地址映射与共享

### 1．课程设计的内容及要求

任务书中的课题内容及要求

对照课题内容要求简要说明自己实现的功能情况

本次实验的基本内容是：

* 用 Bochs 调试工具跟踪 Linux 0.11 的地址翻译（地址映射）过程，了解 IA-32 和 Linux 0.11 的内存管理机制；
* 在 Ubuntu 上编写多进程的生产者—消费者程序，用共享内存做缓冲区；
* 在信号量实验的基础上，为 Linux 0.11 增加共享内存功能，并将生产者—消费者程序移植到 Linux 0.11。

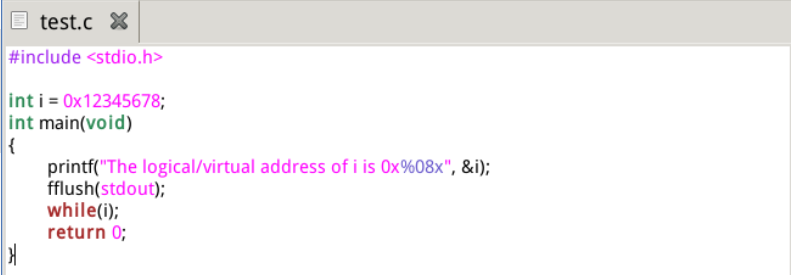
### 选择和使用现代工具

蓝桥云虚拟机

### 4．课题实现

一 跟踪地址翻译过程

1. 在ubuntu下编写test.c，然后拷贝到linux-0.11环境中。





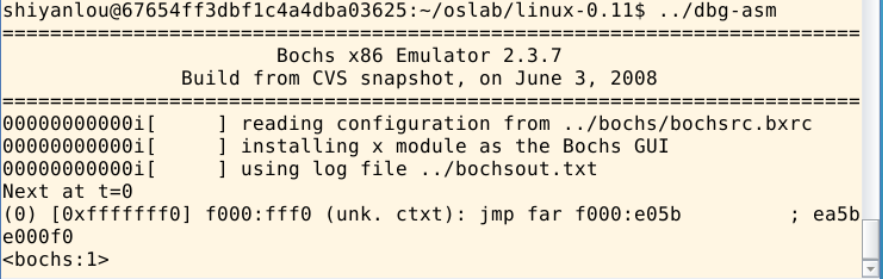
1. 编译内核，然后运行../dbg-asm启动调试器。



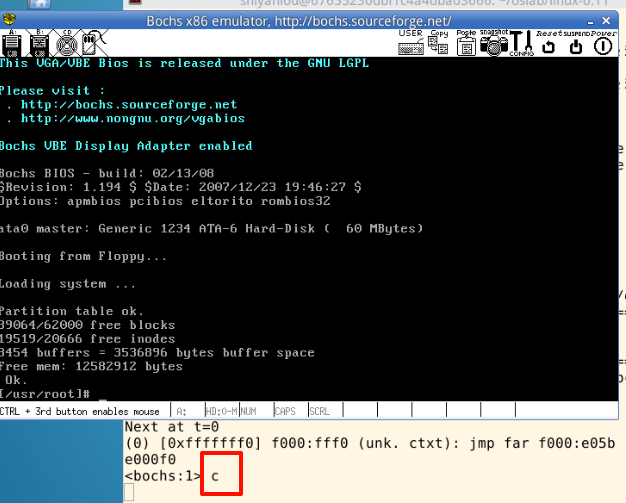
此时bochs窗口处于黑屏



此时终端显示如下：



其中Next at t=0表示下面的指令是Bochs启动后要执行的第一条软件指令，输入命令c（既是continue），直接运行程序，bochs就会启动linux-0.11。



