信号量的实现与应用

### 1．课程设计的内容及要求

本次实验的基本内容是：

* 在 Ubuntu 下编写程序，用信号量解决生产者——消费者问题；
* 在 0.11 中实现信号量，用生产者—消费者程序检验之。

实验目的：

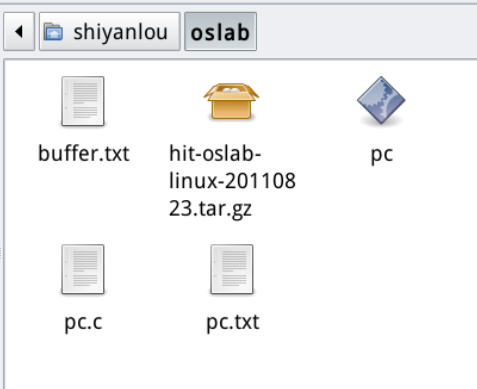
* 加深对进程同步与互斥概念的认识；
* 掌握信号量的使用，并应用它解决生产者——消费者问题；
* 掌握信号量的实现原理。

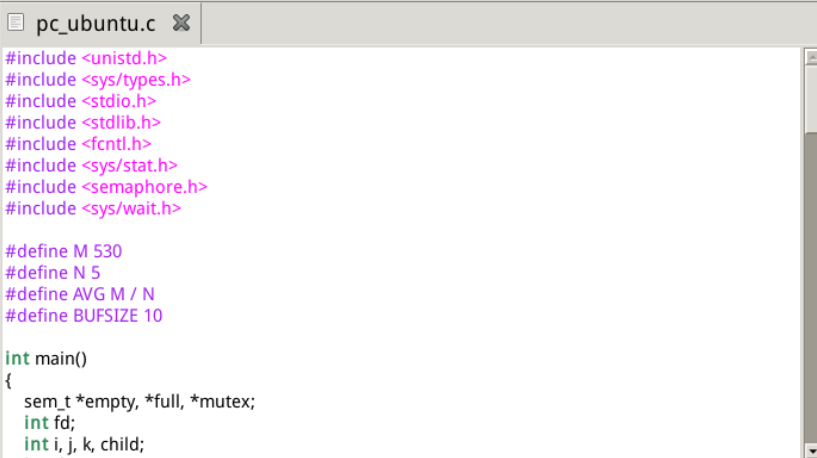
