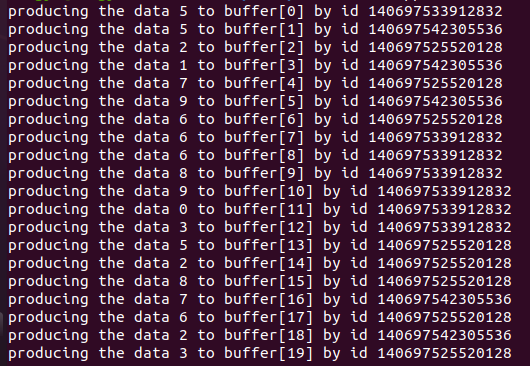
**操作系统大作业**

**18364057 林文龙**

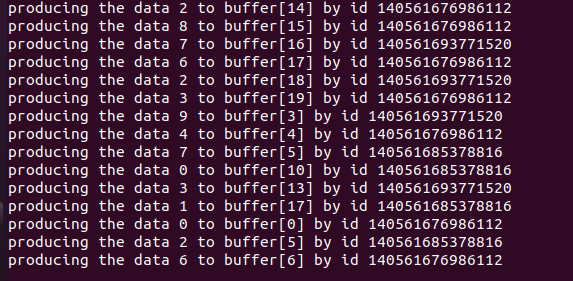
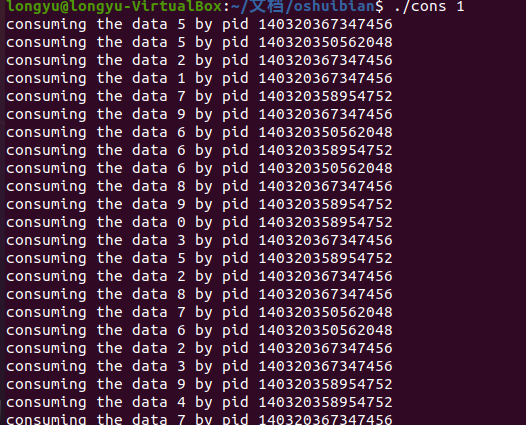
1. **生产者消费者问题**

**命令行 ./prod 1**



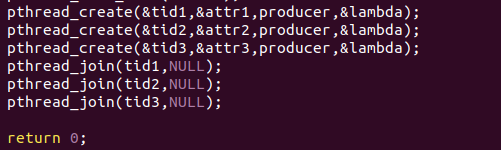
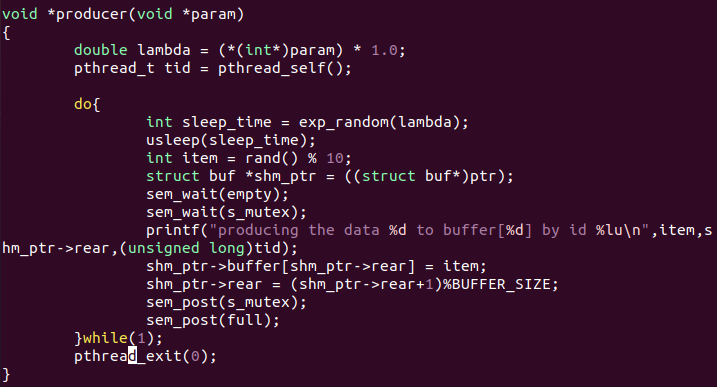
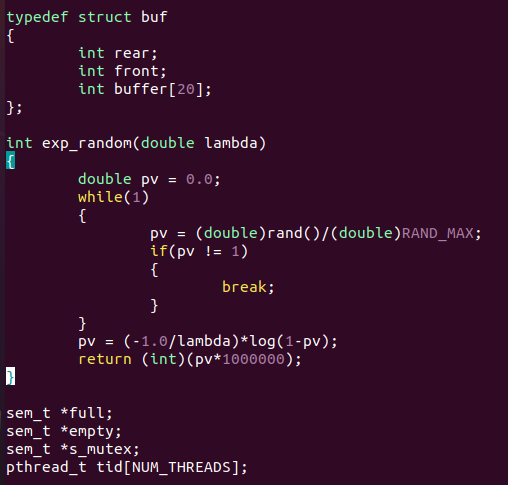
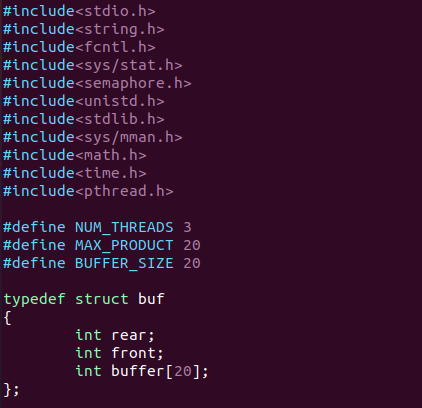
没有消费，生产者产满进程堵塞

