Lab 4 实验报告

181250068 李淳

邮箱：181250068@smail.nju.edu.cn

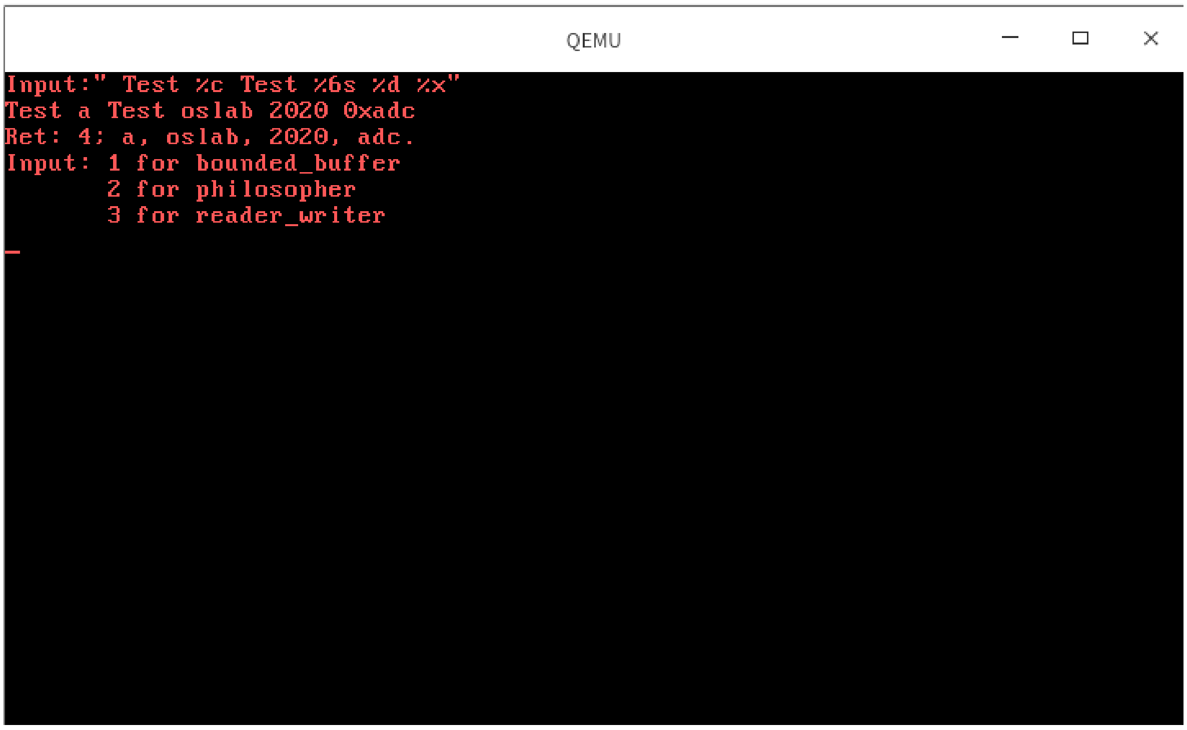
QQ: 936637810

实验进度

我完成了格式化输入函数，进程通信，信号量相关系统调用，进程同步问题

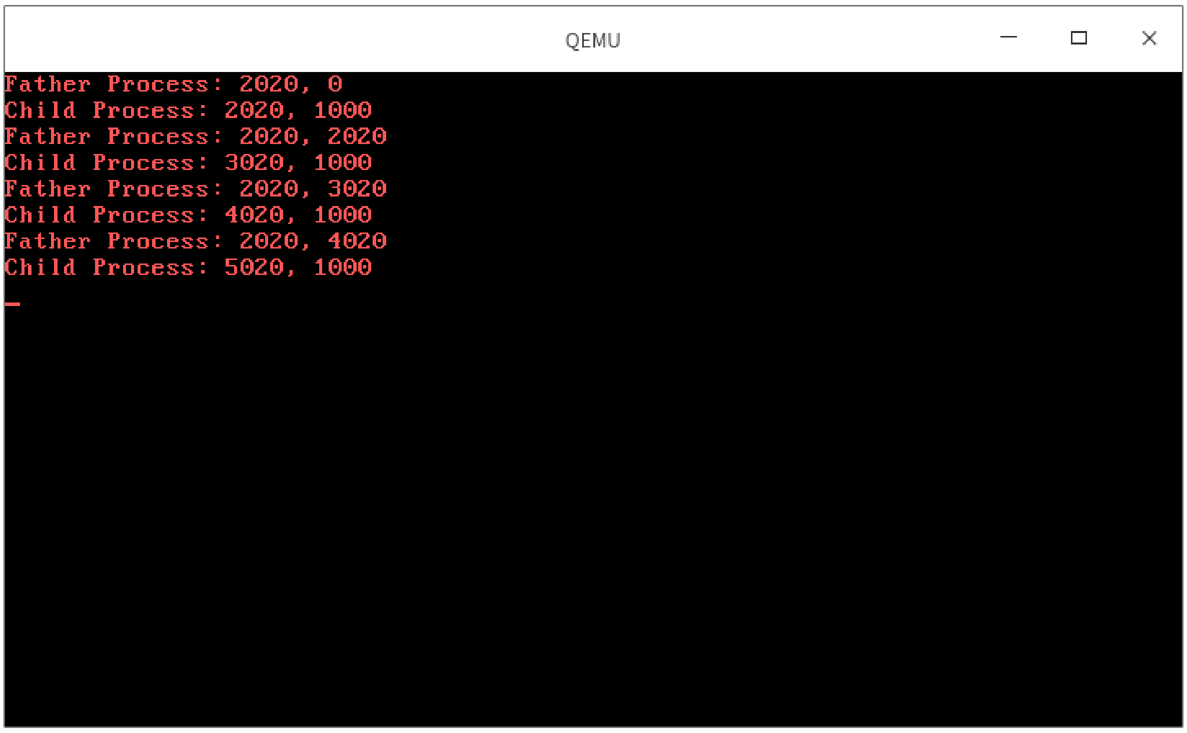
实验结果

格式化输入函数



