浙江工业大学

操作系统原理实验报告

(2021 级)

实验二：读者写者实验

学生姓名：温家伟

学生学号：202103151422

学科专业：大数据分析 2101 班

所在学院：理学院

提交日期：2023 年 11 月 19 日

目录

1 实验目的

熟悉多线程编程

熟悉使用信号量机制解决同步问题

2 实验要求

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。

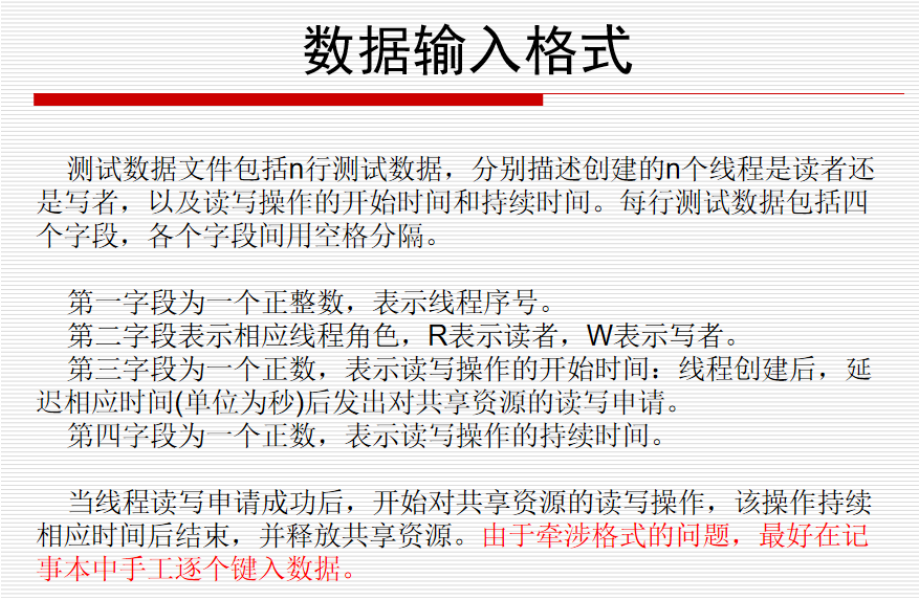

图 1: 实验要求

3 实验内容

读者优先算法: 在读者优先算法中，当有读者正在读取共享资源时，写者会被阻塞，直到所有读者完成读取。这样可以确保共享资源对于读取操作是安全的。伪代码逻辑如下：

当有读者请求读取时，首先检查是否有写者正在写入，如果有则等待，否则增加读者计数。当读者完成读取后，减少读者计数，如果此时读者计数为 0，则唤醒任何等待的写者线程。

数据结构：

- *int readerCount = 0*：当前正在读取的读者数量
- *int writerCount = 0*：当前正在写入的写者数量
- *semaphore rcsignal, filesrc, read_s*：用于同步读者和写者线程的信号量

读者线程函数：

```
void *rp_threadReader(void *p) {
// 读者线程的处理逻辑
// ...
}
```

写者线程函数：

```
void *rp_threadWriter(void *p) {
// 写者线程的处理逻辑
// ...
}
```

主函数调用：

```
void ReaderPriority() {
// 主函数中的调用逻辑
// ...
}
```

写者优先算法

写者优先算法是另一种解决读者-写者问题的算法，其核心思想是优先保障写者对共享资源的访问。在该算法中，当有写者请求写入共享资源时，会阻塞其他读者和写者，直到写者完成写入。下面是写者优先算法的伪代码描述：

当有写者请求写入时，首先检查是否有其他读者或写者正在访问共享资源，如果有则等待，否则增加写者计数。当写者完成写入后，减少写者计数，并根据优先级唤醒其他等待的读者或写者线程。