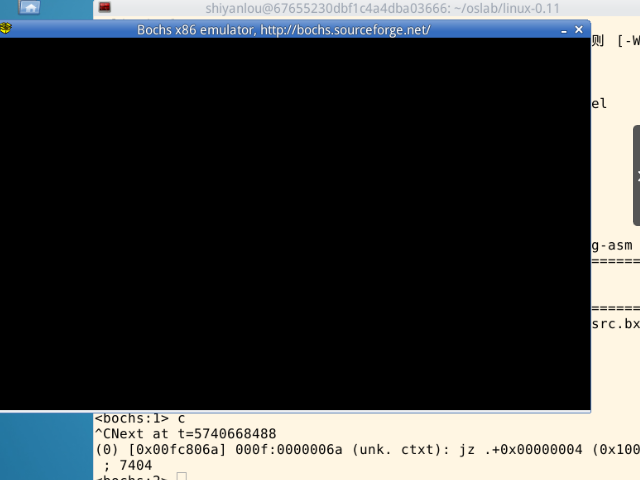
成功启动linux-0.11。

现在编译test.c并且运行。

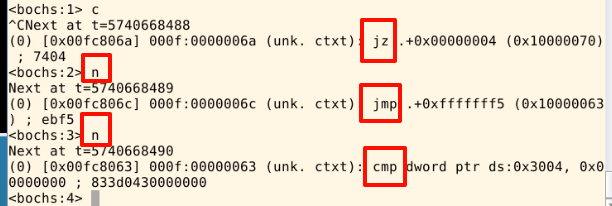


test程序是一个死循环，只会不停占用CPU，并不会退出。

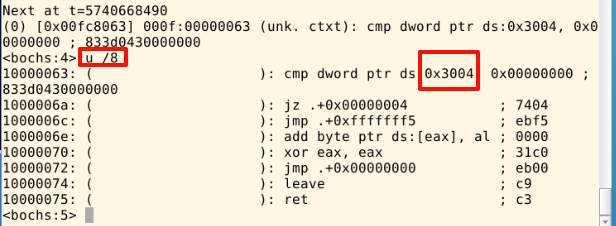
1. 在test运行的时候，在终端中按下ctrl+c，bochs会暂停运行，加入调试状态



可以看到bochs加入调试状态，在终端中后面出现jz…，而不是cmp…。因此我们用n命令单步运行直到停在cmp…

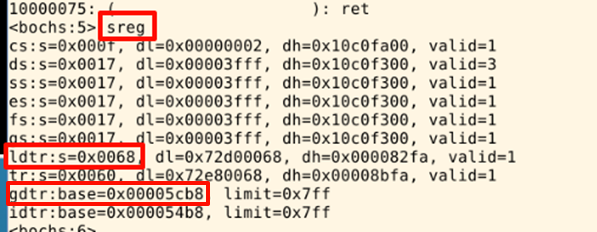


接下来使用命令u /8，显示从当前位置开始的8条指令的反汇编代码。



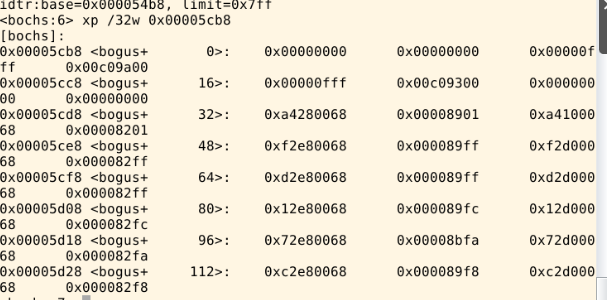
该8条指令就是test.c中从while开始一直到return的汇编代码。其中变量i保存在ds:0x3004这个地址，并不停地和0进行比较，直到它为0，才会跳出循环。

1. 通过sreg命令查看各个寄存器的值



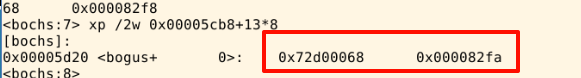
可以看到ldtr的值是0x0068，转换为二进制就是0000000001101000，表示LDT表存放在GDT表的1101B=13D号位置（每位数据的意义参考后文叙述的段选择子）。而GDT表的位置由gdtr给出，即物理地址的 0x00005cb8。

