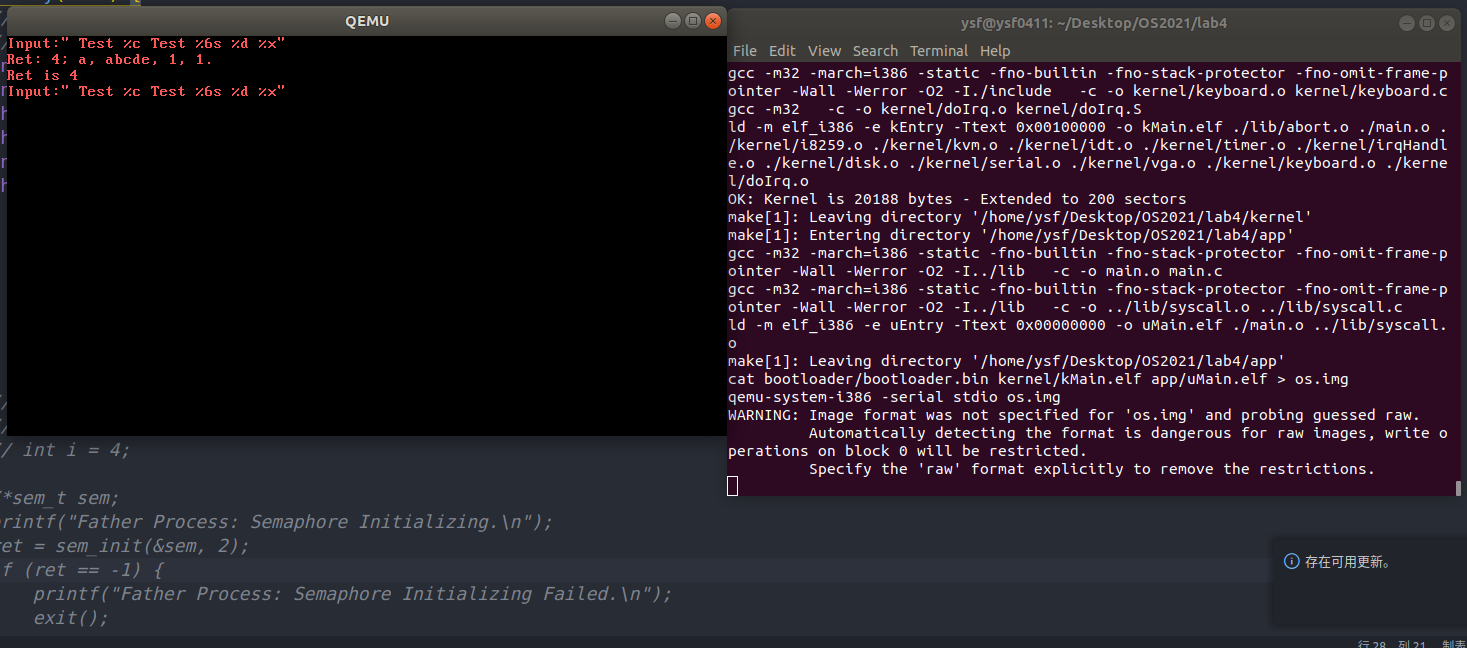
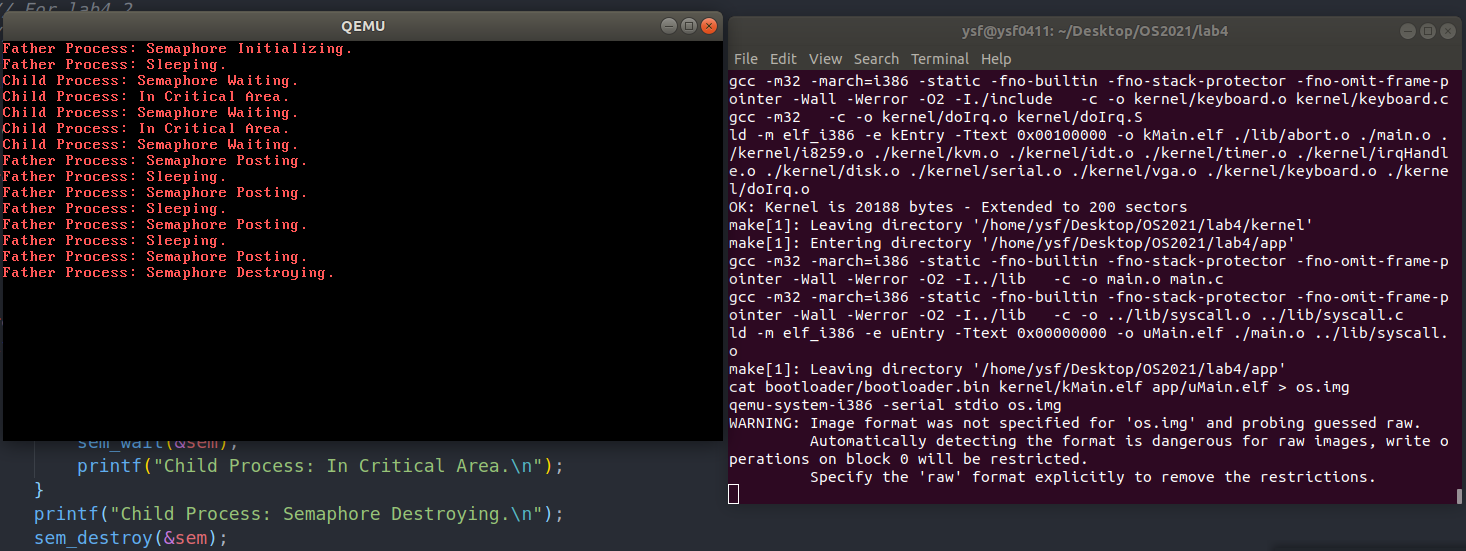
**Lab4实验报告**