在写者优先算法中，当有写者请求写入共享资源时，其他读者和写者都会被阻塞，直到写者完成写入。这样可以确保共享资源对于写入操作是安全的。伪代码逻辑如下：

数据结构：

- *int readerCount = 0*：当前正在读取的读者数量
- *int writerCount = 0*：当前正在写入的写者数量
- *semaphore RCSIGNAL, writeCountSignal, WRT, READ_S*：用于同步读者和写者线程的信号量

读者线程函数：

```
void *wp_threadReader(void *p) {
// 读者线程的处理逻辑
// ...
}
```

写者线程函数：

```
void *wp_threadWriter(void *p) {
// 写者线程的处理逻辑
// ...
}
```

主函数调用：

```
void WriterPriority() {
// 主函数中的调用逻辑
// ...
}
```

4 源代码

```
1
2  #include <iostream>
3  #include <pthread.h>
4  #include <semaphore.h>
5  #include <unistd.h>
6
7  using namespace std;
8
9  int readerCount = 0;
10 int writerCount = 0;
11 sem_t rcsignal;
12 sem_t filesrc;
13 sem_t read_s;
14
15 sem_t RCSIGNAL;
16 sem_t writeCountSignal;
17 sem_t WRT;
18 sem_t READ_S;
19
20 class thread_info
21 {
22 public:
23     int thread_id;
24     char thread_type;
25     double thread_delay;
26     double thread_lastTime;
27 };
28
29 // 读者优先
30 void *rp_threadReader(void *p)
31 {
32     int num_thread = ((thread_info *)p)->thread_id;
```

```
33      double delaytime = ((thread_info
            *)p)->thread_delay;
34      double duration = ((thread_info
            *)p)->thread_lastTime;
35
36      usleep(delaytime * 1000000);
37      cout << "读者线程" << num_thread << "
            发送了一个读请求" << endl;
38      sem_wait(&rcsignal); // P
39      if (readerCount == 0) sem_wait(&filesrc);
40      readerCount++;
41      sem_post(&rcsignal); // V
42      cout << "读者线程" << num_thread << " 开始读文件"
            << endl;
43      usleep(duration * 1000000);
44      cout << "读者线程" << num_thread << " 读完了文件"
            << endl;
45      sem_wait(&rcsignal); // P
46      readerCount--;
47      if (readerCount == 0) sem_post(&filesrc);
48      sem_post(&rcsignal); // V
49      pthread_exit(NULL);
50  }
51
52  void *rp_threadWriter(void *p)
53  {
54      int num_thread = ((thread_info *)p)->thread_id;
55      double delaytime = ((thread_info
            *)p)->thread_delay;
56      double duration = ((thread_info
            *)p)->thread_lastTime;
57
58      usleep(delaytime * 1000000);
```

```
59      cout << "写者线程" << num_thread << "
            发送了一个写请求" << endl;
60      sem_wait(&filesrc);
61      cout << "写者线程" << num_thread << " 开始写文件
            " << endl;
62      usleep(duration * 1000000);
63      cout << "写者线程" << num_thread << " 写完了文件
            " << endl;
64      sem_post(&filesrc);
65      pthread_exit(NULL);
66  }
67
68  // 读者优先
69  void ReaderPriority()
70  {
71      int n_thread, i;
72      cout << "读者优先: " << endl;
73      cout << "输入计划处理的线程数: ";
74      cin >> n_thread;
75      sem_init(&rcsignal, 0, 1);
76      sem_init(&filesrc, 0, 1);
77      pthread_t threads[n_thread];
78      thread_info thread_info[n_thread];
79
80      cout <<
            "分别输入线程序号、线程类别、线程开始时间、线程读写操作时间"
            << endl;
81      for (i = 0; i < n_thread; i++)
82      {
83          cin >> thread_info[i].thread_id >>
              thread_info[i].thread_type >>
              thread_info[i].thread_delay >>
              thread_info[i].thread_lastTime;
```

```
84      }
85
86      for (i = 0; i < n_thread; i++)
87      {
88          if (thread_info[i].thread_type == 'r' ||
                thread_info[i].thread_type == 'R') {
89              pthread_create(&threads[i], NULL,
                    rp_threadReader, &thread_info[i]);
90          }
91          else
92          {
93              pthread_create(&threads[i], NULL,
                    rp_threadWriter, &thread_info[i]);
94          }
95      }
96
97      for (i = 0; i < n_thread; i++)
98      {
99          pthread_join(threads[i], NULL);
100     }
101
102     cout << "所有读者写者均已完成。" << endl;
103 }
104
105 // 写者优先
106 void *wp_threadReader(void *p)
107 {
108     unsigned int delaytime;
109     unsigned int duration;
110     int num_thread;
111
112     num_thread = ((thread_info*)(p))->thread_id;
```

```
113     delaytime = (unsigned
            int)(((thread_info*)(p))->thread_delay *
            1000000);
114     duration = (unsigned
            int)(((thread_info*)(p))->thread_lastTime *
            1000000);
115     usleep(delaytime);
116     cout << "读者线程 " << num_thread << "
