Reader Writer Problem

史柠玮-2014211283

January 9, 2017

Contents

1 实验目的

深刻理解读者写者问题

2 实验要求

2.1 基本要求

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;

2.2 附加限制

- 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。

2.3 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。

2.4 测试数据文件格式

测试数据文件包括 n 行测试数据，分别描述创建的 n 个线程是读者还是写者，以及读写操作的开始时间和持续时间。每行测试数据包括四个字段，各字段间用空格分隔。

1. 第一字段为一个正整数，表示线程序号.
2. 第二字段表示相应线程角色，R 表示读者是，W 表示写者。
3. 第三字段为一个正数，表示读写操作的开始时间。线程创建后，延时相应时间（单位为秒）后发出对共享资源的读写申请。
4. 第四字段为一个正数，表示读写操作的持续时间。当线程读写申请成功后，开始对共享资源的读写操作，该操作持续相应时间后结束，并释放共享资源.

下面是一个测试数据文件的例子：

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

3 实验环境

- Fedora 25 64 位
- Pycharm (IDE)
- Python 3.5

4 实验结果

注[1]

- 读者优先:

```
1.ReaderPriority
2.WriterPriority
3.Exit
Please input a number
1
ReaderPriority:
3.00 Reader thread 1 send the reading require.
3.00 Reader thread 1 began to read file.
4.00 Writer thread 2 send the writing require.
5.00 Reader thread 3 send the reading require.
5.00 Reader thread 3 began to read file.
5.10 Writer thread 5 send the writing require.
6.00 Reader thread 4 send the reading require.
6.01 Reader thread 4 began to read file.
7.00 Reader thread 3 finished reading file.
8.01 Reader thread 1 finished reading file.
11.01 Reader thread 4 finished reading file.
11.02 Writer thread 2 began to write to the file.
16.02 Writer thread 2 finished writing to the file.
16.03 Writer thread 5 began to write to the file.
19.04 Writer thread 5 finished writing to the file.
All reader and writer have finished operating.
```

- 写者优先:

[1] 代码见附件

```
1.ReaderPriority
2.WriterPriority
3.Exit
Please input a number
2
WriterPriority:
3.00 Reader thread 1 send the reading require.
3.00 Reader thread 1 began to read file.
4.00 Writer thread 2 sent the writing acquire.
5.00 Reader thread 3 send the reading require.
5.11 Writer thread 5 sent the writing acquire.
6.00 Reader thread 4 send the reading require.
8.01 Reader thread 1 finished reading file.
8.01 Writer thread 2 began to write to the file.
13.02 Writer thread 2 finished writing to the file.
13.04 Writer thread 5 began to write to the file.
16.05 Writer thread 5 finished writing to the file.
16.05 Reader thread 3 began to read file.
16.05 Reader thread 4 began to read file.
18.07 Reader thread 3 finished reading file.
21.06 Reader thread 4 finished reading file.
All reader and writer have finished operating.
```