命令 ./prod 1 ./cons 1

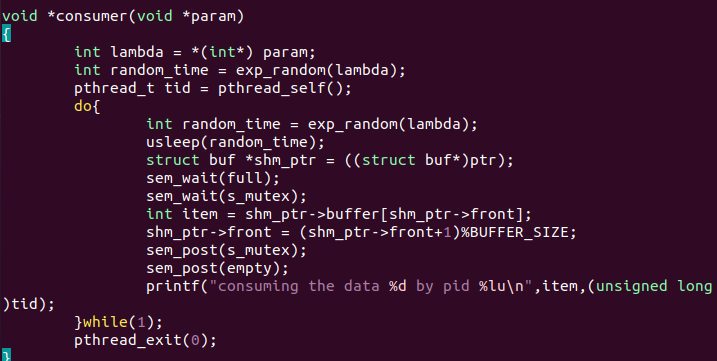
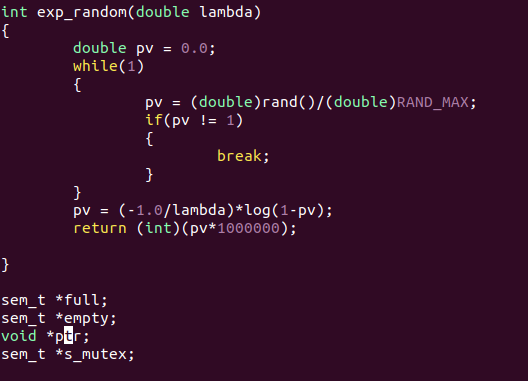
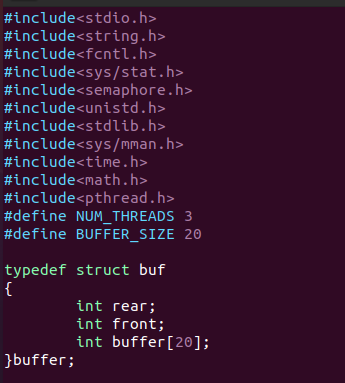