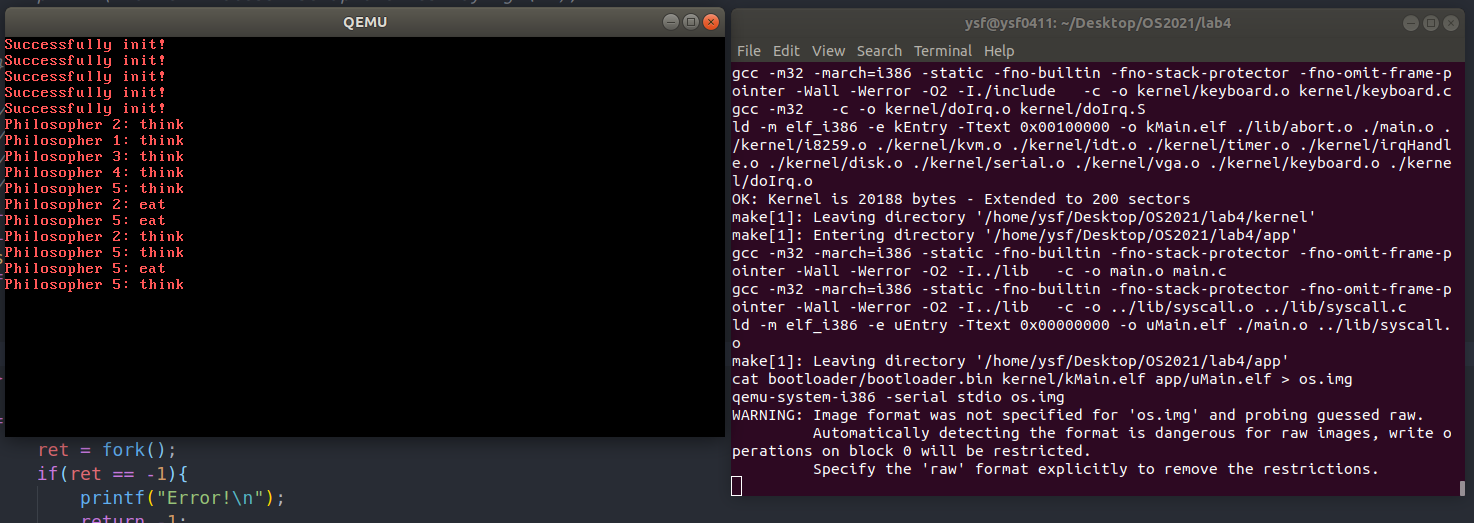
191300073 杨斯凡 191300073@smail.nju.edu.cn

