

操作系统实验一

读者-写者问题



2017-12-10

[裴子祥 计科七班 学号2015211921]

[指导老师：孟祥武]

目录

[一．实验内容与要求 1](#_Toc503277138)

[二． 程序设计与实现 1](#_Toc503277139)

[1． 读者优先 1](#_Toc503277140)

[信号量： 1](#_Toc503277141)

[读者： 1](#_Toc503277142)

[写者： 2](#_Toc503277143)

[2． 写者优先 2](#_Toc503277144)

[信号量： 2](#_Toc503277145)

[读者： 2](#_Toc503277146)

[写者： 3](#_Toc503277147)

[3． 运行结果 4](#_Toc503277148)

[4． 源程序附件 8](#_Toc503277149)

# **一．实验内容与要求**

**题目：**读者优先和写者优先的读者-写者问题实践解决。

**实验内容：**在Windows 环境下，创建一个包含n 个线程的控制进程。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的要求，进行读写操作。

读者-写者问题的读写操作限制：

1. 写-写互斥。
2. 读-写允许。
3. 读-读允许。

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。

**实验环境：**

**MICSOFT WINDOW 10**

**Visual Studio 2015**

1. 程序设计与实现
2. 读者优先

**过程说明：**在读者优先的情况下：除非有写者在写文件，否则没有一个读者需要等待。

新读者到：

* 无读者、写者，新读者可以读
* 有写者等，但有其它读者正在读，则新读者也可以读
* 有写者写，新读者等

新写者到：

* 无读者、写者，新写者可以写
* 有读者读，新写者等待
* 有其它写者，新写者等待

信号量：

int readCount **=** 0**;** //当前读者数目

Semaphore mutex **=** 1**;** //互斥信息，用于readCount的互斥修改

CRITICAL\_SECTION CommonArea**;**//临界区域，初始时是空闲状态

读者：

P**(**mutex**);** //占用互斥量

readCount**++;** //读者数+1

**if(**readCount**==**1**)**

**{**//如果是第一个读者

//则等待写者释放临界资源

等待临界资源释放**;**

**}**

V**(**mutex**);** //互斥量释放

开始读操作**;**

//读完后，读者离开

P**(**mutex**);** //占用互斥量

readCount**--;** //读者数减一

**if(**readCount**==**0**)**

**{**//如果没有读者在读了

释放临界资源**;**

**}**

V**(**mutex**);** //互斥量释放

写者：

等待临界资源释放**;**

开始写操作**;**

释放临界资源**;**

1. 写者优先

**过程说明：**一旦一个写者到来，它应该尽快对文件进行写操作。则新来到的读者不允许进行读操作。

新读者到：

* 无读者、写者，新读者可以读
* 有读者读，无写者等，新读者可以读
* 有写者等，但有写者等待，新读者等待
* 有写者写，新读者等

新写者到：

* 无读者、写者，新写者可以写
* 有读者读，新写者等待
* 有写者写，新写者等待

信号量：

int readCount **=** 0**;** //当前读者数目

int writeCount **=** 0**;** //当前写者数目

CRITICAL\_SECTION ReadArea**;** //读者临界区域，写者也能占用

CRITICAL\_SECTION WriteArea**;** //写者临界区域，读者不能

Semaphore mutex1 **=** 1**;** //互斥信息，用于读者临界区的状态互斥修改

Semaphore mutex2 **=** 1**;** //互斥信息，用于readCount的互斥修改

Semaphore mutex3 **=** 1**;** //互斥信息，用于writeCount的互斥修改

读者：

P**(**mutex1**);** //占用互斥量，以修改读者临界区状态

等待ReadArea释放，然后占用它**;**

P**(**mutex2**);** //占用互斥量，以readCount

readCount**++;** //读者数+1

**if(**readCount**==**1**)**

**{**//如果是第一个读者

//等待写者写完

等待WriteArea释放，然后占用它**;**