### 选择和使用现代工具

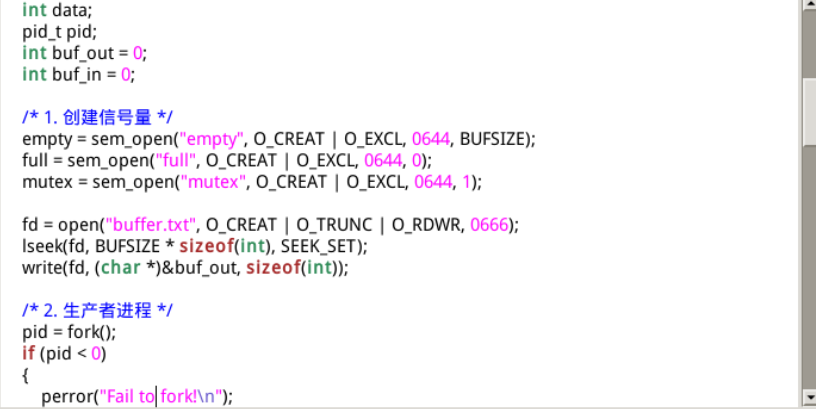
蓝桥云虚拟机

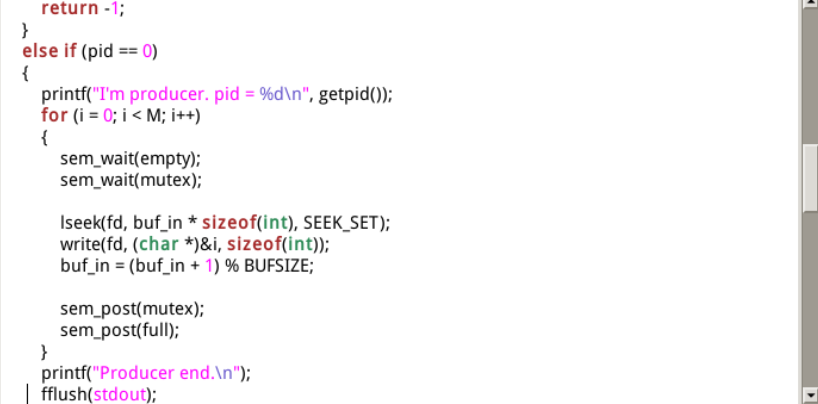
### 4．课题实现

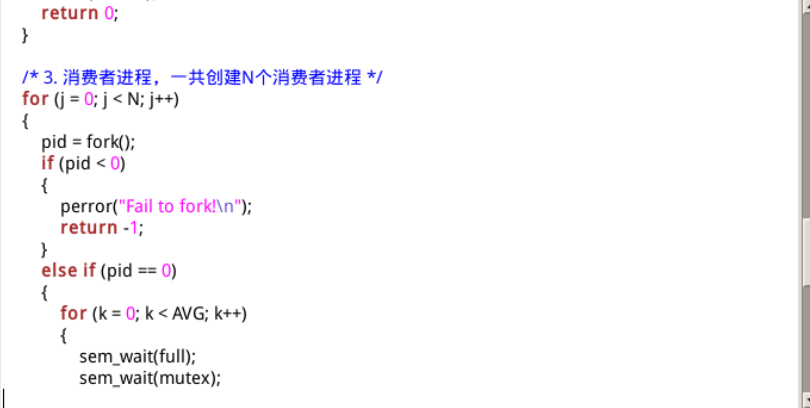
1. 在ubuntu环境下编写pc.c文件，进行测试。

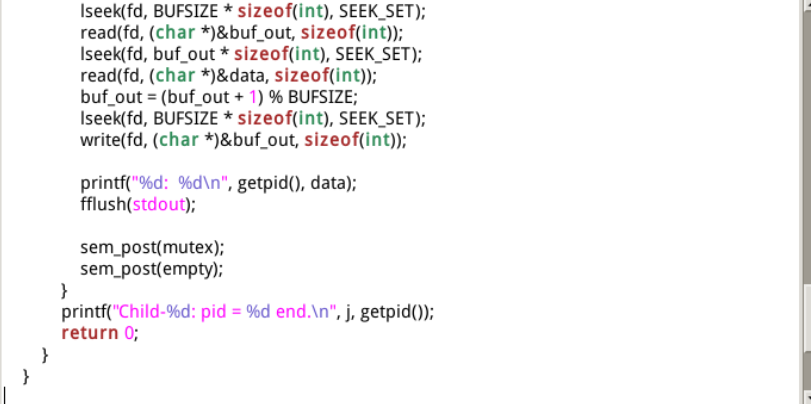














1. 在ubuntu上编译pc.c并且运行。



在编译文件时，利用-lpthread链接ubuntu的线程库。