            发送了一个读请求" << endl;
117     sem_wait(&READ_S);
118     sem_wait(&RCSIGNAL);
119     if (readerCount == 0)
120         sem_wait(&WRT);
121     readerCount++;
122     sem_post(&RCSIGNAL);
123     sem_post(&READ_S);
124
125     cout << "读者线程 " << num_thread << "
            开始读文件" << endl;
126     usleep(duration);
127     cout << "读者线程 " << num_thread << "
            读完了文件" << endl;
128     sem_wait(&RCSIGNAL);
129     readerCount--;
130     if (readerCount == 0)
131         sem_post(&WRT);
132     sem_post(&RCSIGNAL);
133     pthread_exit(NULL);
134 }
135
136 void *wp_threadWriter(void *p)
137 {
138     unsigned int delaytime;
```

```
139     unsigned int duration;
140     int num_thread;
141
142     num_thread = ((thread_info*)(p))->thread_id;
143     delaytime = (unsigned
            int)(((thread_info*)(p))->thread_delay *
            1000000);
144     duration = (unsigned
            int)(((thread_info*)(p))->thread_lastTime *
            1000000);
145     usleep(delaytime);
146     cout << "写者线程 " << num_thread << "
            发送了一个写请求" << endl;
147     sem_wait(&writeCountSignal);
148     if (writerCount == 0)
149         sem_wait(&READ_S);
150     writerCount++;
151     sem_post(&writeCountSignal);
152     sem_wait(&WRT);
153
154     cout << "写者线程 " << num_thread << "
            开始写文件" << endl;
155     usleep(duration);
156     cout << "写者线程 " << num_thread << "
            写完了文件" << endl;
157     sem_post(&WRT);
158     sem_wait(&writeCountSignal);
159     writerCount--;
160     if (writerCount == 0)
161         sem_post(&READ_S);
162     sem_post(&writeCountSignal);
163     pthread_exit(NULL);
164 }
```

```
165
166 void WriterPriority()
167 {
168     int n_thread = 0;
169     pthread_t h_Thread[64];
170     thread_info thread_info[64];
171     int readerCount = 0;
172     int writerCount = 0;
173
174     sem_init(&RCSIGNAL, 0, 1);
175     sem_init(&writeCountSignal, 0, 1);
176     sem_init(&WRT, 0, 1);
177     sem_init(&READ_S, 0, 1);
178     cout << "写者优先" << endl;
179     cout << "输入计划处理的线程数: ";
180     int j;
181     cin >> j;
182     cout <<
            "分别输入线程序号、线程类别、线程开始时间、线程读写操作时间"
           << endl;
183     while (n_thread < j)
184     {
185         int id1;
186         char type1;
187         double delay1, lastTime1;
188         cin >> id1 >> type1 >> delay1 >> lastTime1;
189         thread_info[n_thread].thread_id = id1;
190         thread_info[n_thread].thread_type = type1;
191         thread_info[n_thread].thread_delay = delay1;
192         thread_info[n_thread].thread_lastTime =
               lastTime1;
193         n_thread++;
194     }
```

```
195
196     for (int i = 0; i < n_thread; i++)
197     {
198         if (thread_info[i].thread_type == 'R' ||
                thread_info[i].thread_type == 'r')
199         {
200             pthread_create(&h_Thread[i], NULL,
                    wp_threadReader, &thread_info[i]);
201         }
202         else
203         {
204             pthread_create(&h_Thread[i], NULL,
                    wp_threadWriter, &thread_info[i]);
205         }
206     }
207
208     for (int i = 0; i < n_thread; i++)
209     {
210         pthread_join(h_Thread[i], NULL);
211     }
212     cout << "所有读者写者均已完成。" << endl;
213     sem_destroy(&RCSIGNAL);
214     sem_destroy(&writeCountSignal);
215     sem_destroy(&WRT);
216     sem_destroy(&READ_S);
217 }
218
219 int main() {
220     char choice;
221     while (true) {
222         cout << "请选择：" << endl;
223         cout << "1. 读者优先" << endl;
224         cout << "2. 写者优先" << endl;
```

```
225             cout << "3. 退出" << endl;
226             cout << endl;
227 
228             cin >> choice;
229             if (choice == '1') {
230 
231                 ReaderPriority();
232                 cout << "\n按任意键以继续" << endl;
233                 cin.get();
234             }
235             else if (choice == '2')
236             {
237                 WriterPriority();
238                 cout << "\n按任意键以继续" << endl;
239                 cin.get();
240             }
241             else
242             {
243                 return 0;
244             }
245         }
246 }
```