**}**

V**(**mutex2**);** //释放readCount修改互斥量

释放ReadArea**;**

V**(**mutex1**);** //释放ReadArea状态修改互斥量

开始读操作**;**

P**(**mutex2**);** //占用互斥量，以修改readCount

readCount**--;**

**if(**readCount**==**0**)**

**{**//如果没有读者在读了

释放WriteArea，以唤醒读者**;**

**}**

V**(**mutex2**);**

写者：

P**(**mutex3**);** //占用互斥量，以修改writeCount

writeCount**++;** //写者数+1

**if(**writeCount**==**1**)**

**{**//如果是第一个写者

//等第一个读者读完

等待ReadArea释放，然后占用它**;**

**}**

V**(**mutex3**);**

进入写者临界区**;**

开始写操作**;**

P**(**mutex3**);**

writeCount**--;**

**if(**writeCount**==**0**)**

**{**

//所有写者写完，读者可以读

释放ReadArea**;**

**}**

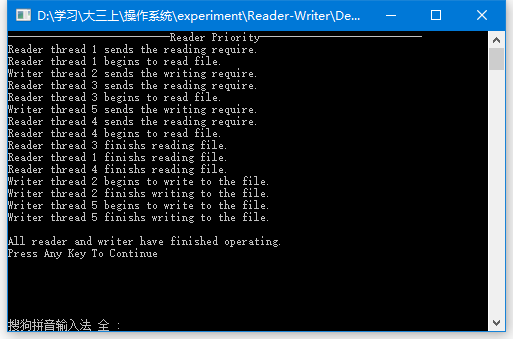
V**(**mutex3**);**

1. 运行结果

**测试用例1eg：**

|  |  |  |  |
| --- | --- | --- | --- |
| 线程序号 | （读者OR写者） | 开始时间 | 持续时间 |
| 1 | R | 3 | 5 |
| 2 | W | 4 | 5 |
| 3 | R | 5 | 2 |
| 4 | R | 6 | 5 |
| 5 | W | 5.1 | 3 |

**读者优先：**



Reader thread 1 sends the reading require.

Reader thread 1 begins to read file.

Writer thread 2 sends the writing require.

Reader thread 3 sends the reading require.

Reader thread 3 begins to read file.

Writer thread 5 sends the writing require.

Reader thread 4 sends the reading require.

Reader thread 4 begins to read file.

Reader thread 3 finishs reading file.

Reader thread 1 finishs reading file.

Reader thread 4 finishs reading file.

Writer thread 2 begins to write to the file.

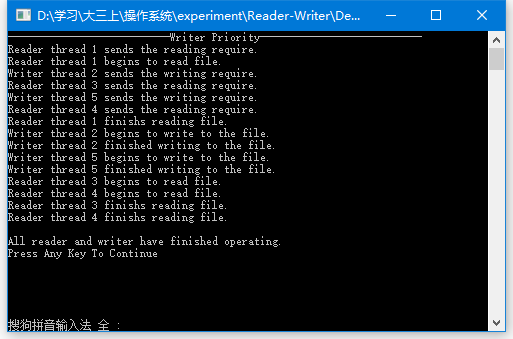
Writer thread 2 finishs writing to the file.

Writer thread 5 begins to write to the file.

Writer thread 5 finishs writing to the file.

All reader and writer have finished operating.

**写者优先：**



Reader thread 1 sends the reading require.

Reader thread 1 begins to read file.

Writer thread 2 sends the writing require.

Reader thread 3 sends the reading require.

Writer thread 5 sends the writing require.

Reader thread 4 sends the reading require.

Reader thread 1 finishs reading file.

Writer thread 2 begins to write to the file.

Writer thread 2 finished writing to the file.

Writer thread 5 begins to write to the file.

Writer thread 5 finished writing to the file.

Reader thread 3 begins to read file.

Reader thread 4 begins to read file.

Reader thread 3 finishs reading file.

