Oslab4 进程同步实验报告

学号：171860659 姓名：吴紫航 联系方式：[401986905@qq.com](mailto:401986905@qq.com)

**一．实验目的**

1.学习基于信号量的进程同步机制

2.加深对生产者消费者问题的理解

3.加深对系统调用和框架代码内存分配的理解

1. **实验设计**
2. 需要完成的任务：
3. 实现sem\_post,sem\_wait,sem\_destroy系统调用，通过框架代码测试用例
4. 修改框架代码使其可以支持8个进程
5. 完成用户态的生产者消费者问题（需要实现getpid系统调用）
6. 代码思路

（1-1）

（1）syscallSemPost:

相当于V操作，先利用传入的sf得到信号量号，如果这个信号量的位置不是use状态则操作失败。

否则，将信号量与0比较

如果小于0，则自增后要释放双向链表上的进程

如果大于等于0，则自增即可，不用释放任何进程

注：这里双向链表的结构比较特殊，值得一提的是如何从双向链表的节点地址得到对应被释放进程的pcb的地址呢？

只要根据等式 被释放进程的（pid地址-blocked地址）==当前进程的（pid地址-blocked地址）

由于当前进程的pid与blocked是可以访问的，被释放进程的blocked通过信号量的链表节点可以查询到，因此，自然得到了被释放进程的pid，即pcb数组的第pid项。更改其state为runable

并在返回前更改父子进程返回值即eax

（2）syscallSemWait

相当于P操作，先利用传入的sf得到信号量号，如果这个信号量的位置不是use状态则操作失败。

否则，将当前信号量自减，如果自减后小于0，则阻塞当前进程，并把它加入信号量的双向链表中，然后切换进程。切换进程使用 asm volatile(“int $0x20”);即可

（3）SyscallSemDestroy

未定义的操作。先利用传入的sf得到信号量号，如果这个信号量的位置不是use状态则操作失败。

否则只要设置信号量状态为未使用（0）即可

（1-2）

修改memoory.h的NR\_SEGMENTS为18即可，因为原本支持4个进程，NR\_SEGMENT原本为10，先要同时进行8个进程，则要增加4个进程，即4\*2=8个段（data、code），因此10增加到18即可

（1-3）

用户态模拟生产者消费者程序，需要两个信号量：full、mutex

Full限制消费者一定要在有产品的时候才能消费，实现同步，初始化为0

Mutex为共享区的锁，实现互斥，起始为1。

父进程fork出6个子进程

pid小的两个（2和3）作为生产者

pid大的四个（4、5、6、7）作为消费者

getpid只需要增加一条系统调用路线即可

函数中调用syscall，定义宏SYS\_PID,在syscallHandle中的switch中补充一条syscallPid的系统调用，将当前进程pid赋值给eax即可返回

将pid和信号量传给对应produce和consume函数。

两个生产者各自Produce 8个产品，

四个消费者各自consume 4个产品

生产者步骤为：try lock，调用sem\_wait(mutex)锁共享区，locked，生产产品，调用sem\_post(full)产生产品，调用sem\_post(mutex)释放锁，unlock

消费者步骤为：try consume,调用sem\_wait(full)查看是否有商品，try lock，调用sem\_wait(mutex)锁共享区，locked，完成消费consumed，调用sem\_post(mutex)释放锁，unlock

注1：关于1-1和1-3测试代码的切换，定义了一个宏#define MY\_TEST 1，使用了条件编译ifndef，else，endif结构，如果定义了MY\_TEST则运行1-3，如注释了MY\_TEST的定义则运行1-1