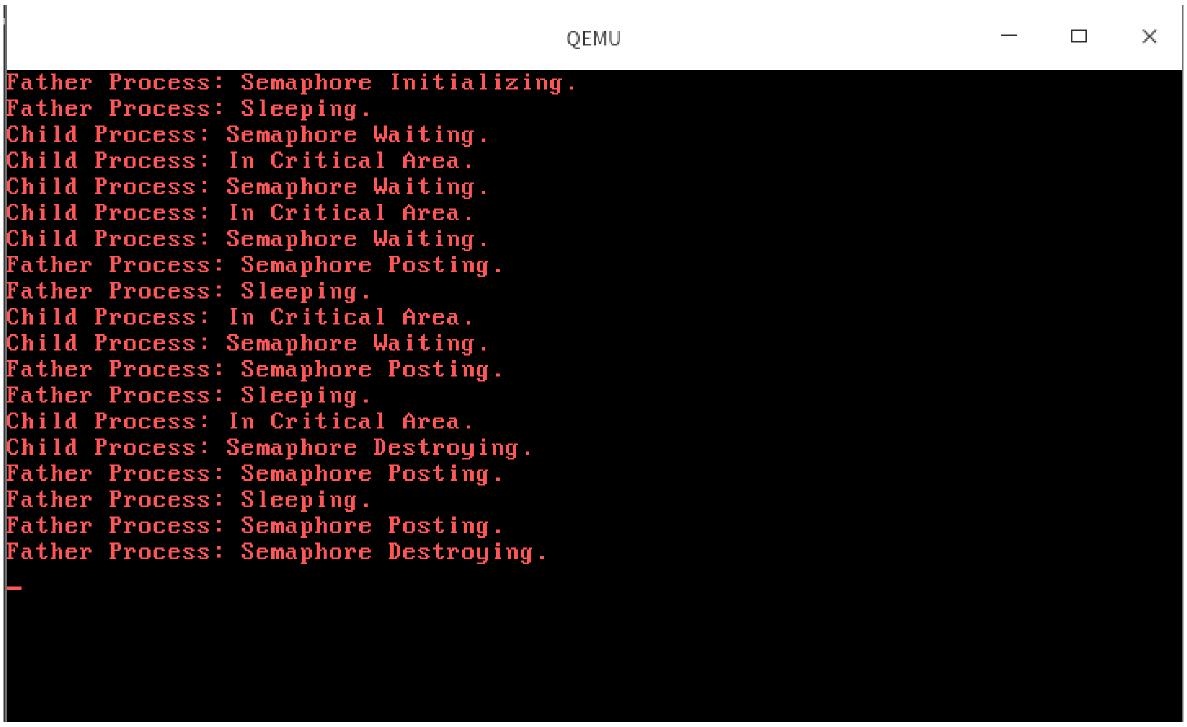
```
Input:" Test %c Test %6s %d %x"
Test a Test oslab 2020 0xadc
Ret: 4; a, oslab, 2020, adc.
Input: 1 for bounded_buffer
       2 for philosopher
       3 for reader_writer

_
```

