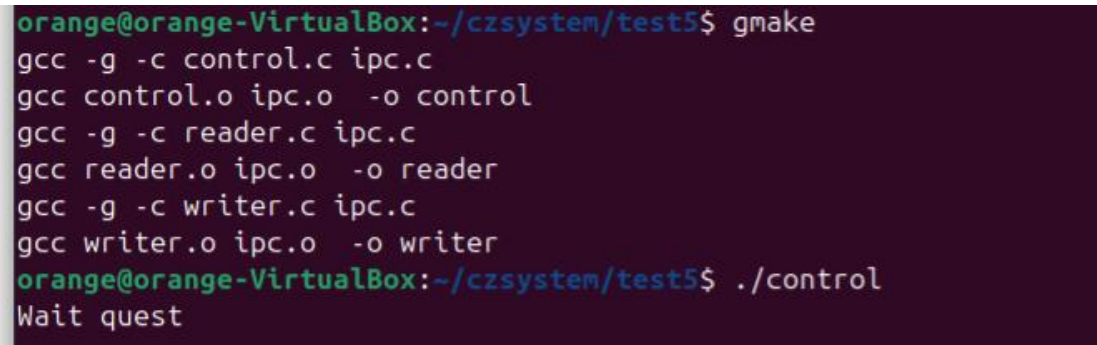
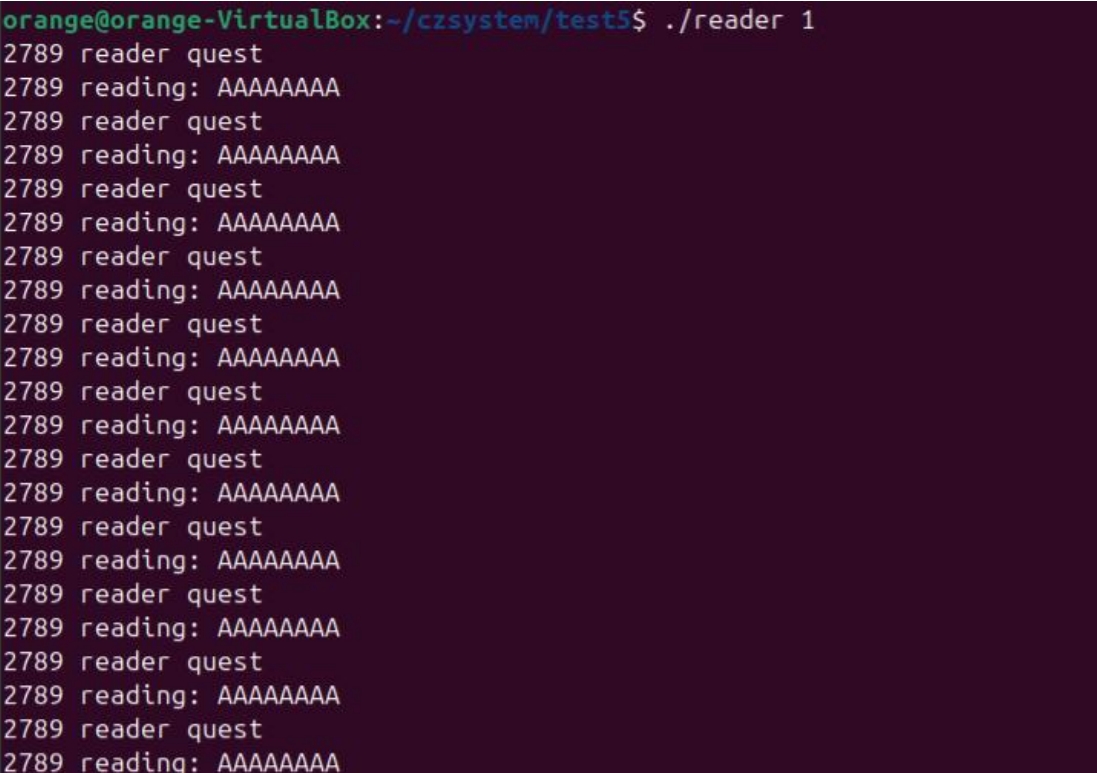


计算机学院 操作系统 课程实验报告

实验题目： 进程互斥		学号： 202200130048
日期： 11.6	班级： 6	姓名： 陈静雯
Email： 1205037094@qq. com		
实验步骤与现象：		
1. 示例实验		
 <pre>orange@orange-VirtualBox:~/czsystem/test5\$ gmake gcc -g -c control.c ipc.c gcc control.o ipc.o -o control gcc -g -c reader.c ipc.c gcc reader.o ipc.o -o reader gcc -g -c writer.c ipc.c gcc writer.o ipc.o -o writer orange@orange-VirtualBox:~/czsystem/test5\$./control Wait quest</pre>		
控制进程在等待请求		
 <pre>orange@orange-VirtualBox:~/czsystem/test5\$./reader 1 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA 2789 reader quest 2789 reading: AAAAAAAAA</pre>		
其中一个读者以 1 秒的延迟快一些读		

```
orange@orange-VirtualBox:~/czsystem/test5$ ./reader 10
2788 reader quest
2788 reading: AAAAAAAA
2788 reader quest
2788 reading: AAAAAAAA
```

一个让它以 10 秒的延迟慢一些读

```
orange@orange-VirtualBox:~/czsystem/test5$ ./control
Wait quest
2788 quest read
2789 quest read
2789 reader finished
2789 quest read
2789 quest read
2789 reader finished
2789 quest read
2789 reader finished
2788 reader finished
2788 quest read
2789 reader finished
2789 quest read
2789 quest read
2789 reader finished
2789 reader finished
2789 quest read
2789 reader finished
```

