1*/
int readCount = 0;
int writeCount = 0;
semaphore read = 1;
semaphore readCountSignal = 1;
semaphore writeCountSignal = 1;

reader()
{
     while(true)
    {
       wait(read);           //申请令牌
       wait(readCountSignal); //申请读者队列计数器

       if(!readCount)        //如果读者队列为空，申请文件资源
          wait(fileSrc);
       readCount++;

       signal(readCountSignal); //释放读者计数器资源
       signal(read);           //释放令牌

       ...
       perform read operation //执行临界区代码
       ...

       wait(readCountSignal);   //申请读者计数器资源
       readCount--;
       if(!readCount)         //如果读者队列为空，释放文件资源
         signal(fileSrc);
       signal(readCountSignal); //释放读者计数器资源
     }
}

writer()
{
    while(true)
    {
      wait(writeCountSignal);  //申请写者计数器资源
      if(!writeCount)         //如果写者队列为空则申请令牌
        wait(read);
```

```
    writeCount++;
    signal(writeCountSignal); //释放写者计数器资源
    wait(file);               //申请文件资源

    ...
    perform write operation   //执行临界区代码
    ...

    signal(fileSrc);          //释放文件资源
    wait(writeCountSignal);   //申请写者计数器资源
    writeCount--;
    if(!writeCount)           //如果写者队列为空则释放令牌
      signal(read);
    signal(writeCountSignal); //释放写者计数器资源
   }
}
```

2.寿司店问题。假设一个寿司店有 5 个座位，如果你到达的时候有一个空座位，你可以立 刻就坐。但是如果你到达的时候 5 个座位都是满的有人已经就坐，这就意味着这些人都是一 起来吃饭的，那么你需要等待所有的人一起离开才能就坐。编写同步原语，实现这个场景的 约束。

操作系统第 4 次作业

张金源/76066001

3.系统中有多个生产者进程和消费者进程，共享用一个可以存 1000 个产品的缓冲区（初始 为空），当缓冲区为未满时，生产者进程可以放入一件其生产的产品，否则等待；当缓冲区 为未空时，消费者进程可以取走一件产品，否则等待。要求一个消费者进程从缓冲区连续取 出 10 件产品后，其他消费者进程才可以取产品，请用信号量 P，V 操作实现进程间的互斥 和同步，要求写出完整的过程；并指出所用信号量的含义和初值。

Buffer array [1000];

（1）生产者之间设互斥信号量 mutex1， 消费者之间设互斥信号量 metex2。

（2）上述进程的同步问题，需设置 3 个信号量，其中 empty 对应空闲的缓冲单元，初值为 1000； full 对应缓冲区中待取走的产品数，初值为 0；另外，还需定义 2 个整型变量 in、 out，分别用来指示下一个可存放产品的缓冲单元、下一个取走的缓冲单元，它们的初值均为 0。

```
buffer array [1000]; //存放产品的缓冲区
buffer nextp; //用于临时存放生产者生产的产品
buffer nextc [10]; //用于临时存放消费者取出的产品
semaphore empty = 1000; //空缓冲区的数目
semaphore full = 0; //满缓冲区的数目
semaphore mutex1 = 1; //用于生产者之间的互斥
semaphore mutex2 = 1; //用于消费者之间的互斥
int in = 0; //指示生产者的存位置
int out = 0; //指示消费者的取位置
Producer() //生产者进程
{
     Produce an item put in nextp; //生产一个产品，存在临时缓冲区
     P(empty); //申请一个空缓冲区
     P(mutex1); //生产者申请使用缓冲区
     array[in]=nextp; //将产品存入缓冲区
     in = (in+1)%1000; //指针后移
     V(mutex1); //生产者缓冲区使用完毕，释放互斥信号量
     V(full);} //增加一个满缓冲区
}
Consumer() //消费者进程
{
     P(mutex2); //消费者申请使用缓冲区
     for(int i = 0;i<10;i++) //一个消费者进程需从缓冲区连续取走 10 件产品
     {
          P(full); //申请一个满缓冲区
          nextc[i] = array[out]; //将产品取出，存于临时缓冲区
          out = (out+1)%1000; //指针后移
```

操作系统第 4 次作业

张金源/76066001

```
            V(empty); //增加一个空缓冲区
        }
        V(mutex2); //消费者缓冲区使用完毕，释放互斥信号量
        Consume the items in nextc; //消费掉这 10 个产品
}
```

4.搜索-插入-删除问题。三个线程对一个单链表进行并发的访问，分别进行搜索、插入和删 除。搜索线程仅仅读取链表，因此多个搜索线程可以并发。插入线程把数据项插入到链表最 后的位置；多个插入线程必须互斥防止同时执行插入操作。但是，一个插入线程可以和多个 搜索线程并发执行。最后，删除线程可以从链表中任何一个位置删除数据。一次只能有一个 删除线程执行；删除线程之间，删除线程和搜索线程，删除线程和插入线程都不能同时执行。 请编写三类线程的同步互斥代码，描述这种三路的分类互斥问题。