1. 实验进度

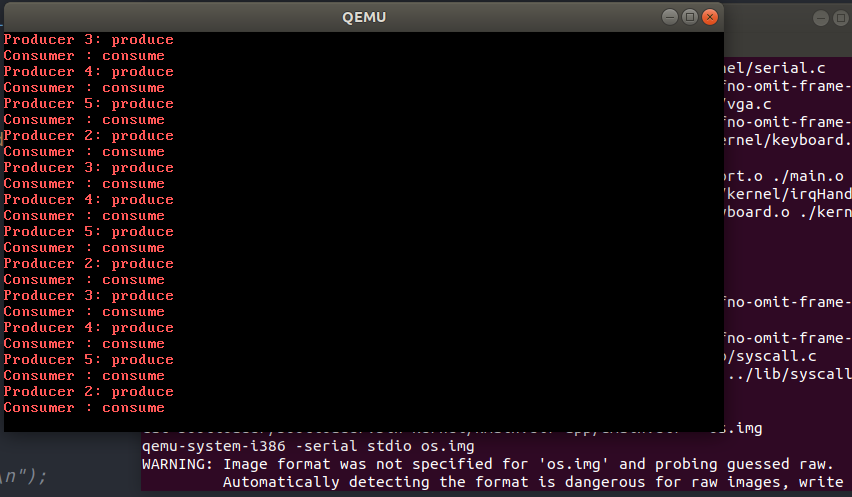
我完成了所有必做内容, 选做内容完成了生产者消费者问题.

1. 实验结果

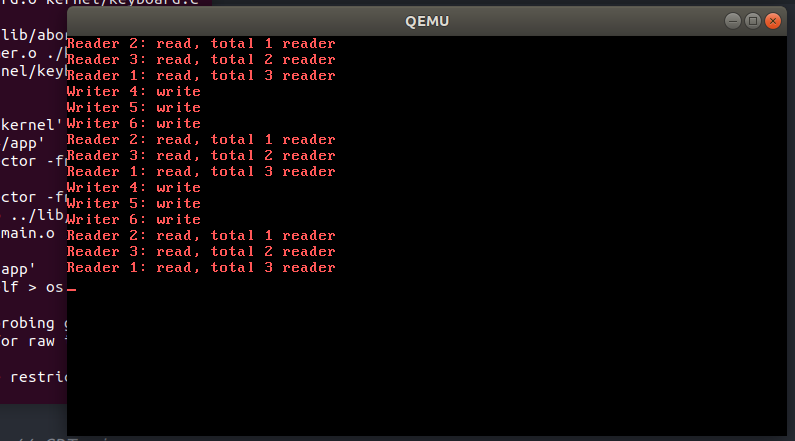
(scanf测试)

(信号量测试)

(哲学家问题)



生产者消费者问题



读者写者问题

1. 代码介绍

这次的代码框架和lab3几乎相同，在lab3代码框架的基础上增加了对信号量的一些操

作。并且为了支持哲学家问题，我将进程数和信号量增加到7个。

信号量在内核之中定义，因此每次对信号量操作的时候，必定需要进入系统调用。跳转到doirq.S中的irqSyscall，然后跳转到irqHandle.c中的irqHandle函数，检查中断号发现是系统调用，于是调用其处理函数syscallHandle，然后根据寄存器eax的值判断系统调用类型，进一步使用相应的处理函数。

而为了得到当前进程号，还需要增加一个系统调用SYS\_PID，以及相关的处理函数。

在创建信号量的时候需要对进行初始化，赋予其资源的数量。首先遍历信号量数组，找到一个空闲的位置。信号量初始化的值在edx寄存器中。初始化的时候，由于没有任何进程需要等待该信号量对应的资源，因此，挂起链表的next和prev都指向自己。然后将对应的索引保存在eax中。下一次使用信号量即可根据索引来找到该信号量。

