華中科技大學

操 作 系 统

实 验 报 告

实验名称	阅读者和写入者问题
班　级	提高 2001 班
学　号	U202013830
姓　名	陈艺文
任课老师	刘澍

电子信息与通信学院

一、 问题描述

有一个公用的数据集，有很多人需要访问，其中一些需要阅读其中的信息，一些需要修改其中的消息。阅读者可以同时访问数据集，而写入者只能互斥的访问数据集，不能与任何的进程一起访问数据区。

二、 说明和主要代码

在程序编译运行后会出现中间一个大的圆圈表示公用的资源，上面一排五个矩形表示5个读者，下面的五个矩形表示五个写入者。每个读者和写入者都有3种状态，休息，等待和操作（读入或者写入）分别用黑颜色，绿颜色，红颜色表示休息，等待和操作。一旦操作者获得资源，可以进行读或者写，我们就划一条从操作者中心到资源中心的线，表示开始操作。

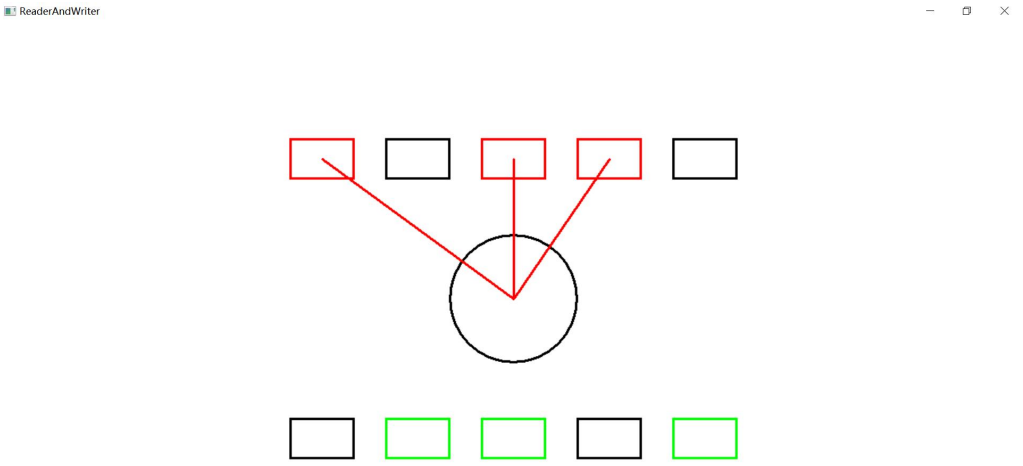

图 2-1 读写操作

ReaderThread()和WriterThread()线程主要代码如下图所示

```
DWORD WINAPI ReaderThread(LPVOID pVoid)
{   int ReaderNum = (int) pVoid;  // get the number of thread(reader)
    DWORD result;
    //Randomize the random number generator
  srand( (unsigned)time( NULL ) * (ReaderNum + 1) );
    readerstate[ReaderNum] = resting;        //reader is resting
    Sleep(P_DELAY);
  for(;;)
  {// Wait until resources are available
    readerstate[ReaderNum] = waiting;    //reader is waiting
    PostMessage(hWndMain, WM_FORCE_REPAINT,0 ,0);
    result=WaitForSingleObject(count, INFINITE);  //get the resource of count
    if(result==WAIT_OBJECT_0)
      Readercount+=1;
    if(Readercount==1)
      EnterCriticalSection(&RP_Write);  //get the critical section
    MTVERIFY(ReleaseMutex(count));        // release the resource of count
    resourcestate[ReaderNum] = read;
    readerstate[ReaderNum] = reading;     //the reader is reading
    PostMessage(hWndMain, WM_FORCE_REPAINT,0 ,0);
    Sleep(P_DELAY/4);

    result=WaitForSingleObject(count, INFINITE); //get the resource of count
    if(result==WAIT_OBJECT_0)
      Readercount-=1;
    if(Readercount==0)
      LeaveCriticalSection(&RP_Write);     //leave the cristical section
    MTVERIFY(ReleaseMutex(count));           //release the resource of count
    readerstate[ReaderNum] = resting;        //the reader is reading
    resourcestate[ReaderNum] = UNUSED;
    PostMessage(hWndMain, WM_FORCE_REPAINT,0 ,0);
        Sleep(P_DELAY);
  }
  return 0;
}
```

```
DWORD WINAPI WriterThread(LPVOID pVoid)
{  int writerNum = (int)pVoid;   // get the nunber of thread(writer)
  //Randomize the random number generator
  srand( (unsigned)time( NULL ) * (writerNum + 1) );
  writerstate[writerNum] = resting;  //the writer id resting
  Sleep(P_DELAY);
  for(;;)
  { // Wait until resources are available
    writerstate[writerNum] = waiting;   //the writer is waiting
    PostMessage(hWndMain, WM_FORCE_REPAINT,0 ,0);
      EnterCriticalSection(&RP_Write);   // enter the critical section
    writerstate[writerNum] = writing;  // the writer is writing
    resourcestate[writerNum] = write;
    PostMessage(hWndMain, WM_FORCE_REPAINT,0 ,0);
    Sleep(P_DELAY/4);
    LeaveCriticalSection(&RP_Write);   //leave the critical section
    writerstate[writerNum] = resting;  // the writer is resting
    resourcestate[writerNum] = UNUSED;
    PostMessage(hWndMain, WM_FORCE_REPAINT,0 ,0);
    Sleep(P_DELAY);
  }
  return 0;
}
```

