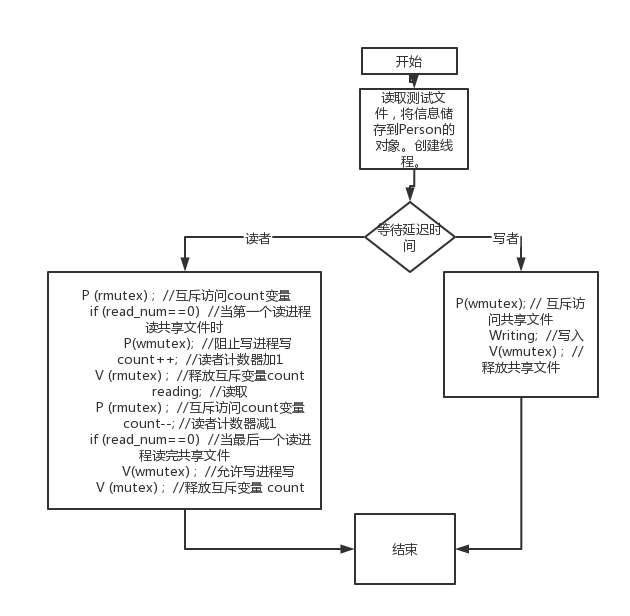
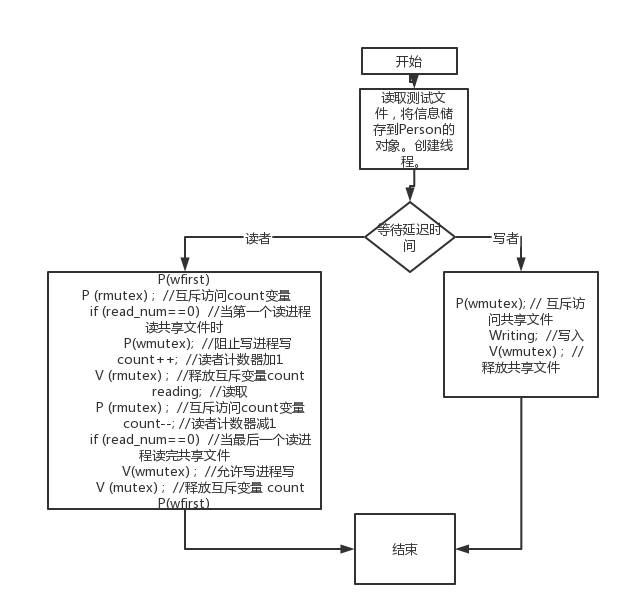
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 实习题目：读者写者问题 | | | | |
| 完成人 | 姓名： | 课序号： | 班级： | 学号： |
| 实习内容及要求简要描述 | 创建一个控制台进程。此进程包含 n 个线程。用这 n 个线程来表示 n 个读者或写者。每个线程按相应测试数据文件(后面有介绍)的要求进行读写操作。用信号量机制分别实现读者优先和写者优先的读者-写者问题。  读者-写者问题的读写操作限制(包括读者优先和写者优先)：  1)写-写互斥，即不能有两个写者同时进行写操作。  2)读-写互斥，即不能同时有一个线程在读，而另一个线程在写。  3)读-读允许，即可以有一个或多个读者在读。  读者优先的附加限制：如果一个读者申请进行读操作时已有另一个读者正在进行读操作，则该读者可直接开始读操作。  写者优先的附加限制：如果一个读者申请进行读操作时已有另一写者在等待访问共享资源，则该读者必须等到没有写者处于等待状态后才能开始读操作。  运行结果显示要求：要求在每个线程创建、发出读写操作申请、开始读写操作和结束读写操作时分别显示一行提示信息，以确定所有处理都遵守相应的读写操作限制。 | | | |
| 主要代码结构  （附注释） | #include<thread>  #include<iostream>  #include<windows.h>  #include<fstream>  #include<mutex>  #define maxNumber 50//并发线程最大数量  class Person{//储存每个对象的相关信息，用于后续创建线程提供参数      private:          int number;//编号          char type;//类型，读者或写者          float delay;//延迟时间          float last;//行为持续时间      public:          Person(){};//无参构造函数          Person(int number,char type,float delay,float last)//含参构造函数          char getType()          int getNumber()          float getDelay()          float getLast()  };  mutex wmudex;//写互斥量  mutex rmudex;//读互斥量  mutex w\_num\_mutex;//发起请求的写者数量的互斥量  mutex r\_num\_mutex;//在读读者数量的互斥量  int reader\_num=0,write\_num=0;//发起请求的写者数量，在读读者数量  ////////////////////////////////////////////////  读者优先：  void func(Person arg)//行为  {      Sleep(arg.getDelay()\*1000);//等待延迟时间      if(arg.getType()=='R')//读者行为      {         第一个读者锁住写互斥量  开始阅读  ...          Sleep(arg.getLast()\*1000);//行为持续时间          ...  最后一个读者释放写互斥量        }      else if(arg.getType()=='W')//写者行为      {          第一个写者锁住读信号量          Sleep(arg.getLast()\*1000);//行为持续时间          最后一个发起了写请求的人结束写后释放读信号量      }  }  ////////////////////////////////////////////////////  写者优先：  void func(Person arg)//行为  {      Sleep(arg.getDelay()\*1000);//等待延迟时间      if(arg.getType()=='R')//读者行为      {  P(wfirst)  第一个读者锁住写互斥量  开始阅读  ...  Sleep(arg.getLast()\*1000);//行为持续时间  ...  最后一个读者释放写互斥量  V(wfirst)      }      else if(arg.getType()=='W')//写者行为      {  第一个写者锁住读信号量  Sleep(arg.getLast()\*1000);//行为持续时间  最后一个发起了写请求的人结束写后释放读信号量      }  }  int main()//  } | | | |
| 结果分析（或错误原因分析） | 结果显示符合实验要求。 | | | |