得出txt文件部分结果如下：



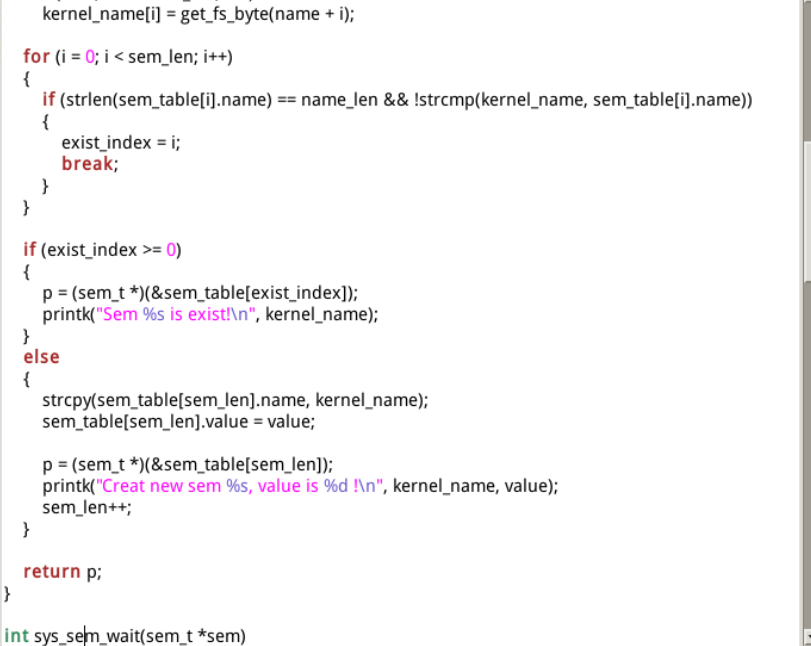
正常运行

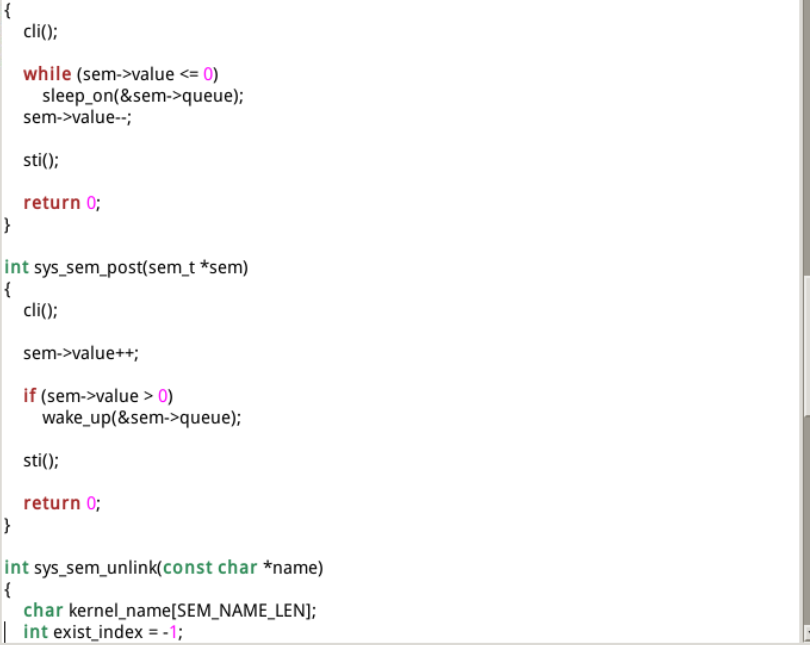
1. 新建头文件include/linux/sem.h，定义sem.h。



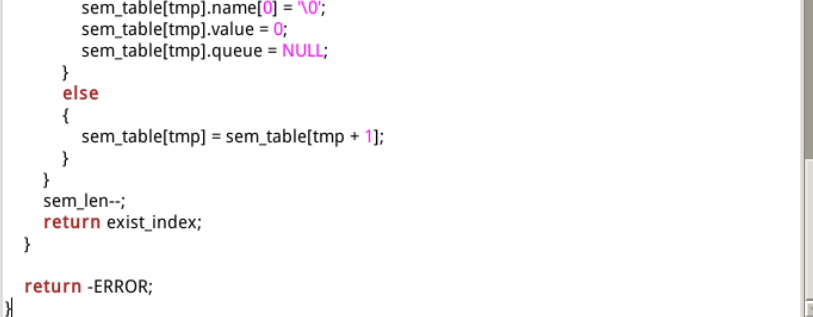
1. 新建文件kernel/sem.c，实现信号量的相关函数：sys\_sem\_open、sys\_sem\_wait、sys\_sem\_post、sys\_sem\_unlink。