然后是信号量的P操作。该操作需要申请一个资源。如果此时该资源的值小于0，则需要等待，于是阻塞该进程。即把该进程挂起在信号量的链表上，然后调用int$0x20指令切换进程。为了和sleep区分，把此时sleeptime设置为-1。

1. 修改代码位置

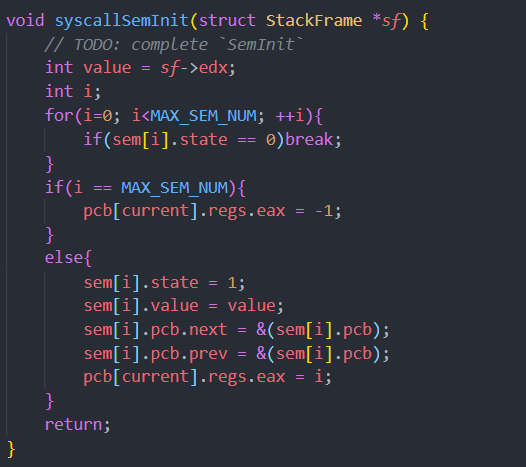
首先要完成scanf函数，也就是syscallReadStdIn函数，它完成的工作是. 如果dev[STD\_IN].value == 0，将当前进程阻塞在dev[STD\_IN]上。进程被唤醒，读取keyBuffer中的字符。

因此我首先模仿教程中的阻塞代码，将当前进程阻塞，然后使用了“int $0x20”进行进程切换，然后去读取字符缓冲区上的字符，把这些字符传给用户进程，并且把当前进程的eax寄存器的值设置为字符数量，作为返回值。



然后是实现信号量的函数，主要是syscallSemInit、syscallSemWait、syscallSemPost和syscallSemDestroy四个函数。

对于syscallSemInit函数，只需要在信号量数组中找到一个空闲的位置，对这个位置上的信号量进行初始化，并且把信号量数组的索引传给当前进程的eax作为返回值，以便于后序对这个信号量的调用。



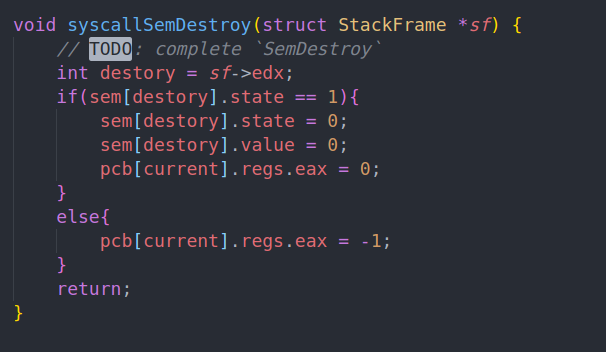
对于syscallSemWait，只需要对给定的信号量的value进行减一，如果大于等于0直接返回即可，如果小于0，需要挂起当前进程，然后使用“int $0x20”进行进程的切换，并且返回0表示操作成功。



对于syscallSemWait，首先需要对给定的信号量加一，如果大于零直接返回即可，如果小于0，需要把当前信号量上的挂起的进程从队列中移除，并且设为就绪态即可，并且返回0表示操作成功。



对于sysSemDestory，只需要把当前的信号量的state设为0，value设为0，返回0即可。



对于哲学家问题，按照教程上的伪代码完成即可，只需要每次fork出子进程即可。实现getpid函数，只需要仿照其他的系统调用，处理函数为syscallGetPid，只需要将eax的值设为current即可。

但是，有一个问题，就是框架代码仅仅支持4个进程和4个信号量同时工作，但是哲学家问题有5个进程和信号量，因此需要对框架代码进行修改，我首先将信号量的数目设为了8，然后将进程的数目设为了9，修改进程数目只需要修改GDT size即可。

1. bug及其解决

在实现读者写者问题的时候，我发现每次都会中途停下，在我进行了很久的调试之后，

发现是count的值出现了问题，然后我分析原因，发现是count在每个进程之间独立，每次进行修改的时候，只对本进程的count进行了修改，其他进程的count还是0，这就导致了bug的出现。

于是我在内核中定义了一个变量memory，作为count，然后每次更改的之后将更改的值赋予memory，然后读取的时候读取memory作为count的值进行操作，这样就可以解决这个问题了。