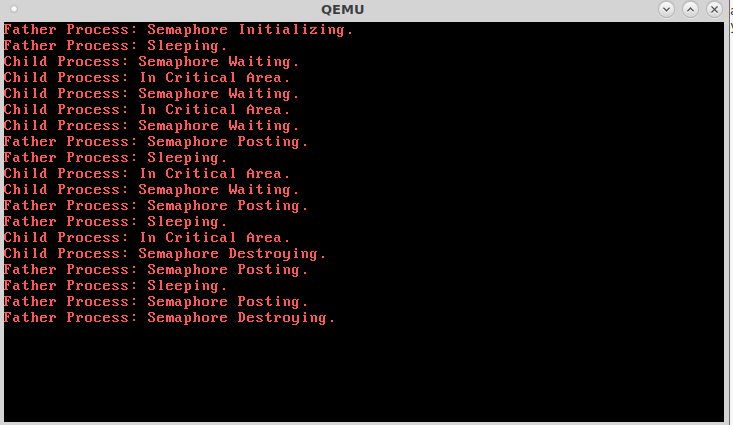
 

注2：为方便核对实验1-3阶段结果，将输出中，在输出到qemu同时，内容输出到串口，因此终端可以方便的查看到输出！

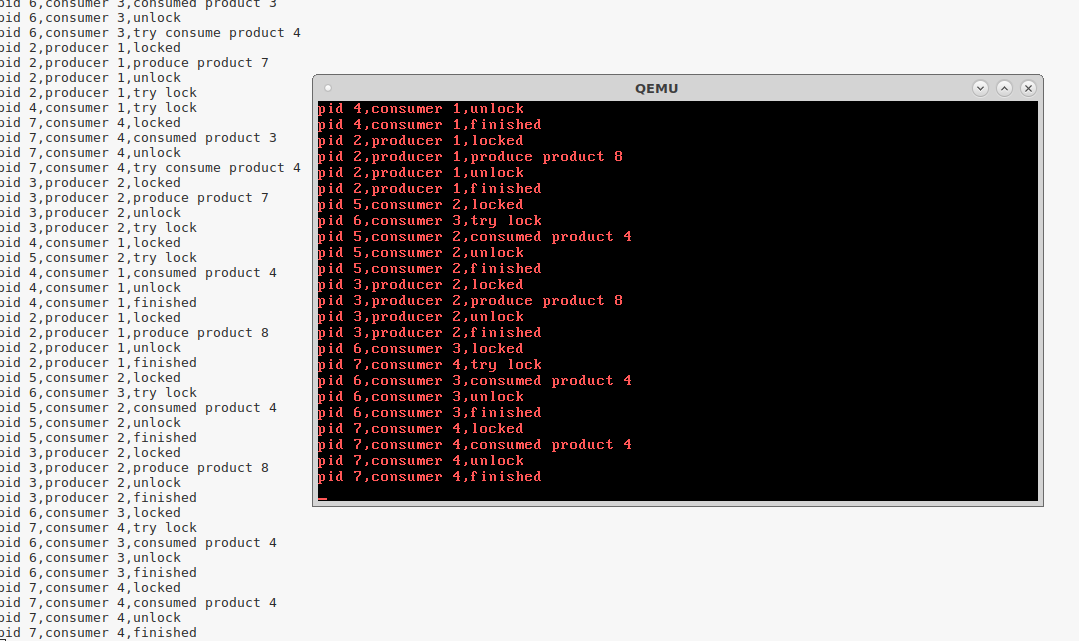
注3：在produce和consume的过程中，为突出pv操作的作用，用户函数每运行几行就会调用sleep切换进程，可以更加清晰的感受同步、互斥的机制！

1. 实验结果

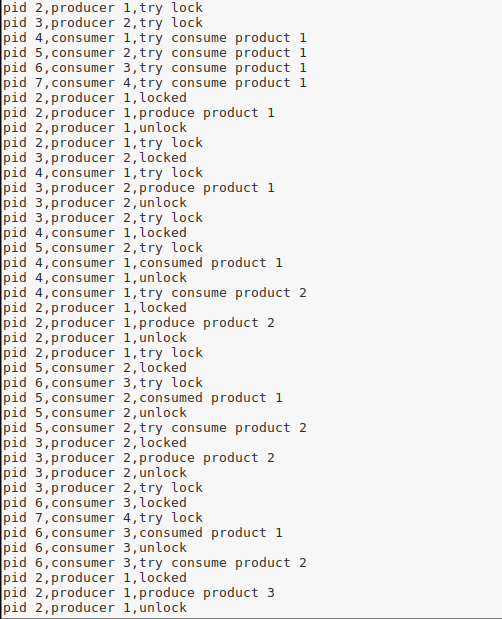
（1-1）

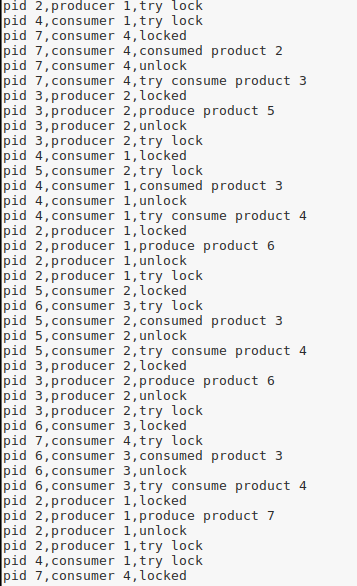


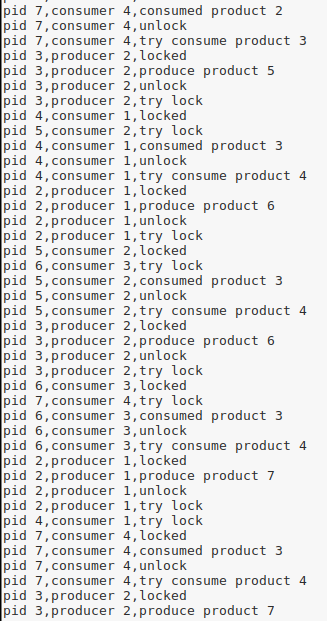
（1-3）qemu输出

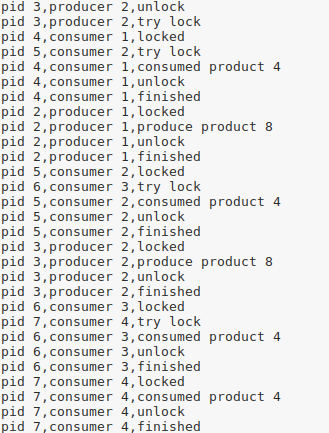


（1-3）串口输出









1. 实验收获

加深了对同步、互斥机制、共享区的理解

1. 实验建议

暂无