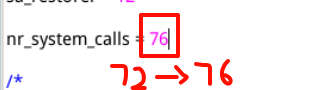


1. 新增4个系统调用。

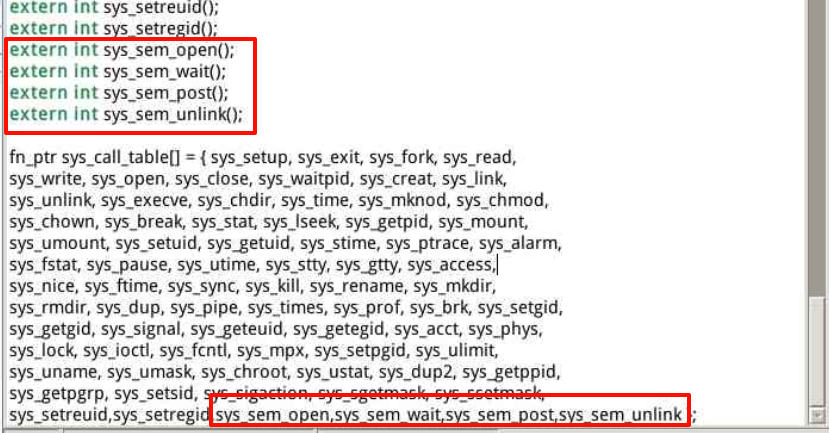
修改/include/unistd.h，添加新增的系统调用的编号：



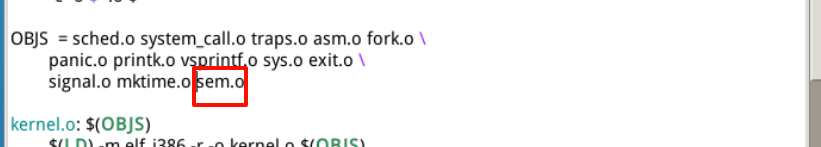
修改/kernel/system\_call.s，需要修改总的系统调用数：

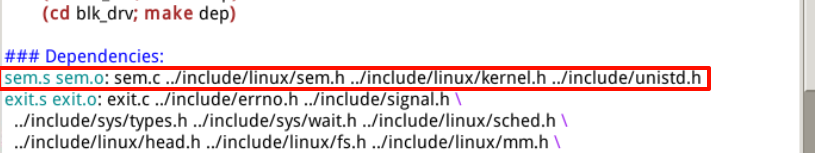


修改/include/linux/sys.h，声明全局新增函数：