1. 利用xp /32w 0x00005cb8查看从物理地址0x00005cb8开始的32个字（128字节）的内容，即GDT表的前16项。



其中GDT表中的每一项占64位（8个字节，即2个字），我们已经知道LDT表存放在GDT表的13号位置，所以要查找的项的地址就是 0x00005cb8+13\*8。

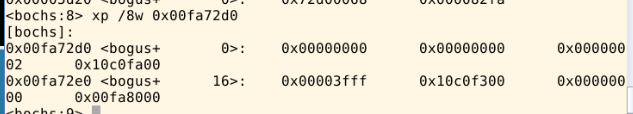
1. 输入xp /2w 0x00005cb8+13\*8



与sreg命令中的在ldtr所在行的dl和dh数值一样，那么说明正确的。

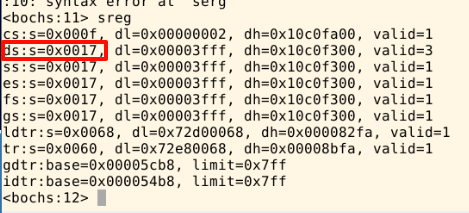
现在将现在将0x72d00068的前四位和0x000082fa后两位组合为0x00fa72d0，就是LDT表的物理地址。

1. 输入xp /8w 0x00fa72d0，得到LDT表的前8个字内容。



LDT表的前4项内容。

1. 利用sreg命令查看ds选择子的内容。

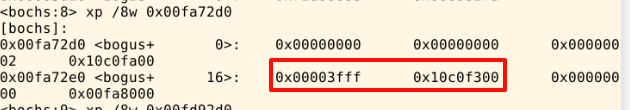


ds 的值是0x0017，而段选择子ds是一个16位寄存器。

1. 输入calc ds:0x3004命令验证线性地址

线性地址=段基址+段内偏移

通过xp /8w 0x00fa72d0查看



0x00003fff的前四位和0x10c0f300的前两位和后两位组合成0x10000000。

0x10000000 + 0x3004 = 0x10003004。

输入calc ds:0x3004验证结果。



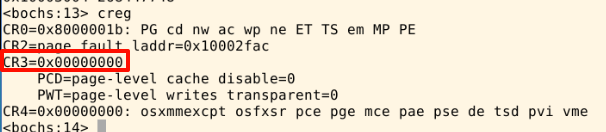
正确。

1. 查找页表

线性地址中的页目录号、页表号和页内偏移，它们分别对应了 32 位线性地址的10位 + 10位 + 12位。

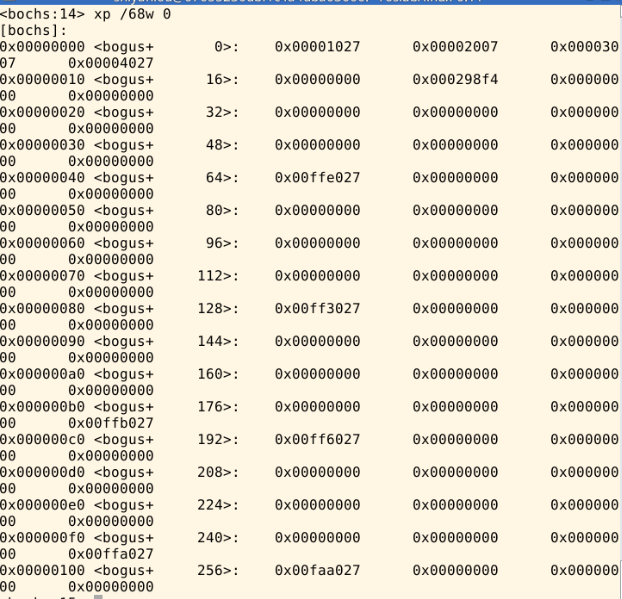
所以线性地址0x10003004 = 0001000000 0000000011 000000000100（二进制）的页目录号就是64（1000000），页表号为3（0000000011），页内偏移为 4（000000000100）。