进程通信



```
Father Process: 2020, 0
Child Process: 2020, 1000
Father Process: 2020, 2020
Child Process: 3020, 1000
Father Process: 2020, 3020
Child Process: 4020, 1000
Father Process: 2020, 4020
Child Process: 5020, 1000
_
```

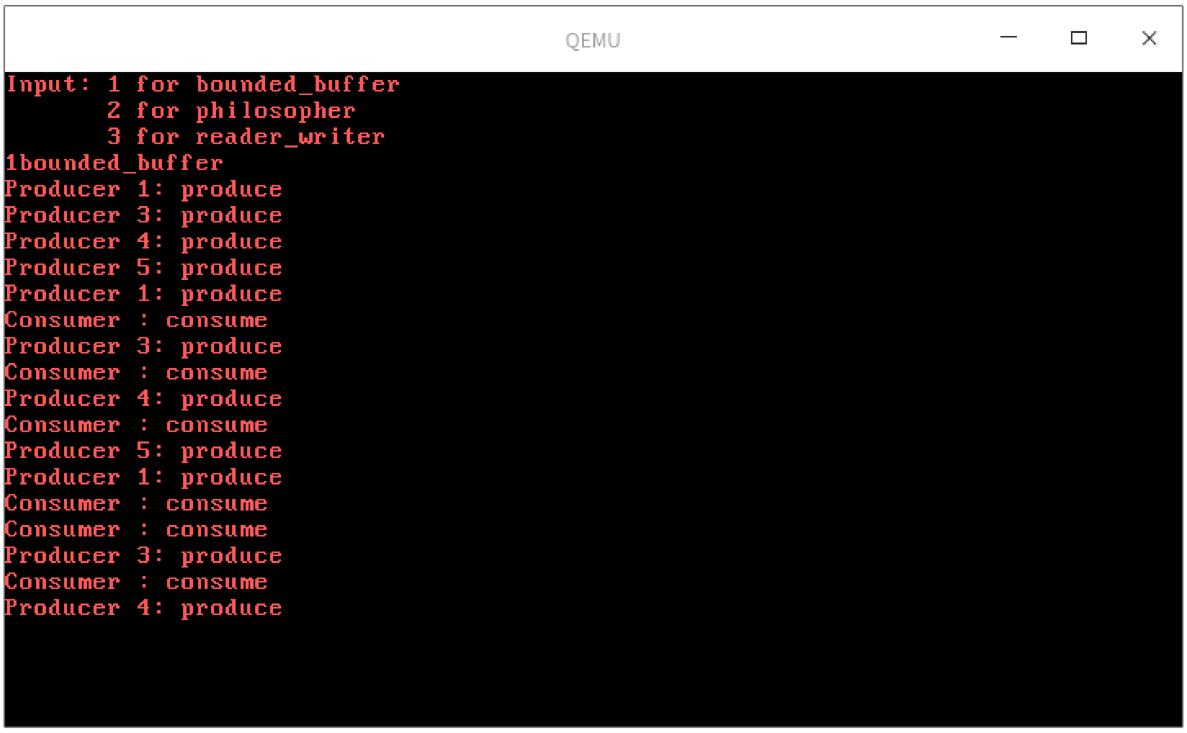
信号量相关系统调用



```
Father Process: Semaphore Initializing.
Father Process: Sleeping.
Child Process: Semaphore Waiting.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Father Process: Semaphore Posting.
Father Process: Sleeping.
Child Process: In Critical Area.
Child Process: Semaphore Waiting.
Father Process: Semaphore Posting.
Father Process: Sleeping.
Child Process: In Critical Area.
Child Process: Semaphore Destroying.
Father Process: Semaphore Posting.
Father Process: Sleeping.
Father Process: Semaphore Posting.
Father Process: Semaphore Destroying.
_
```