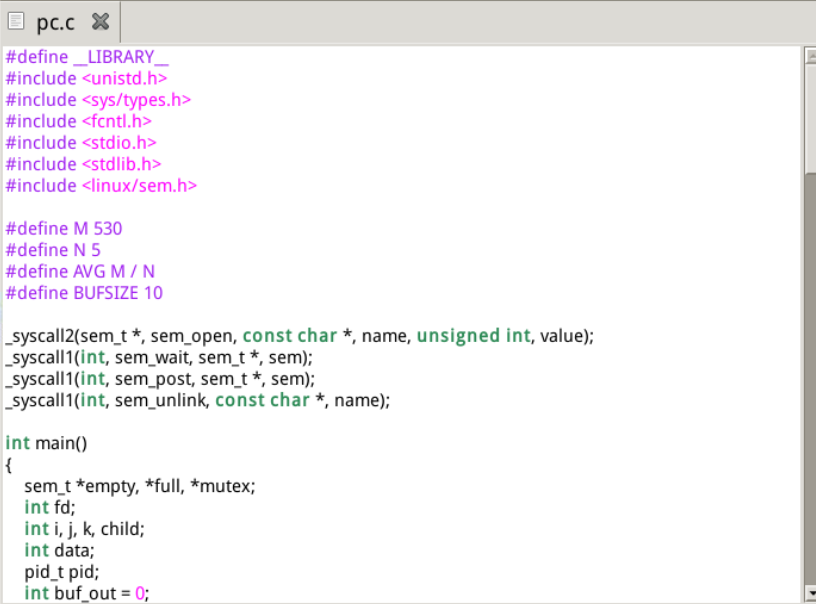


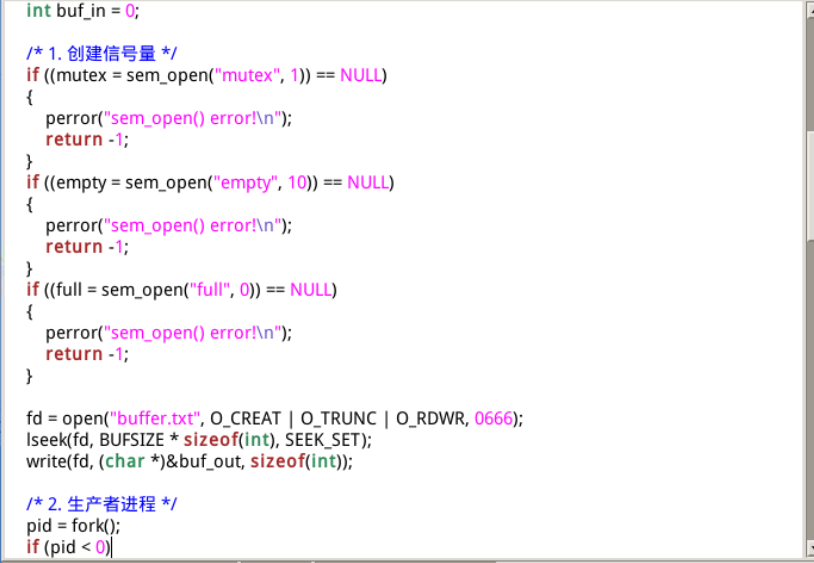
修改 linux-0.11/kernel/Makefile，添加sem.c编译规则：





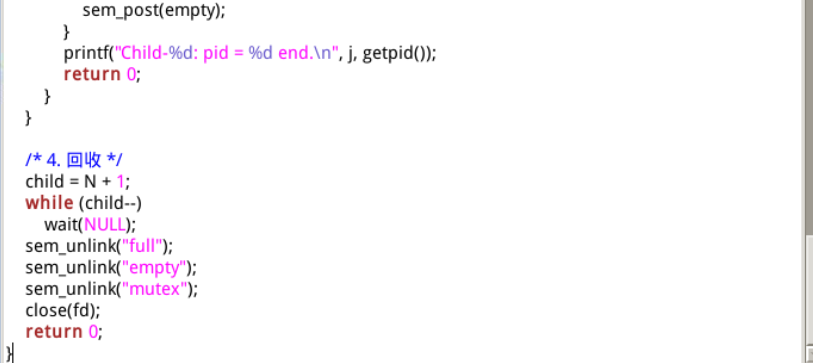
1. 在ubuntu上运行的信号量pc.c程序在Linux0.11不能直接运行，因为Linux0.11并没有实现sem\_xxx相关的函数，所以我们要自己编写相关的函数，并提供调用接口。