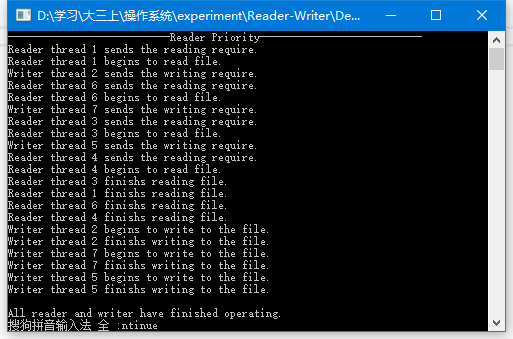
Reader thread 4 finishs reading file.

All reader and writer have finished operating.

**测试用例2eg：**

|  |  |  |  |
| --- | --- | --- | --- |
| 线程序号 | （读者OR写者） | 开始时间 | 持续时间 |
| 1 | R | 1 | 6 |
| 2 | W | 3 | 5 |
| 3 | R | 5 | 2 |
| 4 | R | 6 | 5 |
| 5 | W | 5 | 3 |
| 6 | R | 3 | 4 |
| 7 | W | 4 | 2 |

**读者优先:**



Reader thread 1 sends the reading require.

Reader thread 1 begins to read file.

Writer thread 2 sends the writing require.

Reader thread 6 sends the reading require.

Reader thread 6 begins to read file.

Writer thread 7 sends the writing require.

Reader thread 3 sends the reading require.

Reader thread 3 begins to read file.

Writer thread 5 sends the writing require.

Reader thread 4 sends the reading require.

Reader thread 4 begins to read file.

Reader thread 3 finishs reading file.

Reader thread 1 finishs reading file.

Reader thread 6 finishs reading file.

Reader thread 4 finishs reading file.

Writer thread 2 begins to write to the file.

Writer thread 2 finishs writing to the file.

Writer thread 7 begins to write to the file.

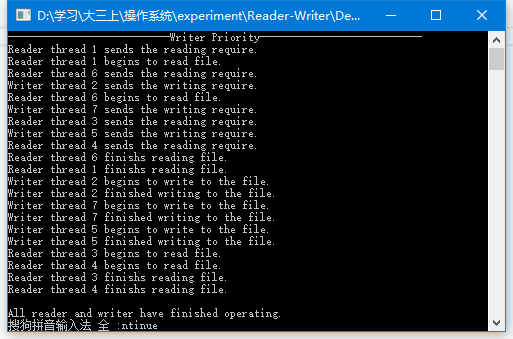
Writer thread 7 finishs writing to the file.

Writer thread 5 begins to write to the file.

Writer thread 5 finishs writing to the file.

All reader and writer have finished operating.

**写者优先:**



Reader thread 1 sends the reading require.

Reader thread 1 begins to read file.

Reader thread 6 sends the reading require.

Writer thread 2 sends the writing require.

Reader thread 6 begins to read file.

Writer thread 7 sends the writing require.

Reader thread 3 sends the reading require.

Writer thread 5 sends the writing require.

Reader thread 4 sends the reading require.

Reader thread 6 finishs reading file.

Reader thread 1 finishs reading file.

Writer thread 2 begins to write to the file.

Writer thread 2 finished writing to the file.

Writer thread 7 begins to write to the file.

Writer thread 7 finished writing to the file.

Writer thread 5 begins to write to the file.

Writer thread 5 finished writing to the file.

Reader thread 3 begins to read file.

Reader thread 4 begins to read file.

Reader thread 3 finishs reading file.

Reader thread 4 finishs reading file.

All reader and writer have finished operating.

1. 源程序附件

#include <Windows.h>

#include <conio.h>

#include <cstdlib>

#include <fstream>

#include <iostream>

#include <cstdio>

#include <ctime>

#include <string>

**using** **namespace** std**;**

#define READER 'R' //读者

#define WRITER 'W' //写者

#define INTE\_PER\_SEC 1000 //每秒时钟中断数目

#define MAX\_THREAD\_NUM 64 //最大线程数

#define MAX\_FILE\_NUM 32 //最大数据文件数目

#define MAX\_STR\_LEN 32 //字符串长度

int readCount **=** 0**;** //读者数目

int writeCount **=** 0**;** //写者数目

CRITICAL\_SECTION RP\_Write**;** //临界区

CRITICAL\_SECTION cs\_Write**;**

CRITICAL\_SECTION cs\_Read**;**

**typedef** struct threadInfo

**{**

int serial**;** //线程序号

char type**;** //线程类别(R为读者，W为写者

double delay**;** //线程创建后，延时时间

double persist**;** //线程持续时间

**}**ThreadInfo**;**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 读者优先-读者进程 \*

\*infoPtr:读者线程信息 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadPriority\_RThread**(**ThreadInfo **\***infoPtr**)**

