操作系统实验指导书

进程同步控制

北京邮电大学 计算机学院

目录

一、实验目的

本实验旨在让学生动手设计一个进程同步控制的实验，更深刻的理解进程协作机制。

二、实验要求及内容

利用信号量机制，实现读者写者问题的解决方案。

在 OpenEuler/Linux 环境下，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）：

写-写互斥，即不能有两个写者同时进行写操作。

读-写互斥，即不能同时有一个线程在读，而另一个线程在写。

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、开始读写操作和结果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。

三、实验形式

个人实现

四、实验设备环境

OpenEuler/Linux 环境 PC 机，C++开发环境。

五、实验步骤

5.1 读者-写者 实验原理及步骤

数据文件:

文件 Thread.txt 含有 n 行数据，每一行数据代表一个读写请求。比如：

R，100　　//读请求，读操作的持续时间为 100ms

W，200　　//写请求，写操作的持续时间为 200ms

持续时间可以自己设定。

临界区

可以定义一个共享环形队列，初始化队列长度为 100，初始化队列元素为 0。写者进入临界区，向队尾添加一个随机数；读者进入临界区，从队首取走一个数据。

读者优先：

主函数首先读取数据文件 Thread.txt，为每一行请求创建一个线程，其中读请求创建读者线程，读者线程执行函数 RP_ReaderThread()，写请求创建写者线程，写者线程执行函数 RP_WriterThread()。

单纯使用信号量不能解决读者与写者问题，必须引入计数器 read_count 对读进程计数； mutex 是用于对计数器 read_count 操作的互斥信号量；RP_Write 表示是否允许写的信号量.

RP_ReaderThread()函数的实现如下：

```
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);
```

RP_WriterThread()函数的实现如下：

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

写临界区……

signal(&RP_Write);

读者优先的设计思想是读进程只要看到有其它读进程正在读，就可以继续进行读；写进程必须等待所有读进程都不读时才能写，即使写进程可能比一些读进程更早提出申请。该算法只要还有一个读者在活动，就允许后续的读者进来，该策略的结果是，如果有一个稳定的读者流存在，那么这些读者将在到达后被允许进入。而写者就始终被挂起，直到没有读者为止。

写者优先：

主函数首先读取文件 Thread.txt，为每一行请求创建一个线程，其中读请求创建读者线程，读者线程执行函数 WP_ReaderThread（），写请求创建写者线程，写者线程执行函数 WP_WriterThread()。

在读者优先的算法的基础上增加了一个排队信号量 cs_Read，读、写进程在每次操作前都要等待 cs_Read 信号量。

WP_WriterThread()函数实现如下：

```
wait(mutex1);
write_count++;
if(write_count==1)
   wait(&cs_Read);
signal(mutex1);
wait(&cs_Write);
写临界区……
signal(&cs_Write);
wait(mutex1);
write_count--;
if(write_count==0)
   signal(&cs_Read);
signal(mutex1);
```

WP_ReaderThread()函数实现如下：

```
wait(&cs_Read);
wait(mutex2);
```

```
read_count++;
if(read_count==1)
   wait(&cs_Write);
signal(mutex2);
signal(&cs_Read);
读临界区……
wait(mutex2);
read_count--;
if(read_count==0)
      signal(&cs_Write);
signal(mutex2);
```

写者优先的设计思想是在一个写者到达时如果有正在工作的读者，那么该写者只要等待正在工作的读者完成，而不必等候其后面到来的读者就可以进行写操作。该算法当一个写者在等待时，后到达的读者是在写者之后被挂起，而不是立即允许进入。

5.2 **读者-写者** 编译及运行方法

实验前必须安装好 C++编译软件。

将自己设计的 C 语言程序 thread.cpp 以及测试数据文件 thread.txt 在 C++中组成一个工程并调试正确，再编译链接生成 thread.exe 可执行文件。

测试数据文件，可以自行创建一个 thread.txt 文件，其中包括 n 行测试数据，分别描述创建的 n 个线程是读者还是写者，以及读写操作的持续时间。

要求：

（1）将数据文件名称和其中的行数作为运行参数。

（2）记录每个线程的创建时间、进入/退出临界区的时间。

（3）显示每个线程在临界区中的操作结果，比如，哪个写者向队尾加入了一个数字，显示数字。哪个读者，从队首取走一个数字，显示取出的数据。

（4）建议运行期间所有输出，保存在一个输出文件中，便于分析。

（5）主线程等待所有读写线程结束后，再继续。

5.3 **读者-写者** 实验结果说明

程序运行结果分析首先需要从理论的角度分析，以测试数据文件 thread.txt 画出相应的读者优先和写者优先的线程运行顺序，如图 1。再将实际的实验运行结果与理论分析结果进行对比分析，以验证实验的正确性，并分析结果。

