

**Lab1 操作系统初步**

**实验报告**

**孙睿**

**16281079**

**安全1601班**

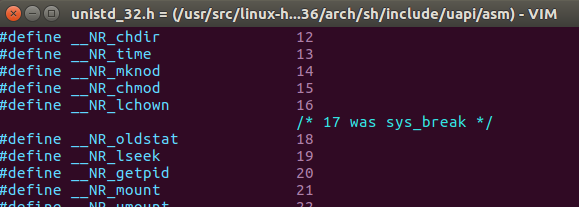
# 实验目的

1. 了解系统调用不同的封装形式。
2. 并发实验
3. 内存分配实验
4. 了解并发的共享问题

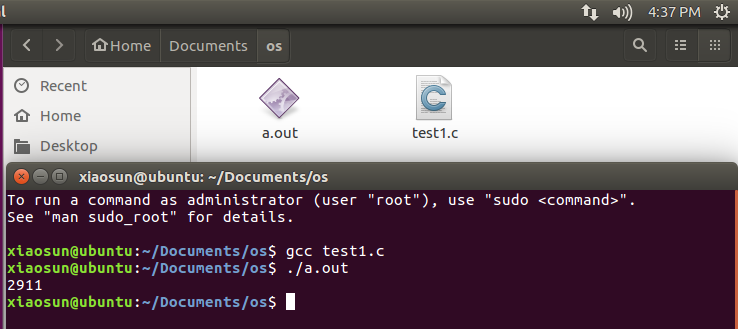
# 实验内容及步骤

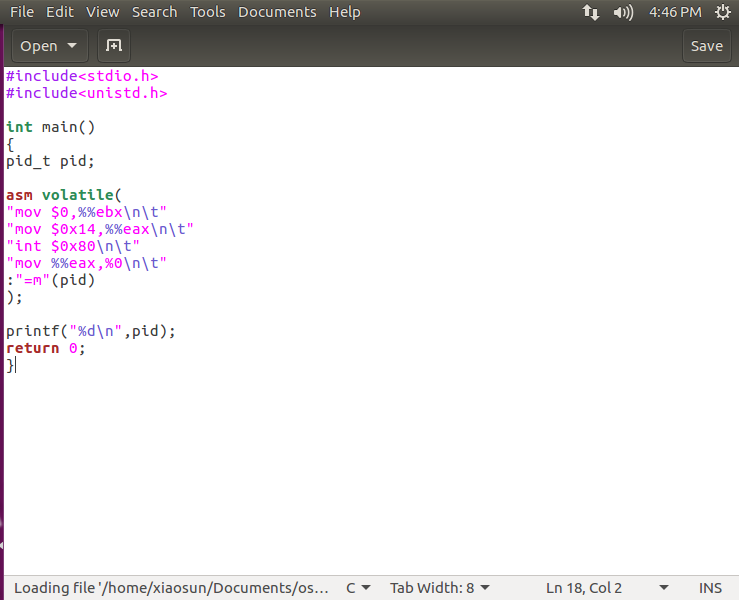
1. 参考下列网址中的程序。阅读分别运行用API接口函数getpid()直接调用和汇编中断调用两种方式调用Linux操作系统的同一个系统调用getpid的程序(请问getpid的系统调用号是多少？linux系统调用的中断向量号是多少？)。2、上机完成习题1.13。3、阅读pintos操作系统源代码，画出系统调用实现的流程图。

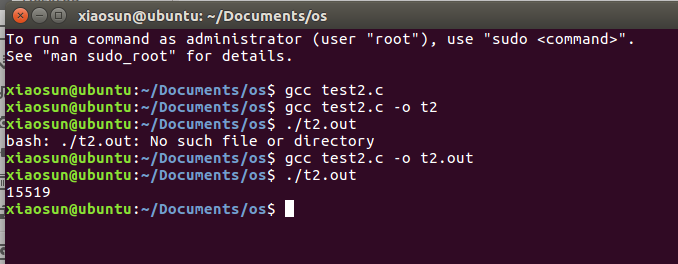
（1） getpid的系统调用号是20；linux系统调用的中断向量号是0x80。