进程不在阻塞，正常运行

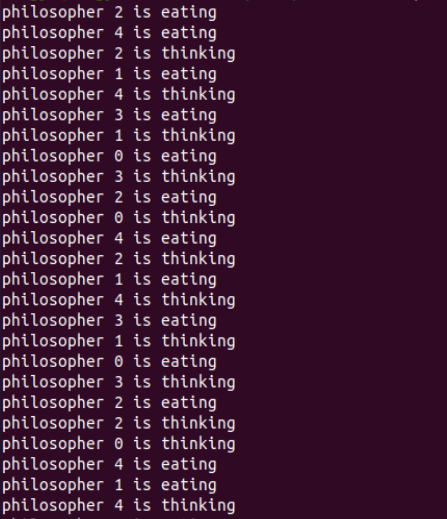
生产者代码：



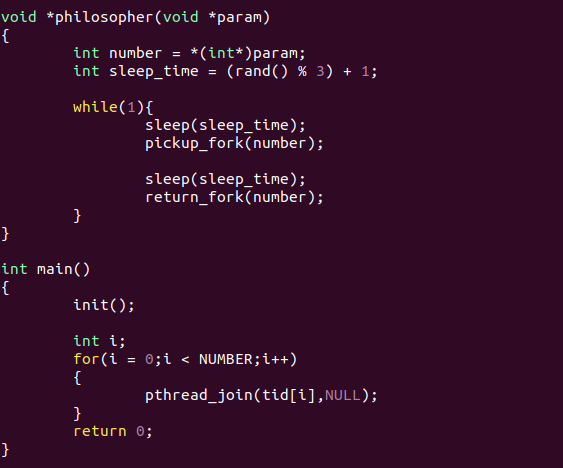
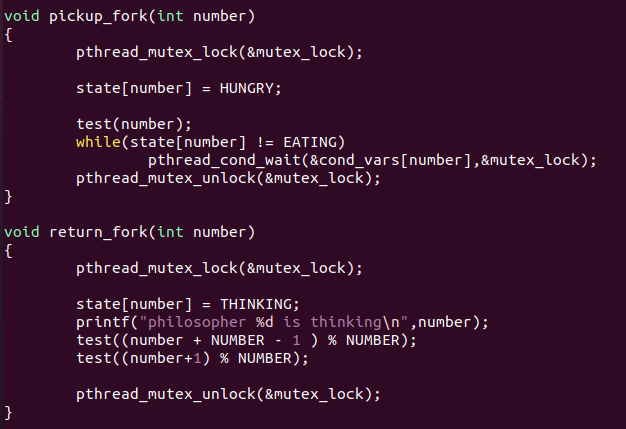
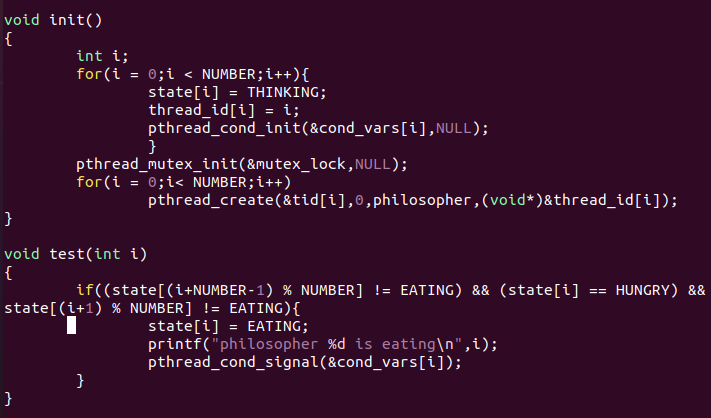
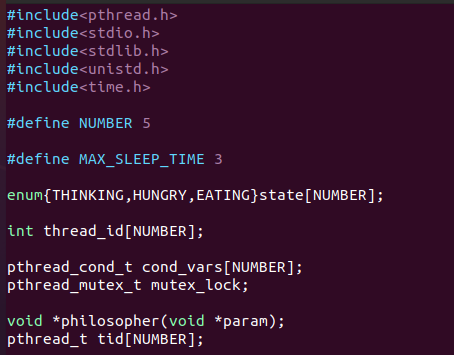
消费者代码：



1. 哲学家问题

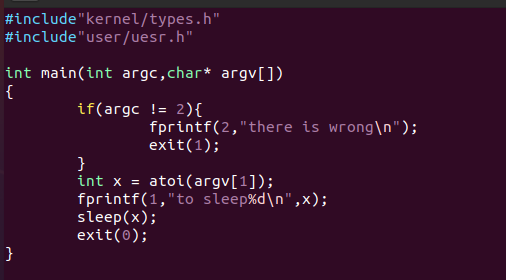
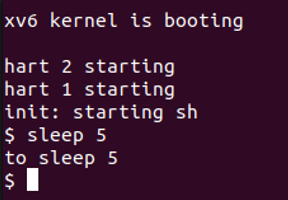


代码:

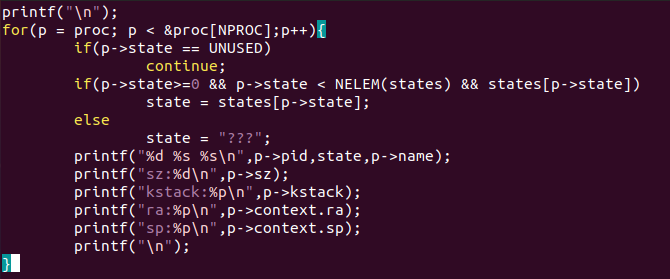


3.sleep

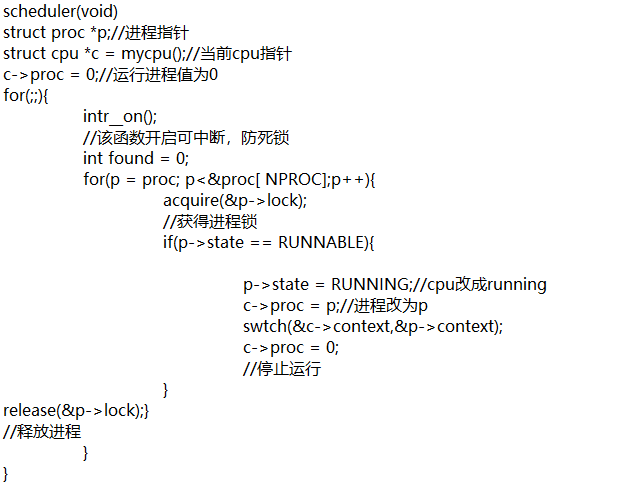
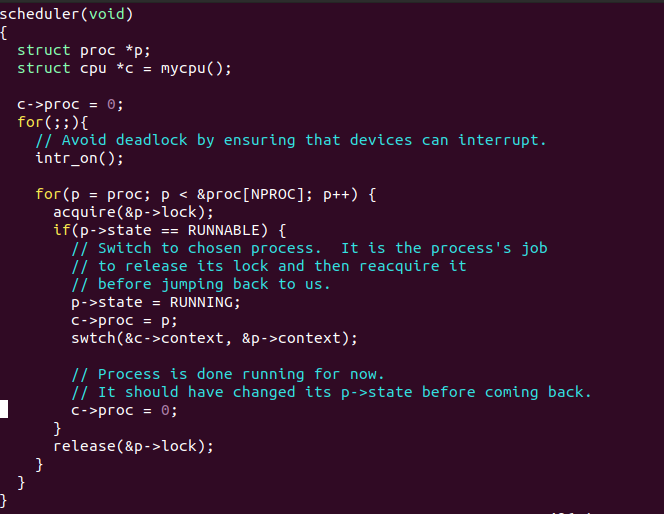
运行结果：

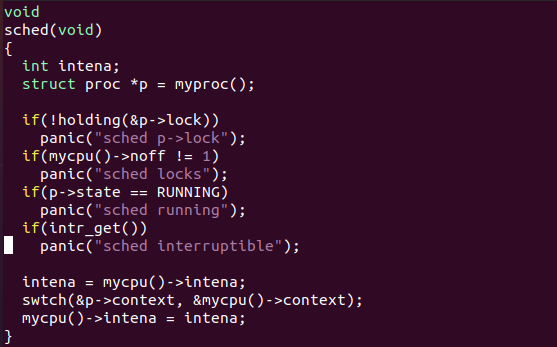


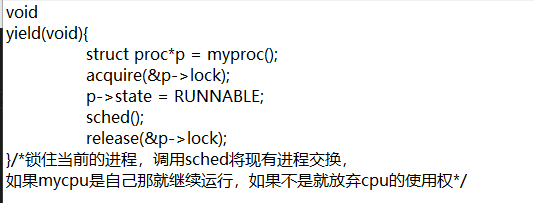
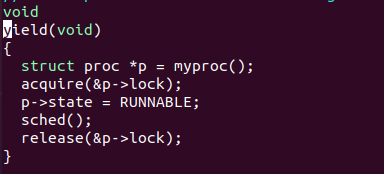
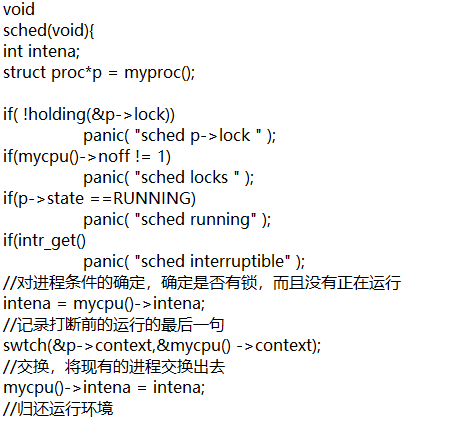
2.（a）



（b）







（c）

Xv6算法比较原始，是通过不断的遍历，循环实现的每个CPU初始化完毕后就载入scheduler函数开始进程调度，scheduler函数是个死循环，不停的通过内循环遍历所有进程状态，由于遍历的过程中可能会改变进程状态，所以，一次遍历的内循环外需要加锁保护，scheduler()内循环执行了一遍。然后释放进程列表的锁，这是考虑到没有进程待执行，但另外的CPU需要进行进程切换。如果不释放进程列表所，另外的CPU就没有进行进程切换，导致CPU空转锁死。然后开始新一次的外循环，外循环开始的时候会开起中断，是为了防止没有进程待执行，但有进程在IO阻塞的时候，关闭中断使得进程永远都不会唤醒，从而导致CPU中断锁死。

修改：

将scheduler中

**if(setjmp(&cpus[cpu()].jmpbuf) == 0)**

**longjmp(&p->jmpbuf);**

改为：

**cprintf("setjmp called in scheduler\n");**

**if(setjmp(&cpus[cpu()].jmpbuf) == 0){**

**cprintf("setjmp in scheduler returned 0; longjmp\n");**

**longjmp(&p->jmpbuf);**

**}else**

**cprintf("setjmp in scheduler returned 1\n");**