

# 操作系统

## 读者-写者问题实验报告

**编译环境：**Microsoft Visual Studio 2019

**Windows SDK 版本：**10.0

**平台工具集：**Visual Studio 2019 (v142)

**语言：**C/C++

### 一、实验需求：

本课程实验内容引自《Windows 内核实验教程》(陈向群、林斌等编著，机械工业出版社，2002.9)。

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。请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：

- 1) 写-写互斥；
- 2) 读-写互斥；
- 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。

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，以确信所有处理都遵守相应的读写操作限制。

### 二、实验原理：

#### 读者优先：

信号量：mutex=1;//对 read\_count 的互斥操作

RP\_Write=1;//保证读者优先，读者与写者互斥

读者：

```
wait(mutex);
```

```
read_count++;
```

```
if(read_count==1)
```

```
wait(&RP_Write);
```

```

signal(mutex);

读临界区……

wait(mutex);

read_count--;

if(read_count==0)

    signal(&RP_Write);

signal(mutex);

```

写者:

```

wait(&RP_Write);

写临界区……

signal(&RP_Write);

```

写者优先:

信号量: mutex1=1;//保证每个读者按顺序依次进入临界区  
 mutex2=1;//对 read\_count 的互斥操作  
 mutex3=1;//对 write\_count 的互斥操作  
 cs\_Read =1;//保证写者优先, 读者与写者互斥  
 cs\_Write=1;//保证如果有读者正在读, 写者等待当前读者读完后再

写

读者:

```

wait(mutex1);

wait(&cs_Read);

wait(mutex2);

read_count++;

if(read_count==1)

    wait(&cs_Write);

signal(mutex2);

```

```
signal(&cs_Read);

signal(mutex1);

读临界区……

wait(mutex2);

read_count--;

if(read_count==0)

    signal(&cs_Write);

signal(mutex2);
```

写者:

```
wait(mutex3);

write_count++;

if(write_count==1)

    wait(&cs_Read);

signal(mutex3);

wait(&cs_Write);

写临界区……

signal(&cs_Write);

wait(mutex3);

write_count--;

if(write_count==0)

    signal(&cs_Read);

signal(mutex3);
```

三、输入:

1、屏幕输入 1 则选择读者优先，调用 ReaderPriority("thread.dat")函数；选择 2 则选择写者优先，调用 WriterPriority("thread.dat")函数；选择 3 则退出。

2、从 thread.dat 文件读入读者写者线程信息。

## 四、输出：

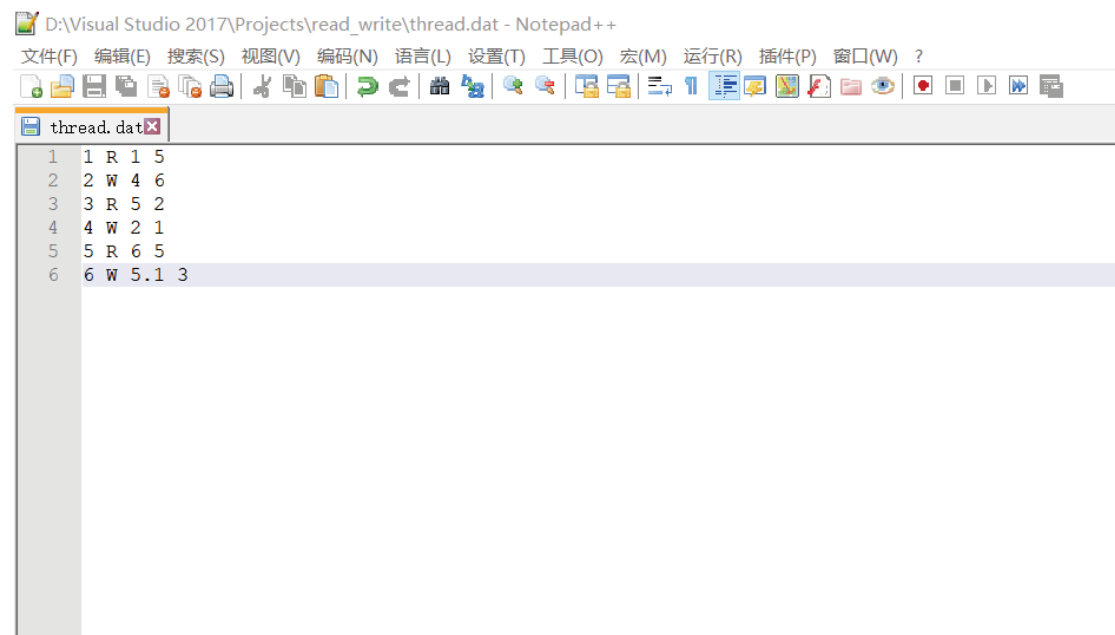
屏幕输出各读者写者线程信息。

## 五、详细代码：

各函数的作用及详细实现和代码见附录工程文件。

## 六、测试样例：

thread.dat:

A screenshot of the Notepad++ application window. The title bar reads "D:\Visual Studio 2017\Projects\read\_write\thread.dat - Notepad++". The menu bar includes "文件(F)", "编辑(E)", "搜索(S)", "视图(V)", "编码(N)", "语言(L)", "设置(T)", "工具(O)", "宏(M)", "运行(R)", "插件(P)", "窗口(W)", and "?". The toolbar contains various icons for file operations and editing. The editor window shows a file named "thread.dat" with the following content:

```
1 1 R 1 5
2 2 W 4 6
3 3 R 5 2
4 4 W 2 1
5 5 R 6 5
6 6 W 5.1 3
```

运行后：

等待用户输入优先级

```
D:\Visual Studio 2017\Projects\read_write\Debug\read_write.exe
*****
1:读者优先
2:写者优先
3:退出
*****
请输入(1,2 或 3):
```

输入 1:

可以看出，写者进程 4、2、6 虽然都请求了写操作，但是却是等待所有的读者 1、3、5 读完了再按顺序进行的写入操作，符合读者优先的要求

```
D:\Visual Studio 2017\Projects\read_write\Debug\read_write.exe
读者优先:
读者进程 1 发出读请求
读者进程 1 开始读文件
写者进程 4 发出写请求
写者进程 2 发出写请求
读者进程 3 发出读请求
读者进程 3 开始读文件
写者进程 6 发出写请求
读者进程 5 发出读请求
读者进程 5 开始读文件
读者进程 1 完成读文件
读者进程 3 完成读文件
读者进程 5 完成读文件
写者进程 4 开始写文件
写者进程 4 完成写文件
写者进程 2 开始写文件
写者进程 2 完成写文件
写者进程 6 开始写文件
写者进程 6 完成写文件
所有的读者写者均已完成操作
按任意键结束:
```

输入 2:

可以看出，除了第一个读者进程 1 在读文件外，虽然读者 3、5 都请求了读操作，但由于写者进程 4、2、6 请求了写操作，因为写者优先，所以读者 3、5 都要等待写者 4、2、6 写入完成才能读，而写者 4、2、6 必须等待读者进程 1 读操作完成后才能写。读者 1 读操作完成后，写者 4、2、6 按顺序写入文件，写入完成后，读者 3、5 同时读文件，符合写者优先的要求

```
D:\Visual Studio 2017\Projects\read_write\Debug\read_write.exe
写者优先:
读者进程 1 发出读请求
读者进程 1 开始读文件
写者进程 4 发出写请求
写者进程 2 发出写请求
读者进程 3 发出读请求
写者进程 6 发出写请求
读者进程 5 发出读请求
读者进程 1 完成读文件
写者进程 4 开始写文件
写者进程 4 完成写文件
写者进程 2 开始写文件
写者进程 2 完成写文件
写者进程 6 开始写文件
写者进程 6 完成写文件
读者进程 3 开始读文件
读者进程 5 开始读文件
读者进程 3 完成读文件
读者进程 5 完成读文件
所有的读者写者均已完成操作
按任意键结束:
```

输入 3:

释放线程，程序退出

```
Microsoft Visual Studio 调试控制台
D:\Visual Studio 2017\Projects\read_write\Debug\read_write.exe (进程 19840) 已退出，代码为 0。
按任意键关闭此窗口。 . .
```

## 七、实验总结：

通过这个读者-写者实验，我更为深刻地认识了读者-写者问题，了解了各种线程、信号量等 API 接口及函数的使用。我还对原版程序进行了简单的修改，使之屏幕显示更为人性化。以及通过这个实验，我自己又写了一个关于多生产者，多消费者，多缓冲区的程序，也在所附工程文件中。