通过creg命令查看CR3寄存器指引（在IA-32下，页目录表的位置由CR3寄存器指引）。

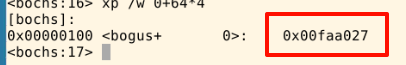


页目录表的基址为0。

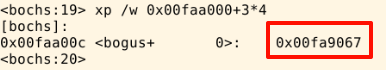
通过xp /68w 0查看页目录表前68个字内容。



通过xp /w 0+64\*4查看页目录号64就是第65个页目录项。



可以看出页表所在的物理页框号为0x00faa，即页表在物理内存为0x00faa000处，所以从该位置开始查找3号页表项（每个页表项4个字节，即1个字），通过命令xp /w 0x00faa000+3\*4查看。

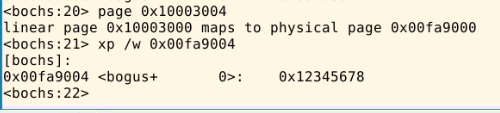


067是属性，fa9就是物理页框号。

1. 检查物理地址。

线性地址0x10003004对应的物理页框号为0x00fa9，接着和页内偏移 0x004接到一起，得到 0x00fa9004，就是变量i的物理地址。

输入命令page 0x10003004或者命令xp /w 0x00fa9004可以得到如下内容：



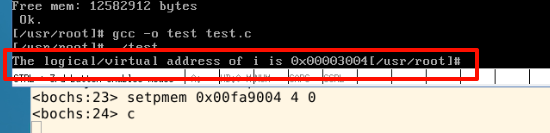
通过page 0x10003004输出内容，可以看到线性地址 0x10003000 对应的物理页地址就是 0x00fa6000。