**{**

HANDLE h\_Mutex**;** //互斥变量

h\_Mutex **=** OpenMutex**(**MUTEX\_ALL\_ACCESS**,** **false,** "mutex\_for\_readCount"**);**

DWORD wait\_for\_mutex**;** //等待互斥变量所有权

DWORD m\_delay**;** //延迟时间

DWORD m\_persist**;** //读文件持续时间

int m\_serial**;** //线程序号

//从参数中获取信息

m\_serial **=** infoPtr**->**serial**;**

m\_delay **=** infoPtr**->**delay**\***INTE\_PER\_SEC**;**

m\_persist **=** infoPtr**->**persist**\***INTE\_PER\_SEC**;**

Sleep**(**m\_delay**);** //暂时挂起，延迟时间的时长

printf**(**"Reader thread %d sends the reading require.\n"**,** m\_serial**);**

//等待互斥信号，保证对readcount的访问、修改、互斥

wait\_for\_mutex **=** WaitForSingleObject**(**h\_Mutex**,** **-**1**);** //P操作

//读者数目增加

readCount**++;**

**if** **(**readCount **==** 1**)**

**{**

//如果是第一个读者，等待资源被写者写完

EnterCriticalSection**(&**RP\_Write**);**

**}**

ReleaseMutex**(**h\_Mutex**);** //V操作

printf**(**"Reader thread %d begins to read file.\n"**,** m\_serial**);**

Sleep**(**m\_persist**);** //持续时间

//退出线程

printf**(**"Reader thread %d finishs reading file.\n"**,** m\_serial**);**

//等待互斥信号，保证对reaadCount的访问、修改互斥

wait\_for\_mutex **=** WaitForSingleObject**(**h\_Mutex**,** **-**1**);**

//读者书减少

readCount**--;**

**if** **(**readCount **==** 0**)**

**{**

//如果读者全部读完，唤醒写者

LeaveCriticalSection**(&**RP\_Write**);**

**}**

ReleaseMutex**(**h\_Mutex**);** //V操作

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 读者优先-写者进程 \*

\*infoPtr:写者线程信息 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReadPriority\_WThread**(**ThreadInfo **\***infoPtr**)**

**{**

DWORD m\_delay**;** //延迟时间

DWORD m\_persist**;** //持续时间

int m\_serial**;** //线程序号

//从参数中获取信息

m\_serial **=** infoPtr**->**serial**;**

m\_delay **=** infoPtr**->**delay**\***INTE\_PER\_SEC**;**

m\_persist **=** infoPtr**->**persist**\***INTE\_PER\_SEC**;**

Sleep**(**m\_delay**);** //延迟等待

printf**(**"Writer thread %d sends the writing require.\n"**,**m\_serial**);**

//等待临界资源

EnterCriticalSection**(&**RP\_Write**);**

//开始写文件

printf**(**"Writer thread %d begins to write to the file.\n"**,**m\_serial**);**

Sleep**(**m\_persist**);**

//退出程序

printf**(**"Writer thread %d finishs writing to the file.\n"**,** m\_serial**);**

//释放临界资源

LeaveCriticalSection**(&**RP\_Write**);**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 读者优先函数 \*

\* fileName:文件名(初始化文件) \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void ReaderPriority**(**char **\***fileName**)**