附录一、流程图

**读者优先：**



**写者优先：**



附录二、源程序

**读者优先：**

#include<thread>

#include<iostream>

#include<windows.h>

#include<fstream>

#include<mutex>

#define maxNumber 50

using namespace std;

////////

class Person{

    private:

        int number;

        char type;

        float delay;

        float last;

    public:

        Person(){};

        Person(int number,char type,float delay,float last)

        {

            this->number=number;

            this->type=type;

            this->delay=delay;

            this->last=last;

        }

        char getType(){

            return type;

        }

        int getNumber(){

            return number;

        }

        float getDelay(){

            return delay;

        }

        float getLast(){

            return last;

        }

};

mutex wmudex;

mutex rmudex;

mutex w\_num\_mutex;

mutex r\_num\_mutex;

mutex wfirst;

int reader\_num=0,write\_num=0;

void func(Person arg)

{

    Sleep(arg.getDelay()\*1000);

    if(arg.getType()=='R')

    {

        cout<<"reader"<<arg.getNumber()<<"发出读请求"<<endl;

        rmudex.lock();

        if(reader\_num==0)

            wmudex.lock();

        r\_num\_mutex.lock();

        reader\_num++;

        r\_num\_mutex.unlock();

        rmudex.unlock();

        cout<<"reader"<<arg.getNumber()<<"开始阅读"<<endl;

        /////

        Sleep(arg.getLast()\*1000);

        cout<<"reader"<<arg.getNumber()<<"结束阅读"<<endl;

        rmudex.lock();

        r\_num\_mutex.lock();

        reader\_num--;

        r\_num\_mutex.unlock();

        if(reader\_num==0)

            wmudex.unlock();

        rmudex.unlock();

    }

    else if(arg.getType()=='W')

    {

        cout<<"writer"<<arg.getNumber()<<"发出写请求"<<endl;

        w\_num\_mutex.lock();

        write\_num++;

        w\_num\_mutex.unlock();

        wmudex.lock();

        if(write\_num==1)

            rmudex.lock();

        cout<<"writer"<<arg.getNumber()<<"开始写作"<<endl;

    /////

        Sleep(arg.getLast()\*1000);

        cout<<"writer"<<arg.getNumber()<<"结束写作"<<endl;

        wmudex.unlock();

        w\_num\_mutex.lock();

        write\_num--;

        w\_num\_mutex.unlock();

        if(write\_num==0)

            rmudex.unlock();

    }

}

int main()

{

    thread list[maxNumber];

    Person people[maxNumber];

    int n=0;

    fstream f;

    f.open("information.txt",ios::in);

    int number=0;

    char type=' ';

    float delay,last;

    while(f>>number>>type>>delay>>last)

    {

        //cout<<number<<" "<<type<<" "<<delay<<" "<<last<<endl;

        people[n]= Person(number,type,delay,last);

        n++;

        if(type=='R')

        cout<<"reader"<<n;

        else cout<<"writer"<<n;

        cout<<"线程创建"<<endl;

    }

    f.close();

    /////

    for(int i=0;i<n;i++)

    {

        list[i]= thread(func,people[i]);

    }

    for(int i=0;i<n;i++)

    {

        list[i].join();

    }

    //system("pause");

    return 0;

}

**写者优先：**

新增互斥信号量wfirst,稍微改动func()。

void func(Person arg)

{

    Sleep(arg.getDelay()\*1000);

    if(arg.getType()=='R')

    {

        cout<<"reader"<<arg.getNumber()<<"发出读请求"<<endl;

        wfirst.lock();

        rmudex.lock();

        r\_num\_mutex.lock();

        if(reader\_num==0)

            wmudex.lock();

        reader\_num++;

        r\_num\_mutex.unlock();

        rmudex.unlock();

        cout<<"reader"<<arg.getNumber()<<"开始阅读"<<endl;

        /////

        Sleep(arg.getLast()\*1000);

        cout<<"reader"<<arg.getNumber()<<"结束阅读"<<endl;

        //rmudex.lock();

        r\_num\_mutex.lock();

        reader\_num--;

        if(reader\_num==0)

            wmudex.unlock();

        r\_num\_mutex.unlock();

        //rmudex.unlock();

        wfirst.unlock();

    }

    else if(arg.getType()=='W')

    {

        cout<<"writer"<<arg.getNumber()<<"发出写请求"<<endl;

        w\_num\_mutex.lock();

        write\_num++;

        if(write\_num==1)

            wfirst.lock();

        w\_num\_mutex.unlock();

        wmudex.lock();

        // if(write\_num==1)

        //     rmudex.lock();

        cout<<"writer"<<arg.getNumber()<<"开始写作"<<endl;

    /////

        Sleep(arg.getLast()\*1000);

        cout<<"writer"<<arg.getNumber()<<"结束写作"<<endl;

        wmudex.unlock();

        w\_num\_mutex.lock();

        write\_num--;

         if(write\_num==0)

            wfirst.unlock();

        w\_num\_mutex.unlock();

        // if(write\_num==0)

        //     rmudex.unlock();

    }

}

附录三、运行结果截屏

读者优先：



写者优先：