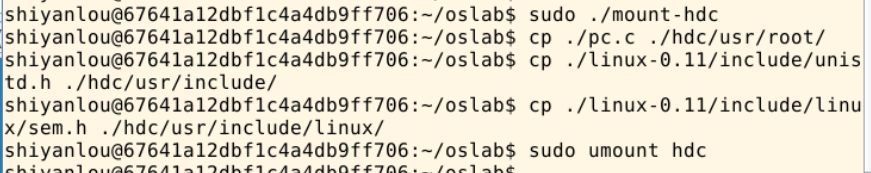




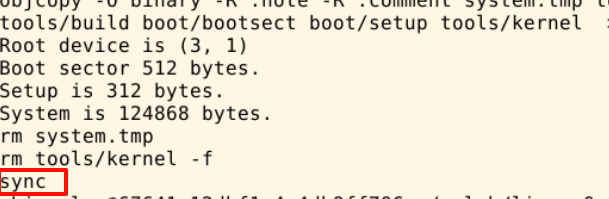




1. 挂在hdc，将所需文件拷贝到linux-0.11系统中。

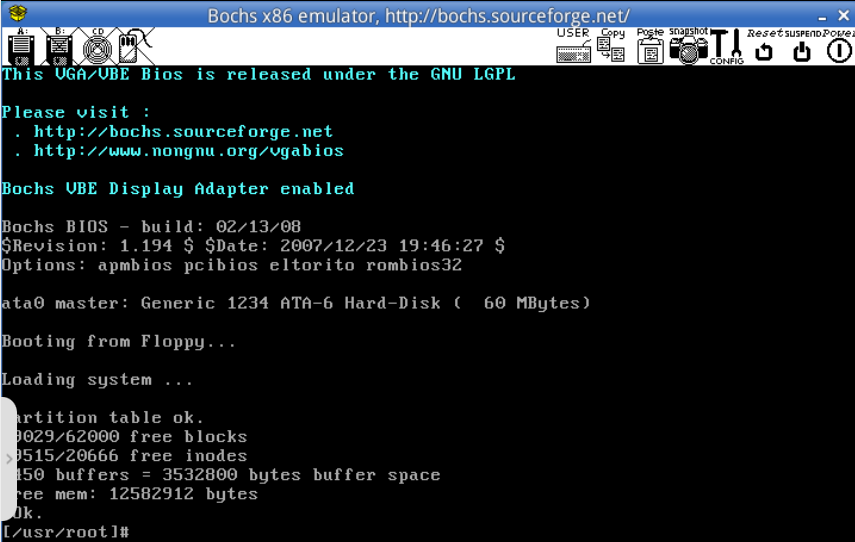


1. 编译内核，运行bochs

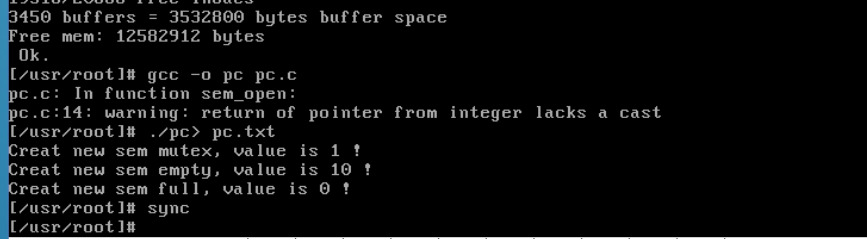


出现sync，成功编译内核。

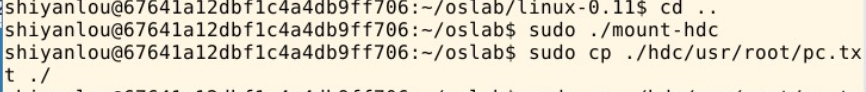
运行bochs



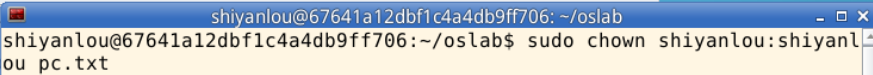
1. 编译pc.c文件，运行程序，并且将结果保存在pc.txt中。



1. 挂在hdc，将pc.txt文件移动到ubuntu中



1. 修改文件的所有者和新组群。



1. 查看部分pc.txt文件。





### 5．课题实现结果

正确全面地表达运行测试结果，要有全面清晰的输入值和运行结果

1. 在ubuntu下编写程序，运行后的部分结果如下：



1. 在linux-0.11环境下运行的部分结果如下：





前面的进程会切换，但是切换的不太自然，在进程16输出0~9，才切换到18，然后进程18输出10~19，切换到14，如此反复切换。

### 6．结果分析与心得体会

6.1 结果分析

实验结果表明，当信号量的初值和操作逻辑设计合理时，能够确保多进程/线程的正常执行，避免资源冲突。具体来说，当多个进程共享临界资源时，信号量机制能够确保只有一个进程访问该资源，从而避免了数据不一致和异常情况的发生。在实验中，我们通过设计多种场景来测试信号量的应用，如生产者消费者问题、读者写者问题等，所有测试用例均能顺利执行，验证了信号量在解决同步和互斥问题中的有效性。