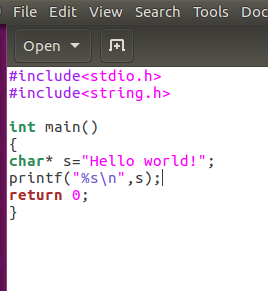


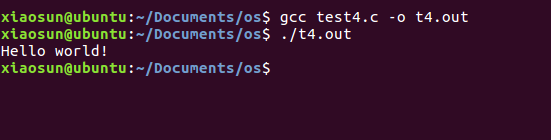


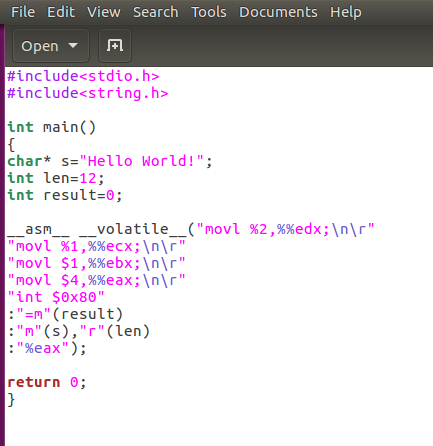


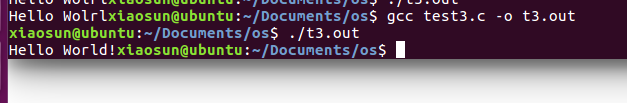


（2）习题1.13

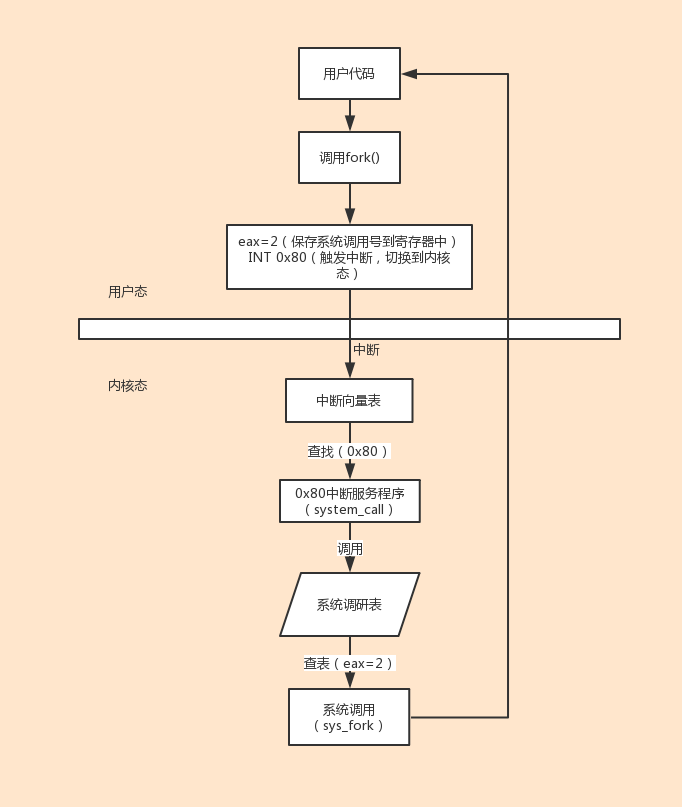








（3）系统调用实现的流程图



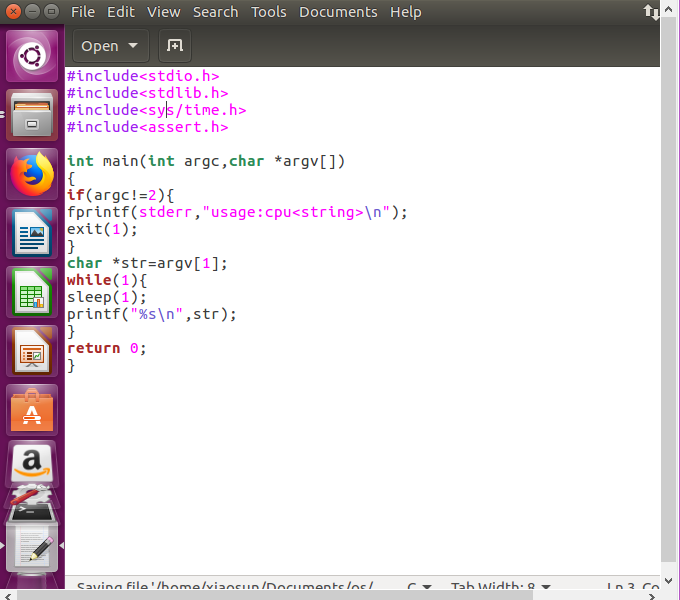
2、（1）编译运行该程序（cpu.c），观察输出结果，说明程序功能。

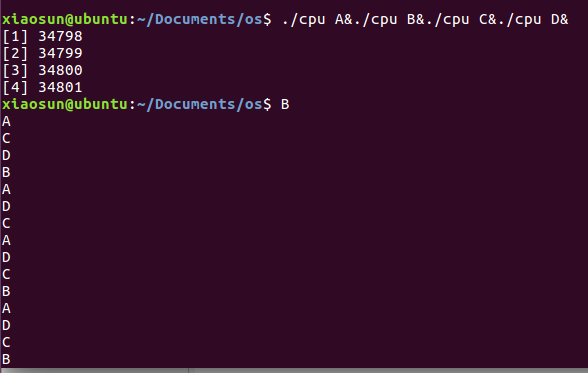
(编译命令： gcc -o cpu cpu.c –Wall)（执行命令：./cpu）