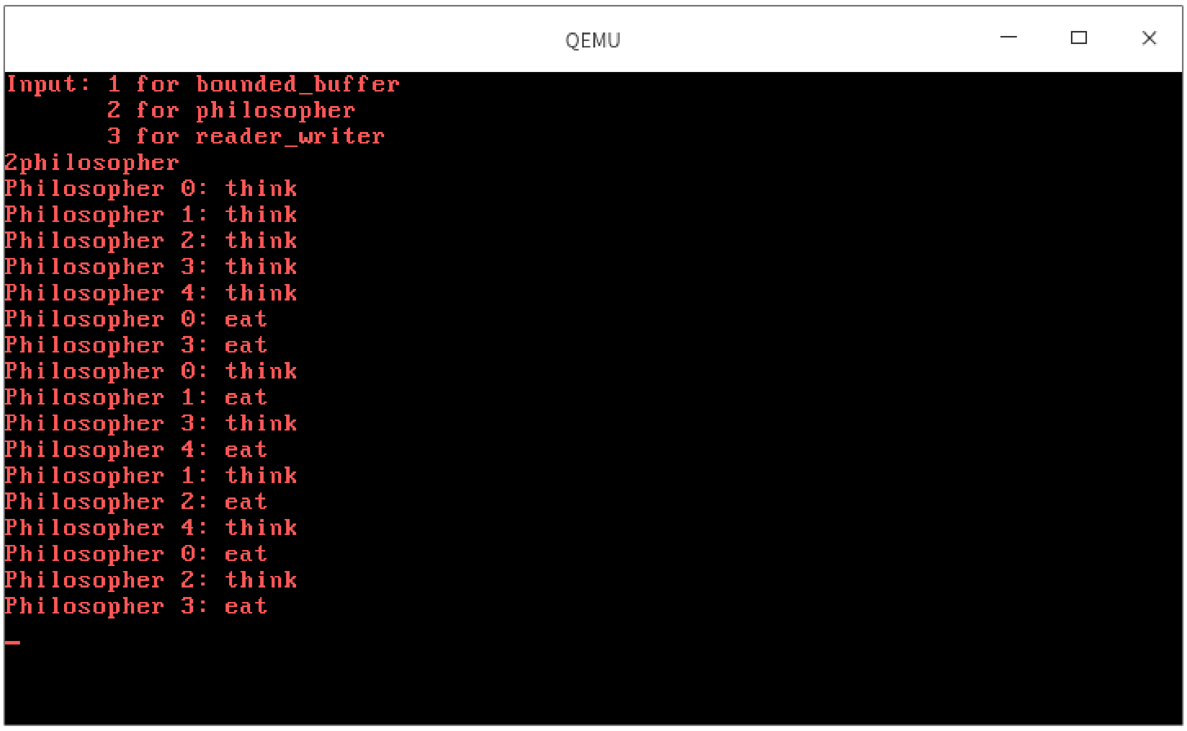
进程同步问题

生产者消费者问题



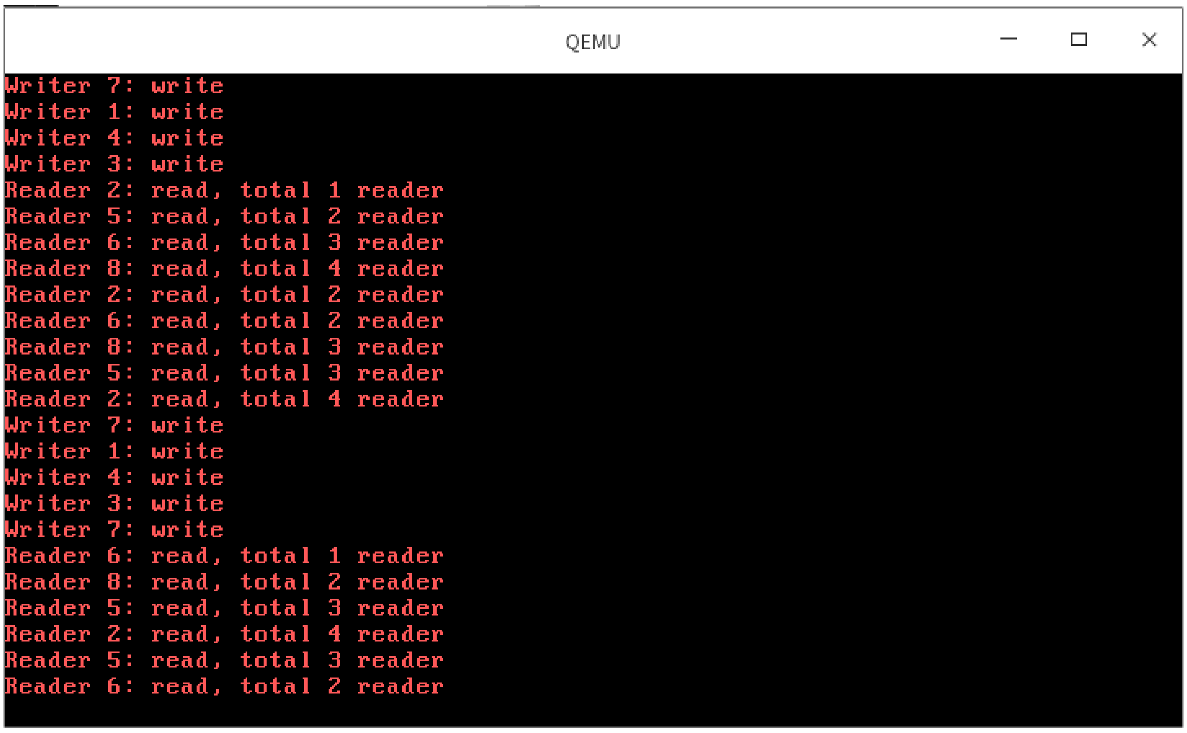
```
Input: 1 for bounded_buffer
       2 for philosopher
       3 for reader_writer
1bounded_buffer
Producer 1: produce
Producer 3: produce
Producer 4: produce
Producer 5: produce
Producer 1: produce
Consumer : consume
Producer 3: produce
Consumer : consume
Producer 4: produce
Consumer : consume
Producer 5: produce
Producer 1: produce
Consumer : consume
Consumer : consume
Producer 3: produce
Consumer : consume
Producer 4: produce
```

哲学家就餐问题



```
Input: 1 for bounded_buffer
       2 for philosopher
       3 for reader_writer
2philosopher
Philosopher 0: think
Philosopher 1: think
Philosopher 2: think
Philosopher 3: think
Philosopher 4: think
Philosopher 0: eat
Philosopher 3: eat
Philosopher 0: think
Philosopher 1: eat
Philosopher 3: think
Philosopher 4: eat
Philosopher 1: think
Philosopher 2: eat
Philosopher 4: think
Philosopher 0: eat
Philosopher 2: think
Philosopher 3: eat
_
```

读者写者问题



```
Writer 7: write
Writer 1: write
Writer 4: write
Writer 3: write
Reader 2: read, total 1 reader
Reader 5: read, total 2 reader