通过xp /w 0x00fa9004输出内容，可以看到这个数值就是test.c中i的初值。

1. 通过setpmem 0x00fa9004 4 0将从地址0x00fa9004开始的4个字节都设为0。



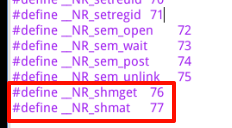
1. 输入c命令使bochs运行。



test成功退出了，说明i的修改成功。

二 实现共享内存（在信号量的实现和应用实验基础上）

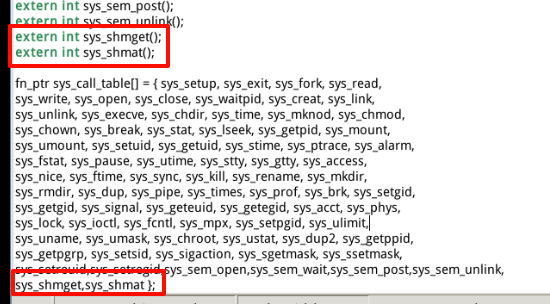
1. 添加系统调用（shmget和shmat）。



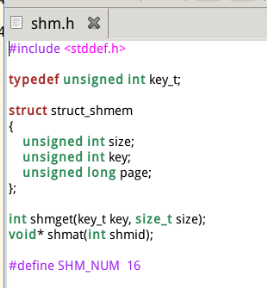
1. 修改系统调用总数。



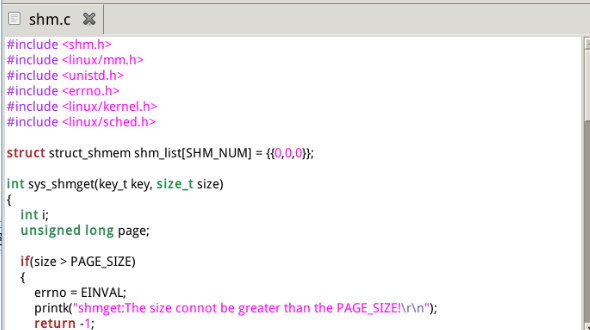
1. 添加系统调用函数名并维护系统调用表。



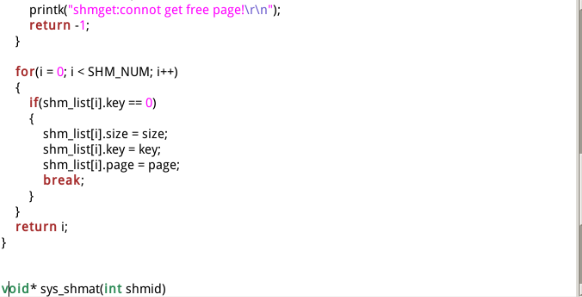
1. 编写shm.h

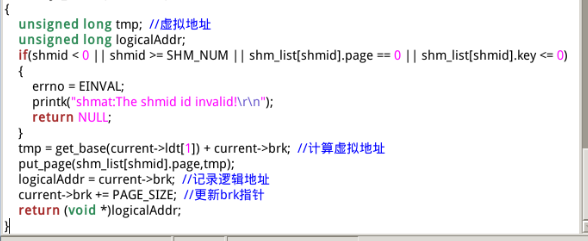


1. 编写shm.c

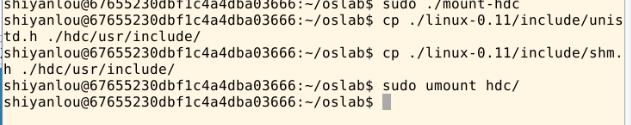




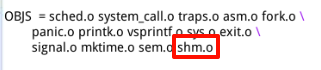


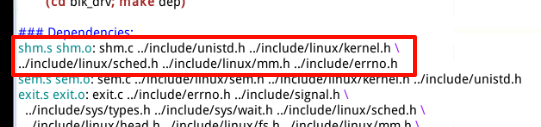


1. 挂载文件