（2）再次按下面的运行并观察结果：执行命令：./cpu A & ; ./cpu B & ; ./cpu C & ; ./cpu D &程序cpu运行了几次？他们运行的顺序有何特点和规律？请结合操作系统的特征进行解释。

**（1）**这个程序的功能是当收到输入的字符，如：A时，不断输出A

**（2）**每输出一次字符，CPU运行了一次。虽然执行命令./cpu A & ; ./cpu B & ; ./cpu C & ; ./cpu D 是按ABCD的顺序，但是程序执行结果不是按这种顺序输出。说明CPU并不是按顺序执行程序，系统可能会某些异常，而先执行其他程序。



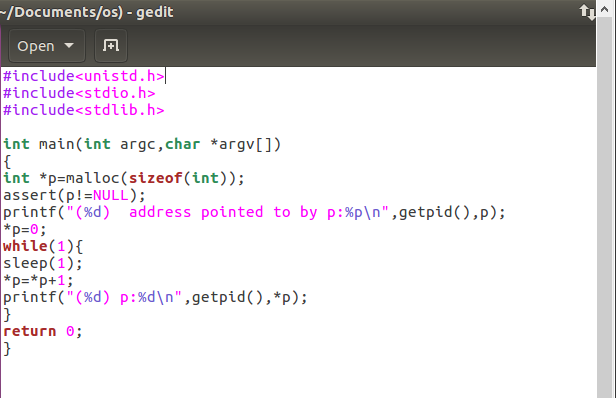


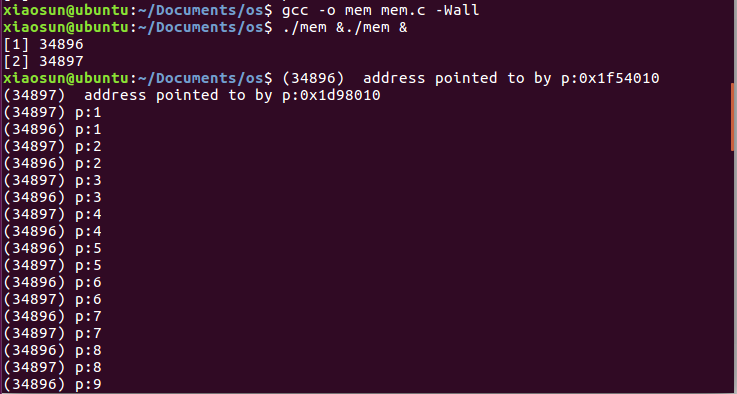
3、（1）阅读并编译运行该程序(mem.c)，观察输出结果，说明程序功能。(命令： gcc -o mem mem.c –Wall)