然而，也出现了一些问题，例如在某些特殊的高并发情况下，虽然信号量能够有效控制进程的执行顺序，但在极端条件下的性能瓶颈还是不可忽视的。特别是当多个进程频繁请求和释放信号量时，可能会对系统性能产生一定影响，这需要在实际应用中进行更细致的优化和调整。

6.2 心得体会

6.2.1 优点：

对操作系统中并发控制机制的理解更加深入，特别是对信号量的实际应用有了更直观的感受。信号量作为一种非常基础且重要的同步工具，在多进程/线程环境中的作用至关重要，能够有效保证系统资源的有序访问。在实验过程中，我对信号量的工作原理、P操作和V操作的具体应用有了更深刻的理解，也提高了自己在操作系统实验中解决问题的能力。

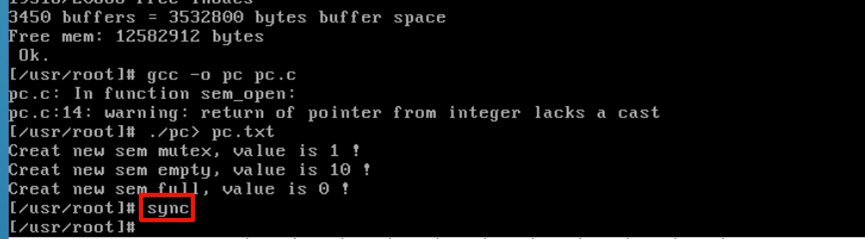
6.2.2 不足：

尽管本次实验取得了预期的结果，但在实现过程中，我也发现自己在一些细节处理上的不足。例如，在设计某些复杂的同步场景时，由于缺乏足够的经验，容易忽略一些边界情况，导致在极端情况下程序的执行效率受到影响。并且对信号量的调度和优化策略也没有进行深入探讨，未能全面考虑信号量在高并发环境下的性能表现。未来在进一步学习的过程中，我需要加强对操作系统底层机制的理解，提升自己在并发控制和资源调度方面的综合能力。