5 实验结果

```
[bbjsxl@VM-20-8-centos code5]$ make
g++ -std=c++11 test.cc -o test -lpthread
[bbjsxl@VM-20-8-centos code5]$ ./test
请选择：
1. 读者优先
2. 写者优先
3. 退出

1
读者优先：
输入计划处理的线程数：5
分别输入线程序号、线程类别、线程开始时间、线程读写操作时间
1    R    3    5
2    W    4    5
3    R    5    2
4    R    6    5
5    W    5.1    3
读者线程1 发送了一个读请求
读者线程1 开始读文件
写者线程2 发送了一个写请求
读者线程3 发送了一个读请求
读者线程3 开始读文件
写者线程5 发送了一个写请求
读者线程4 发送了一个读请求
读者线程4 开始读文件
读者线程3 读完了文件
读者线程1 读完了文件
读者线程4 读完了文件
写者线程2 开始写文件
写者线程2 写完了文件
写者线程5 开始写文件
写者线程5 写完了文件
所有读者写者均已完成。

按任意键以继续
请选择：
1. 读者优先
2. 写者优先
3. 退出

2
写者优先
输入计划处理的线程数：5
分别输入线程序号、线程类别、线程开始时间、线程读写操作时间
1    R    3    5
2    W    4    5
3    R    5    2
4    R    6    5
5    W    5.1    3
读者线程 1 发送了一个读请求
读者线程 1 开始读文件
写者线程 2 发送了一个写请求
读者线程 3 发送了一个读请求
写者线程 5 发送了一个写请求
读者线程 4 发送了一个读请求
读者线程 1 读完了文件
写者线程 2 开始写文件
写者线程 2 写完了文件
写者线程 5 开始写文件
写者线程 5 写完了文件
读者线程 3 开始读文件
读者线程 4 开始读文件
读者线程 3 读完了文件
读者线程 4 读完了文件
所有读者写者均已完成。

按任意键以继续
请选择：
1. 读者优先
2. 写者优先
3. 退出

3
[bbjsxl@VM-20-8-centos code5]$
```


图 2： 实验运行结果图