**{**

DWORD n\_thread **=** 0**;** //线程数目

DWORD serial\_thread**;** //线程序号

DWORD wait\_for\_all**;** //等待所有线程结束

//互斥对象

HANDLE h\_Mutex**;**

h\_Mutex **=** CreateMutex**(NULL,** **false,** "mutex\_for\_readCount"**);**

//线程对象的数组

HANDLE h\_Thread**[**MAX\_THREAD\_NUM**];**

ThreadInfo thread\_info**[**MAX\_THREAD\_NUM**];**

readCount **=** 0**;** //初始化readCount

InitializeCriticalSection**(&**RP\_Write**);** //初始化临界资源

ifstream inFile**;**

inFile**.**open**(**fileName**);**

**if** **(!**inFile**.**is\_open**())**

**{**

cout **<<** "Failed to open the " **<<** fileName **<<** endl**;**

**return;**

**}**

cout **<<** "---------------------------Reader Priority---------------------------" **<<** endl**;**

**while** **(!**inFile**.**eof**())**

**{**

//读入读者或写者初始信息

inFile **>>** thread\_info**[**n\_thread**].**serial**;**

inFile **>>** thread\_info**[**n\_thread**].**type**;**

inFile **>>** thread\_info**[**n\_thread**].**delay**;**

inFile **>>** thread\_info**[**n\_thread**].**persist**;**

//下一个读者或写者

n\_thread**++;**

//读取回车换行符

inFile**.**get**();**

**}**

**for** **(**int i **=** 0**;** i **<** **(**int**)**n\_thread**;** **++**i**)**

**{**

**if** **(**thread\_info**[**i**].**type **==** READER **||** thread\_info**[**i**].**type **==** 'r'**)**

**{**

//创建读者进程

h\_Thread**[**i**]** **=** CreateThread**(NULL,** 0**,**

**(**LPTHREAD\_START\_ROUTINE**)**ReadPriority\_RThread**,**

**&**thread\_info**[**i**],**

0**,** **&**serial\_thread**);**

**}**

**else**

**{**

//创建写者进程

h\_Thread**[**i**]** **=** CreateThread**(NULL,** 0**,**

**(**LPTHREAD\_START\_ROUTINE**)**ReadPriority\_WThread**,**

**&**thread\_info**[**i**],**

0**,** **&**serial\_thread**);**

**}**

**}**

//等待所有线程结束

wait\_for\_all **=** WaitForMultipleObjects**(**n\_thread**,** h\_Thread**,** **true,** **-**1**);**

cout **<<** endl **<<** "All reader and writer have finished operating." **<<** endl**;**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 写者优先-读者线程 \*

\*infoPtr:读者线程信息 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WritePriority\_RThread**(**ThreadInfo **\***infoPtr**)**

**{**

//互斥变量

HANDLE h\_mutex1**;** //用于访问cs\_Read临界区

h\_mutex1 **=** OpenMutex**(**MUTEX\_ALL\_ACCESS**,** **false,** "mutex1"**);**

HANDLE h\_mutex2**;** //用于readCount

h\_mutex2 **=** OpenMutex**(**MUTANT\_ALL\_ACCESS**,** **false,** "mutex2"**);**

DWORD wait\_for\_mutex1**;** //等待互斥变量所有权

DWORD wait\_for\_mutex2**;** //用于readCount

DWORD m\_delay**;** //延迟时间

DWORD m\_persist**;** //读操作持续时间

int m\_serial**;** //线程序号

//从参数中获取线程信息

m\_serial **=** infoPtr**->**serial**;**

m\_delay **=** infoPtr**->**delay**\***INTE\_PER\_SEC**;**

m\_persist **=** infoPtr**->**persist**\***INTE\_PER\_SEC**;**

Sleep**(**m\_delay**);** //延迟等待

printf**(**"Reader thread %d sends the reading require.\n"**,** m\_serial**);**

wait\_for\_mutex1 **=** WaitForSingleObject**(**h\_mutex1**,** **-**1**);** //P操作

//进入读者临界区

EnterCriticalSection**(&**cs\_Read**);** //P操作

//阻塞互斥对象，保证对readCount的访问、修改互斥

wait\_for\_mutex2 **=** WaitForSingleObject**(**h\_mutex2**,** **-**1**);** //P操作