1. 修改文件编译规则。





1. 编译内核

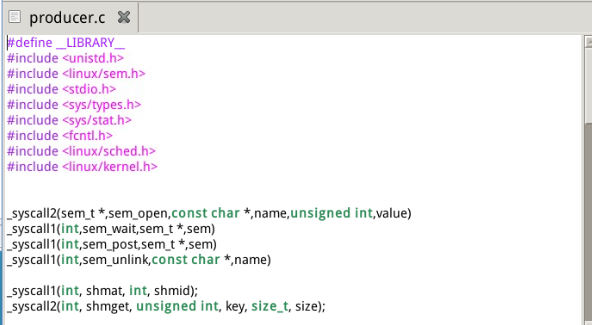


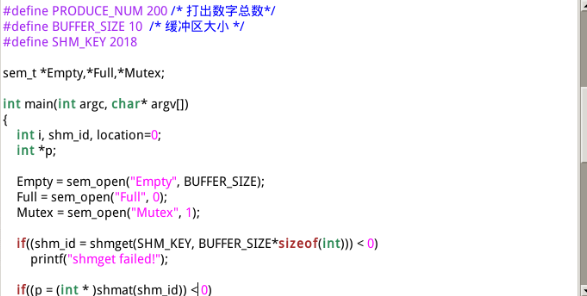
成功编译内核。

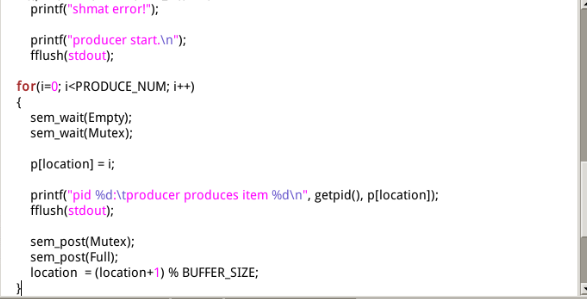
三 基于共享内存的生产者-消费者程序

1. 编写producer.c和consumer.c

producer.c

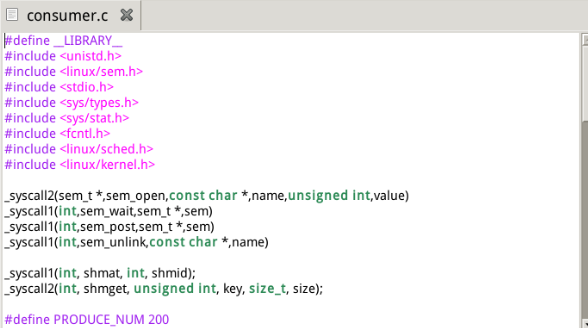


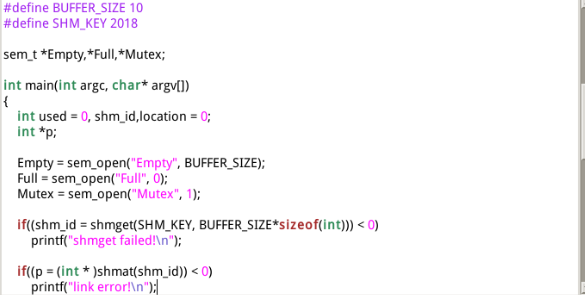


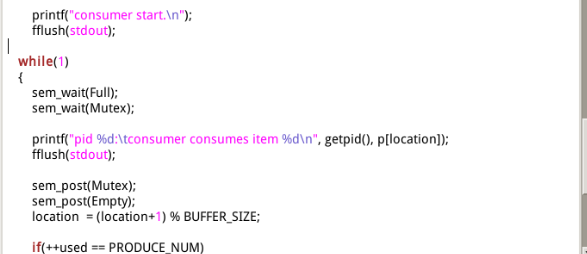




consumer.c

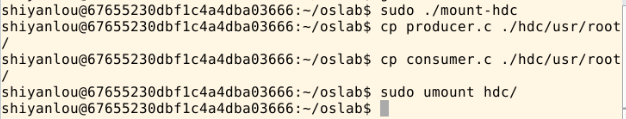






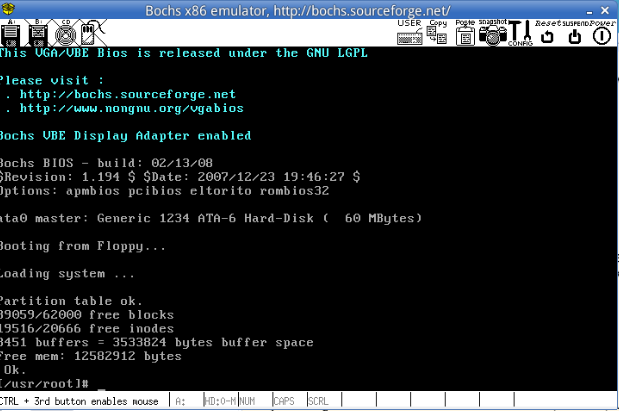


1. 挂载文件

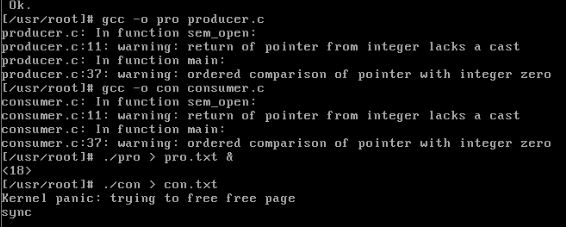


1. 编译内核并且运行

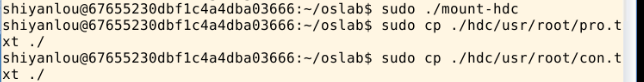




1. 编译文件并且运行，然后输出文件txt类型



1. 将生成的txt文件拷贝到ubuntu下

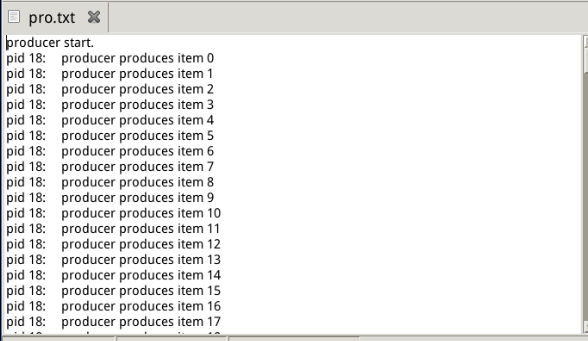