三、 伪代码

1. **读优先策略**

```
Semaphore mutex = 1, wrt = 1;
Readcount : integer;
Readcount = 0;
Parbegin
    //读者reader 线程
    Readeri:begin
        resting;//休息状态
 Wait(mutex);
 readcount:=readcount+1;
        If readcount=1 then Wait(wrt);//如果是第一个读者，需要等待写操作释放
        Signal(mutex);
        reading; //进入 reading 状态
        Wait(mutex);
        readcount:= readcount-1;
        If readcount=0 then Signal(wrt);//如果是最后一个读者离开，需要释放写操作状态
        Signal(mutex);
    end
    //写者writer 线程
    Writeri:begin
        resting;//休息状态
 Wait(wrt);//等待写操作
        writing; //进入 writing 状态
        Signal(wrt);//释放写操作
    end
coend
```

2. **写优先策略**

```
1. semaphore resource = 1;  //用于写者和写者、读者和写者互斥访问文件
2. semaphore wmuter = 1; //用于写者和写者互斥访问writercount
3. semaphore rmutex = 1; //用于互斥访问readcount
4. semaphore readTry = 1; //用于实现写者优先
5. int readcount = 0;   //记录当前读者数量
6. int writercount = 0;  //记录当前写者数量
7.
8. Readeri:begin
9.   wait(readTry);
10.  wait(rmutex);
11.  if(readcount == 0)  wait(resource);//第一个读者，等待访问文件
12.  readcount++;
13.  signal(rmutex);
14.  signal(readTry);
15.  reading;//进入reading操作
16.  wait(rmutex);
17.  readcount--;
18.  if(readcount == 0)  signal(resource);//最后一个读者，释放文件
19.  signal(rmutex);
20.    end
21.Writeri:begin
22.  wait(wmutex);
23.  writercount++;
24.  if(writercount == 1)  wait(readTry);//第一个写者，等待读者释放
25.  signal(wmutex);
26.  wait(resource);//等待文件
27.  writing;//进入writing操作
28.  signal(resource);//释放文件
29.  wait(wmutex);
30.  writercount--;
31.  if(writercount == 0)  signal(readTry);//最后一个写者，释放读者
32.  signal(wmutex);
33.    end
34.
```

四、 思考题

1. 程序中使用到了互斥变量和临界资源来实现互斥，这两者在实现机制上有不同吗，若有，是什么不同？

互斥变量和临界资源都是用来实现多线程同步的机制。

互斥变量是一种同步对象，用于协调共享资源的单独访问。互斥变量是一种同步机制，用于保护临界资源的互斥访问。多个进程(线程)均可以访问到一个互斥变量，通过对互斥变量加锁，从而来保护一个临界区，防止其它进程(线程)同时进入临界区，保护临界资源互斥访问。互斥锁需要满足三个条件：互斥、不同线程的临界区没有重叠、只有拥有锁的线程才能解锁。

临界资源是指一段代码或公共资源，通过对多线程的串行化来访问，速度快，适合控制数据访问。临界区是保证在某一时刻只有一个线程能访问数据的简便办法，在任意时刻只允许一个线程对共享资源进行访问。如果有多个线程试图同时访问临界区，那么在有一个线程进入后其他所有试图访问此临界区的线程将被挂起，并一直持续到进入临界区的线程离开。临界区在被释放后，其他线程可以继续抢占，并以此达到用原子方式操作共享资源的目的。

因此，互斥变量和临界资源在实现机制上是不同的。互斥变量是一种同步对象，而临界资源是指一段代码或公共资源。

2. 考虑演示程序是怎样实现在随机调度进程时间、控制进程延迟时间、关闭进程时间的，找出相关函数，变量，并说明。

srand((unsigned)time(NULL) * (writerNum + 1))，这是一个随机数生成器，它使用了一个名为“time”的函数来生成一个随机种子。这个种子是由当前时间和读者数量加1的乘积得到的。这个随机种子将被用来生成随机数，以便在进程之间随机调度时间。

程序使用了一个名为“P_DELAY”的变量来控制进程的延迟时间。P_DELAY　rand()/25*10;它将生成一个介于0和400之间的随机数，这个数字将被用作进程延迟的时间。另外，程序中间的Sleep()函数被用来模拟进程之间的延迟。

并且，在这个程序中，关闭进程的时间也是通过使用Sleep()函数来模拟的。当一个进程完成它的工作后，它会调用Sleep()函数来暂停执行一段时间。这个时间是一个随机数，它在0到400之间。