//修改读者数目

readCount**++;**

**if** **(**readCount **==** 1**)**

**{**

//如果是第一个读者，等待写者写完

EnterCriticalSection**(&**cs\_Write**);**

**}**

ReleaseMutex**(**h\_mutex2**);** //V操作

LeaveCriticalSection**(&**cs\_Read**);** //让其它读者进入临界区

ReleaseMutex**(**h\_mutex1**);** //V操作

//开始读操作

printf**(**"Reader thread %d begins to read file.\n"**,** m\_serial**);**

Sleep**(**m\_persist**);** //读持续时间

//退出线程

printf**(**"Reader thread %d finishs reading file.\n"**,** m\_serial**);**

//阻塞互斥对象mutex2,保证对readCount的访问、修改互斥

wait\_for\_mutex2 **=** WaitForSingleObject**(**h\_mutex2**,** **-**1**);** //P操作

readCount**--;**

**if** **(**readCount **==** 0**)**

**{**

//如果所有读者读完，唤醒写者

LeaveCriticalSection**(&**cs\_Write**);**

**}**

ReleaseMutex**(**h\_mutex2**);** //V操作

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 写者优先-写者线程 \*

\*infoPtr:写者线程信息 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WritePriority\_WThread**(**ThreadInfo**\*** infoPtr**)**

**{**

DWORD wait\_for\_mutex3**;** //用于writeCount

DWORD m\_delay**;** //延迟时间

DWORD m\_persist**;** //写文件持续时间

int m\_serial**;** //线程序号

//互斥对象

HANDLE h\_mutex3**;**

h\_mutex3 **=** OpenMutex**(**MUTEX\_ALL\_ACCESS**,** FALSE**,** "mutex3"**);**

//从参数中获得信息

m\_serial **=** infoPtr**->**serial**;**

m\_delay **=** infoPtr**->**delay**\***INTE\_PER\_SEC**;**

m\_persist **=** infoPtr**->**persist **\***INTE\_PER\_SEC**;**

Sleep**(**m\_delay**);**//延迟等待

printf**(**"Writer thread %d sends the writing require.\n"**,** m\_serial**);**

//阻塞互斥对象mutex3,保证对writecount的访问、修改互斥

wait\_for\_mutex3 **=** WaitForSingleObject**(**h\_mutex3**,** **-**1**);**//P操作

//修改写者数目

writeCount**++;**

**if** **(**writeCount **==** 1**)**

**{**

//第一个写者，等待读者读完

EnterCriticalSection**(&**cs\_Read**);**

**}**

ReleaseMutex**(**h\_mutex3**);** //V操作

//进入写者临界区

EnterCriticalSection**(&**cs\_Write**);**

//开始写操作

printf**(**"Writer thread %d begins to write to the file.\n"**,** m\_serial**);**

Sleep**(**m\_persist**);** //写操作持续时间

//退出线程

printf**(**"Writer thread %d finished writing to the file.\n"**,** m\_serial**);**

//离开临界区

LeaveCriticalSection**(&**cs\_Write**);**

//阻塞互斥对象mutex3,保证对writecount的访问、修改互斥

wait\_for\_mutex3 **=** WaitForSingleObject**(**h\_mutex3**,** **-**1**);**//P操作

writeCount**--;**

**if** **(**writeCount **==** 0**)**

**{**

//写者写完，读者可以读

LeaveCriticalSection**(&**cs\_Read**);**

**}**

ReleaseMutex**(**h\_mutex3**);** //V操作

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 写者优先函数 \*

\* fileName:文件名(初始化文件) \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void WriterPriority**(**char **\***fileName**)**