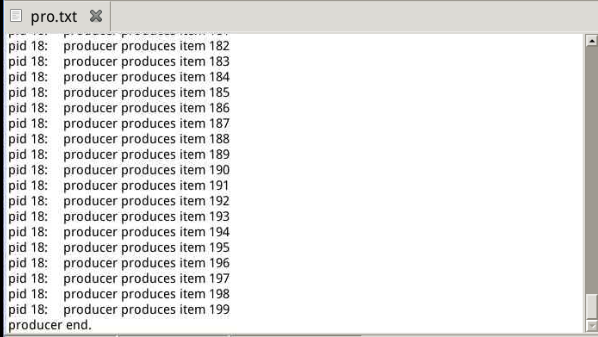
1. 修改文件权限



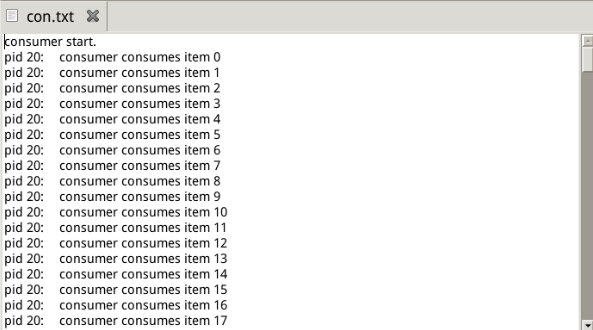
1. 查看pro.txt和con.txt

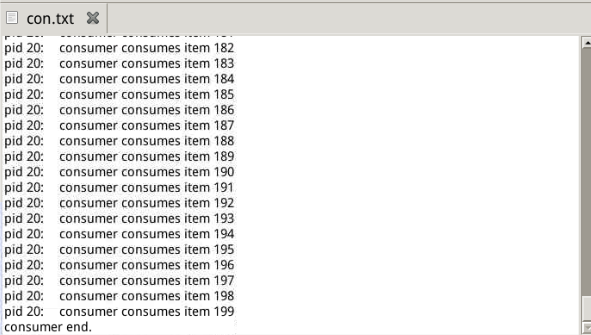
pro.txt部分内容如下：





con.txt





### 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

### 6．结果分析与心得体会

针对自己课题实现结果进行有效分析，并能够自我评价优势和不足等

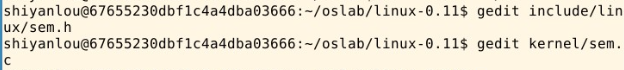
### 主要问题及解决方法

在实验过程中遇到的主要问题及解决方法

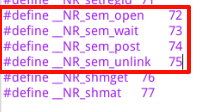
1. 在bochs中编译producer.c文件文件出现如下错误：



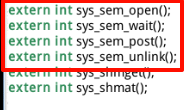
发现重新利用了信号量实验与应用中的文件，将其全部补充上（sem.h、sem.c）。

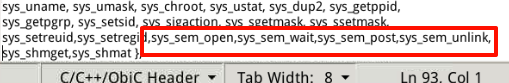


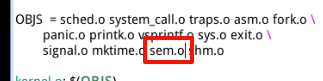
添加系统调用编号，修改系统调用总数，添加系统调用函数名并维护系统调用表。修改编译规则。