控制进程开始响应读者请求，让多个读者同时进入临界区读。再在另一终端窗体中启动一个延迟时间为 8 秒的写者进程，可以看到控制进程在最后一个读者读完后首先响应写者请求，在写者写完后两个读者也同时读到了新写入的内容。

(1) 与以上不同的启动顺序、不同的延迟时间，启动更多的读写者。可以看到它仍能满足我们要求的读写者问题的功能

(2) 为了制造读者或写者的饥饿现象，可以通过调整控制者进程的逻辑，使其偏向于处理某一类请求。例如，可以优先处理写者请求，导致读者长时间无法获得资源。

读者饥饿: 由于控制者进程优先处理写者请求，读者可能会长时间无法获得资源，导致读者饥饿现象。

写者饥饿: 如果控制者进程优先处理读者请求，写者可能会长时间无法获得资源，导致写者饥饿现象。

2. 独立实验

假设理发店的理发室中有 3 个理发椅子和 3 个理发师，有一个可容纳 4 个顾客坐等理发的沙发。此外还有一间等候室，可容纳 13 位顾客等候进入理发室。顾客如果发现理发店中顾客已满（超过 20 人），就不进入理发店。

在理发店内，理发师一旦有空就为坐在沙发上等待时间最长的顾客理发，同时空出的沙发让在等候室中等待时间最长的顾客就坐。顾客理完发后，可向任何一位理发师付款。但理发店只有一本现金登记册，在任一时刻只能记录一个顾客的付款。理发师在没有顾客的时候就坐在理发椅子上睡眠。理发师的时间就用在理发、收款、睡眠上。

```
orange@orange-VirtualBox: ~/czsystem/test5/barber$ ./barber
7号理发师睡眠
8号理发师睡眠
9号理发师睡眠
█
```

没有顾客时，理发师睡眠

```
orange@orange-VirtualBox: ~/czsystem/test5/... × orange@orange-VirtualBox: ~/czsystem/test5/...
orange@orange-VirtualBox: ~/czsystem/test5/barber$ ./customer
0号新顾客直接坐入理发椅
1号新顾客直接坐入理发椅
2号新顾客直接坐入理发椅
3号新顾客直接坐入理发椅
4号新顾客直接坐入理发椅
5号新顾客直接坐入理发椅
6号新顾客直接坐入理发椅
7号新顾客直接坐入理发椅
8号新顾客直接坐入理发椅
```

有顾客进入，理发师工作

```
orange@orange-VirtualBox: ~/czsystem/test5/barber$ ./barber
7号理发师睡眠
8号理发师睡眠
9号理发师睡眠
7号理发师为0号顾客理发
7号理发师收取0号顾客交费
7号理发师睡眠
8号理发师为1号顾客理发
8号理发师收取1号顾客交费
8号理发师睡眠
9号理发师为2号顾客理发
7号理发师为3号顾客理发
9号理发师收取2号顾客交费
9号理发师睡眠
8号理发师为4号顾客理发
7号理发师收取3号顾客交费
7号理发师睡眠
```

结论分析：

1. 对解决非对称性互斥操作的算法有哪些新的理解和认识？

(1) 优先级管理：

写者优先：在读写者问题中，写者通常具有更高的优先级，因为写操作会改变共享资源的状态，而读操作只是查看状态。这种优先级管理可以减少数据不一致的风险。

读者优先：在某些场景下，读者的数量远多于写者，此时可以优先处理读者请求，以提高系统的吞吐量。

(2) 饥饿预防：

轮询机制：通过引入轮询机制，确保每个进程都有机会获得资源，防止某个进程长期被忽略。

超时机制：设置超时时间，如果某个进程等待时间过长，则强制中断当前操作，给予其他进程机会。

(3) 公平性：

公平调度：确保所有进程按照某种公平的原则（如 FIFO）获得资源，避免某个进程因优先级高而一直占用资源。

(4) 优先级继承：当一个低优先级进程持有锁，而高优先级进程等待该锁时，临时提升低优先级进程的优先级，以加快锁的释放。

2. 为什么会出现进程饥饿现象？

(1) 优先级倒置：高优先级进程等待低优先级进程持有的资源，导致高优先级进程被阻塞。

(2) 资源竞争：多个进程竞争同一资源，导致某些进程长时间无法获得资源。

(3) 调度策略：不合理的调度策略可能导致某些进程被无限期地推迟。

3. 本实验的饥饿现象是怎样表现的？

(1) 读者饥饿：控制者进程优先处理写者请求，导致读者长时间无法获得读取共享内存的机会。

(2) 写者饥饿：如果读者数量过多且频繁请求，写者可能长时间无法获得写入共享内存的机会。

4. 怎样解决并发进程间发生的饥饿现象？

(1) 轮询机制：在控制者进程中引入轮询机制，定期检查所有请求队列，确保每个进程都有机会获得资源。

(2) 超时机制：设置超时时间，如果某个进程等待时间过长，则强制中断当前操作，给予其他进程机会。

5. 对于并发进程间使用消息传递解决进程通信问题有哪些新的理解和认识？

(1) 解耦合：消息传递机制使得进程之间松耦合，每个进程只需关注自己的任务，通过消息队列进行通信，提高了系统的灵活性和可维护性。

(2) 异步通信：消息传递支持异步通信，进程可以在发送消息后继续执行其他任务，无需等待接收方的响应，提高了系统的并发性能。

(3) 资源管理：消息传递机制可以帮助管理系统资源，通过消息队列可以控制资源的访问顺序和频率，避免资源竞争和死锁问题。

(4) 错误处理：消息传递机制提供了丰富的错误处理机制，可以通过消息类型

和内容传递错误信息，便于调试和故障排查