Reader 6: read, total 3 reader
Reader 8: read, total 4 reader
Reader 2: read, total 2 reader
Reader 6: read, total 2 reader
Reader 8: read, total 3 reader
Reader 5: read, total 3 reader
Reader 2: read, total 4 reader
Writer 7: write
Writer 1: write
Writer 4: write
Writer 3: write
Writer 7: write
Reader 6: read, total 1 reader
Reader 8: read, total 2 reader
Reader 5: read, total 3 reader
Reader 2: read, total 4 reader
Reader 5: read, total 3 reader
Reader 6: read, total 2 reader
```

实验修改的代码

所有标记了TODO in lab4的方法

实验中遇到的问题

不知道怎么实现随机函数

因为随机函数跟时间可能是相关的，所以在irqHandle中直接用inbyte把TIME_PORT的数据写到了eax里面，发现确实是具有随机性的数据

读者写者问题不知道怎么在多个进程之间同步数据

读者需要同步当前正在读的人数，本来以为是可以不用管道的，但是想了一下好像只有管道可以解决这个问题。

生产者消费者问题中的PV顺序有影响吗？

有，我们把代码改写成这样

```
BoundedBuffer::Deposit(c){
    emptyBuffers->P();
    mutex->P();
    Add c to the buffer;
    mutex->V();
    fullBuffers->V();
}
BoundedBuffer::Remove(c){
    mutex->P();
    fullBuffers->P();
    Remove c from buffer;
    emptyBuffers->V();
    mutex->V();
14  }
```

这样，如果消费者先执行，并且在第10行被挂起，然后进程切换到生产者，然后在第三行被挂起，这样就会产生死锁，因为生产者需要mutex资源，而消费者需要fullBuffers资源，形成了一个资源等待链

有没有更好的方式处理哲学家就餐问题？

暂时没想到