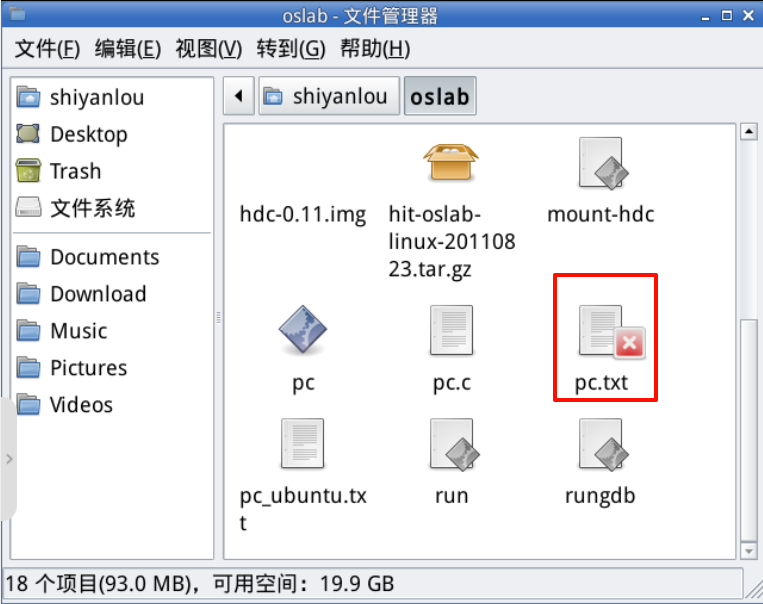
### 主要问题及解决方法

1. 运行bochs后，将信号量运行结果保存在pc.txt文件中，失败了。

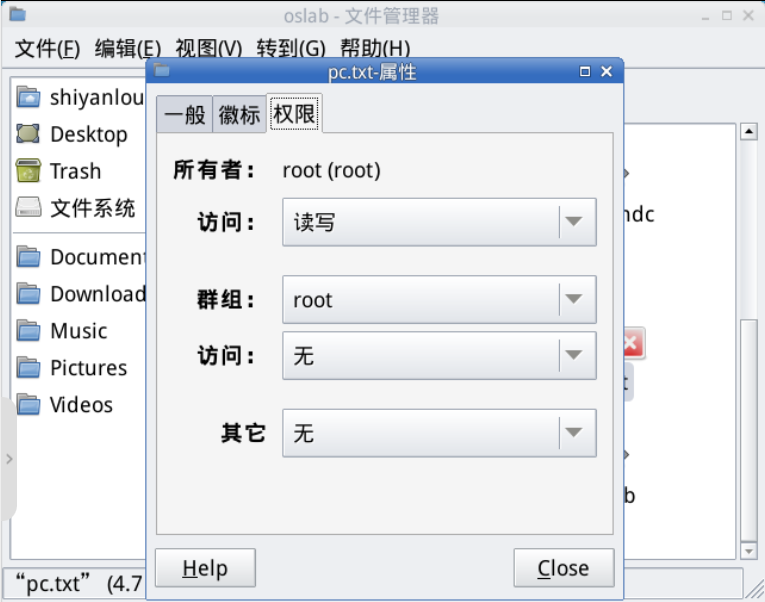
后面发现是在全部输入接受后应该再输入sync，保存数据。



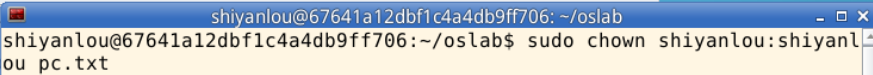
1. 将pc.txt文件转移到ubuntu文件下后，却不能打开，权限不够。



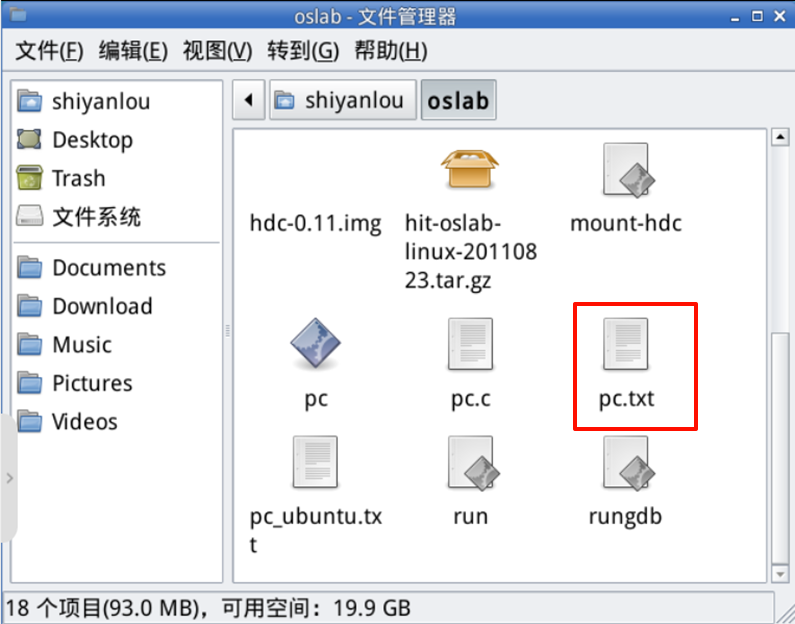
发现权限在linux-0.11环境下的root中。



利用chown指令更改所有者和新群组。



成功可以看到文件，并且所有者和新群组更改为shiyanlou。





### 参考文献

<https://blog.csdn.net/qq_70244454/article/details/128017713>

<https://zhuanlan.zhihu.com/p/13227166199>

<https://kimi.moonshot.cn/chat/cti27kaav1f9npq6f8jg>