**{**

DWORD n\_thread **=** 0**;** //线程数目

DWORD serial\_thread**;** //线程ID

DWORD wait\_for\_all**;** //等待所有线程结束

//创建互斥对象

HANDLE h\_Mutex1**;** //用于cs\_Read临界区状态修改

h\_Mutex1 **=** CreateMutex**(NULL,** FALSE**,** "mutex1"**);**

HANDLE h\_Mutex2**;** //用于readCount互斥修改

h\_Mutex2 **=** CreateMutex**(NULL,** FALSE**,** "mutex2"**);**

HANDLE h\_Mutex3**;** //用于writeCount互斥修改

h\_Mutex3 **=** CreateMutex**(NULL,** FALSE**,** "mutex3"**);**

//线程对象

HANDLE h\_Thread**[**MAX\_THREAD\_NUM**];**

ThreadInfo thread\_info**[**MAX\_THREAD\_NUM**];**

readCount **=** 0**;** //初始化readCount

writeCount **=** 0**;** //初始化writeCount

InitializeCriticalSection**(&**cs\_Write**);** //初始化临界区

InitializeCriticalSection**(&**cs\_Read**);**

ifstream inFile**;**

inFile**.**open**(**fileName**);**

**if** **(!**inFile**.**is\_open**())**

**{**

cout **<<** "Failed to open the " **<<** fileName **<<** endl**;**

**return;**

**}**

cout **<<** "---------------------------Writer Priority---------------------------" **<<** endl**;**

**while** **(!**inFile**.**eof**())**

**{**

//读入读者或写者初始信息

inFile **>>** thread\_info**[**n\_thread**].**serial**;**

inFile **>>** thread\_info**[**n\_thread**].**type**;**

inFile **>>** thread\_info**[**n\_thread**].**delay**;**

inFile **>>** thread\_info**[**n\_thread**].**persist**;**

//下一个读者或写者

n\_thread**++;**

//读取回车换行符

inFile**.**get**();**

**}**

**for** **(**int i **=** 0**;** i **<** **(**int**)**n\_thread**;** **++**i**)**

**{**

**if** **(**thread\_info**[**i**].**type **==** READER **||** thread\_info**[**i**].**type **==** 'r'**)**

**{**

//创建读者进程

h\_Thread**[**i**]** **=** CreateThread**(NULL,** 0**,**

**(**LPTHREAD\_START\_ROUTINE**)**WritePriority\_RThread**,**

**&**thread\_info**[**i**],**

0**,** **&**serial\_thread**);**

**}**

**else**

**{**

//创建写者进程

h\_Thread**[**i**]** **=** CreateThread**(NULL,** 0**,**

**(**LPTHREAD\_START\_ROUTINE**)**WritePriority\_WThread**,**

**&**thread\_info**[**i**],**

0**,** **&**serial\_thread**);**

**}**

**}**

//等待所有线程结束

wait\_for\_all **=** WaitForMultipleObjects**(**n\_thread**,** h\_Thread**,** **true,** **-**1**);**

cout **<<** endl **<<** "All reader and writer have finished operating." **<<** endl**;**

**}**

int main**()**

**{**

char ch**;**

**while** **(true)**

**{**

printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

printf**(**" 1:Reader Priority\n"**);**

printf**(**" 2:Writer Priority\n"**);**

printf**(**" 3:Exit to Windows \n"**);**

printf**(**"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"**);**

printf**(**"Enter your choice( 1, 2 or 3 ):"**);**

**do**

**{**

scanf\_s**(**"%c"**,** **&**ch**);**

getchar**();**

**}** **while** **(**ch **!=** '1' **&&** ch **!=** '2' **&&** ch **!=** '3'**);**

system**(**"cls"**);**

**if** **(**ch **==** '1'**)**

**{**

//读者优先

ReaderPriority**(**"Reader\_Writer.txt"**);**

**}**

**else** **if** **(**ch **==** '2'**)**

**{**

//写者优先

WriterPriority**(**"Reader\_Writer.txt"**);**

**}**

**else**

**{**

cout **<<** "GOOD BYE" **<<** endl**;**

system**(**"pause"**);**

**return** 0**;** //退出

**}**

//结束

cout **<<** "Press Any Key To Continue"**;**

getchar**();**

system**(**"cls"**);**

**}**

system**(**"pause"**);**

**return** 0**;**

**}**