另外，可通过更改测试数据再次验证程序的正确性，同样需要先理论分析，再用实验结果与理论分析对比分析。

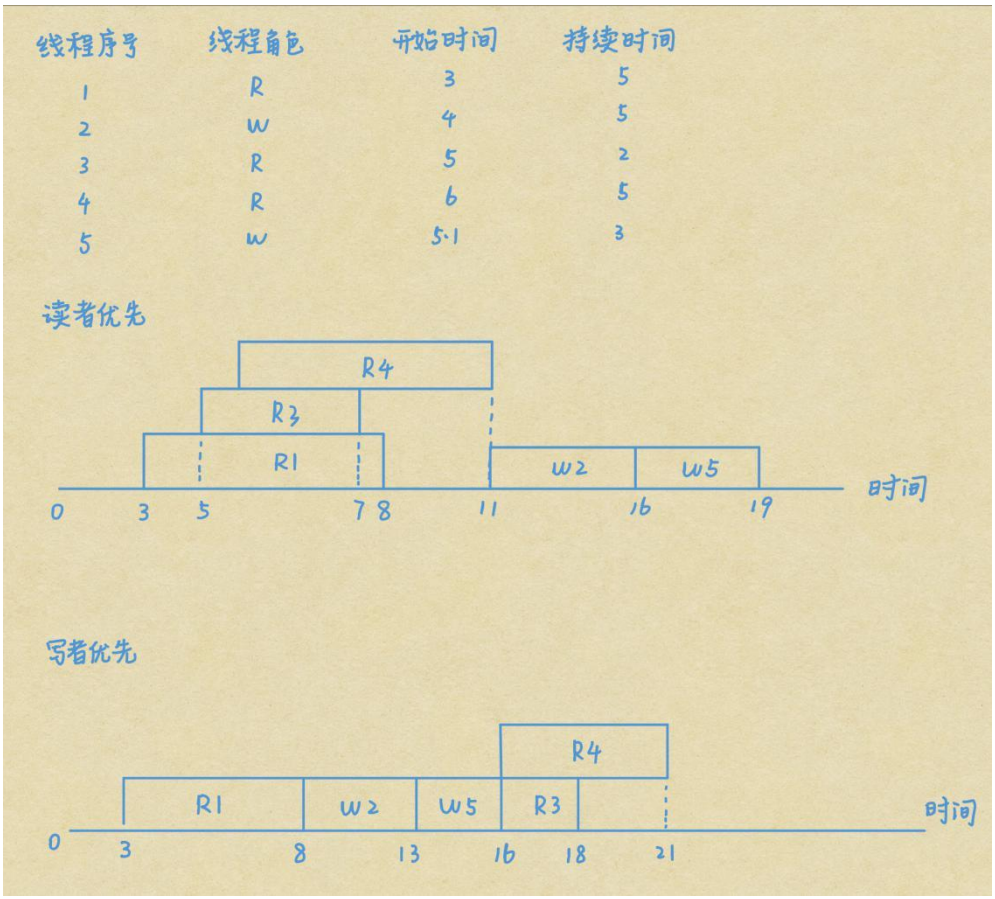

图 1 读者-写者实验测试数据理论分析图

六、实验报告书写要求

学生的实验报告内容要与本指导书内容一致，要有：

实验目的：描述本次实验任务、目的；

实验要求及内容：描述本次实验具体要求，实验的具体内容，另外写出实验原理、整体思想，可给出相应原理图及整体框架图；

实验设备环境：描述本次实验的实验环境；

实验步骤（原理步骤、编译运行方法、实验结果说明）：首先写明实验的具体步骤，描述每个步骤对应的原理、流程图，写明实验的具体编译运行方法，最后展示实验结果，需要从理论分析和实际结果分析两方面分析对比实验结果，得出结论，会出现什么问题，如何解决；

实验心得体会：实验报告最后需要总结和写出心得体会，完成本次实验编写代码的时间、上机调试的时间、编程工具遇到的问题、编程语言遇到的问题、总结本次实验。