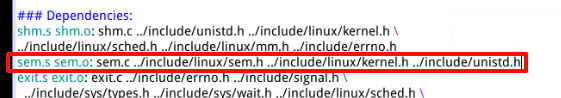


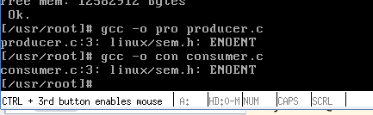






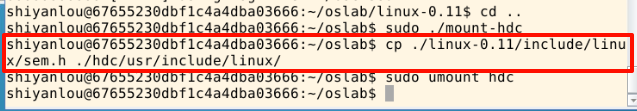




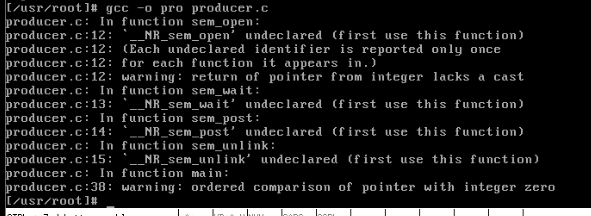


在编译的时候又出错了。

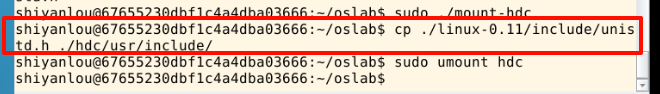
继续排查，查询到是因为没有把sem.h拷贝到linux-0.11系统中。

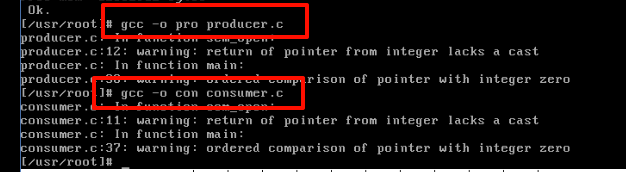


在运行时又出现了新的错误，如下：



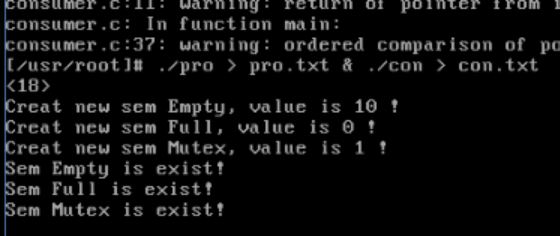
排查发现unistd.h中已经修改，但是还是出现如图说在unistd.h中的错误，发现是没有把跟新的unistd.h拷贝到linux-0.11系统中。





终于，成功编译！

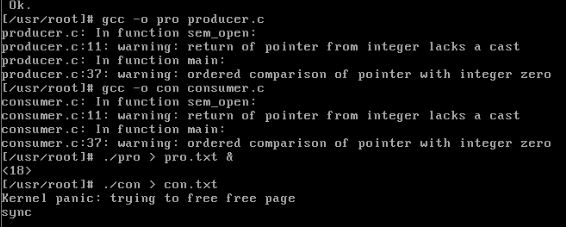
1. 编译成功后，运行./pro和./con时，又出现了如下错误：



后面发现是sem.c和sem.h文件内容有错误，改写后保存，并且把sem.h拷贝到linux-0.11系统中。



重新编译内核并运行。



终于成功了！

### 参考文献

要列出在课题设计实现过程中参考过的文献，表现诚信的科学素养

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

<https://blog.csdn.net/Amentos/article/details/131392469?fromshare=blogdetail&sharetype=blogdetail&sharerId=131392469&sharerefer=PC&sharesource=William_lyx&sharefrom=from_link>

<https://blog.csdn.net/Amentos/article/details/131002529>

<https://blog.csdn.net/qq_70244454/article/details/128303818?fromshare=blogdetail&sharetype=blogdetail&sharerId=128303818&sharerefer=PC&sharesource=William_lyx&sharefrom=from_link>