（2）再次按下面的命令运行并观察结果。两个分别运行的程序分配的内存地址是否相同？是否共享同一块物理内存区域？为什么？命令：./mem &; ./mem &

**（1）**这个程序是一个循环程序，将p循环+1，并输出程序分配的内存地址

**（2）**根据程序输出结果可以看出，两个程序分配的内存地址不相同。一个是0x1f54010,一个是0x1d98010。两个程序不共享同一块物理内存区域。因为如果是同一区域，先执行的程序的结果会对下一个程序产生影响。如果p=1后下一个程序执行时p会加1等于2，继续执行下一个程序，p=2+1=3。





3、（1）阅读并编译运行该程序，观察输出结果，说明程序功能。（编译命令：gcc -o thread thread.c -Wall –pthread）（执行命令1：./thread 1000）

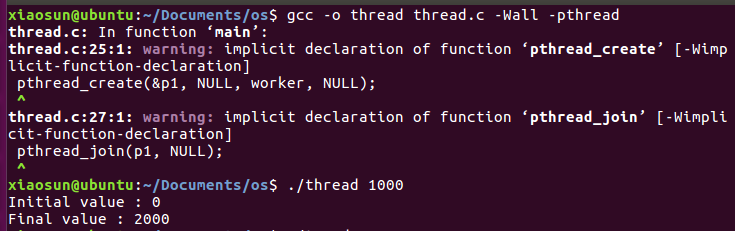
（2）尝试其他输入参数并执行，并总结执行结果的有何规律？你能尝试解释它吗？（例如执行命令2：./thread 100000）（或者其他参数。）

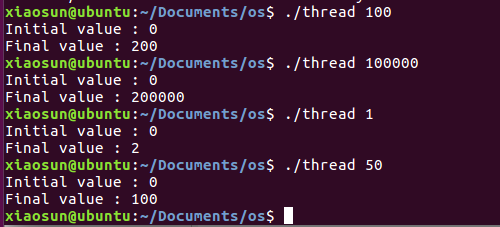
提示：哪些变量是各个线程共享的，线程并发执行时访问共享变量会不会导致意想不到的问题

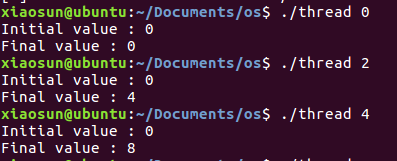
全局变量是各个线程共享的

（1）输出两个线程进行前的结果，再将两个线程结果输出

（2）尝试了很多组参数之后，发现最终结果都是两个线程结果。由于全局变量是共享的，所以两个线程访问同一个counter数据，在执行worker函数时，两个线程会将上一个线程的执行结果代入，所以直观看来程序输出结果为输出参数的2倍。







# 收获和总结

通过这几个实验，我对操作系统给应用程序提供服务的方式–系统调用有所了解，理解系统调用与普通函数调用不同之处。理解汇编代码中，系统调用号通过eax寄存器传递给内核，然后启动INT0x80后，规定返回值送EAX寄存器。软中断就是：进程 ->内核中的设备驱动程序，是一种需要内核为正在运行的进程去做一些事情（通常为I/O）的请求。

CALL 调用子程序 一般会把返回地址压入堆栈, C语言中的函数调用一般都是这种形式 ，所以一般调用程序里面,都会有一个对应的RET指令；INT 软件中断指令会把返回地址压入堆栈,但是程序PC会跳转到对应中断入口处 jmp跳转指令，就是把程序PC转到对应程序上,不会保存任何东西，就像C语言中的.goto语句.所以他不能用RET返回程序(没有压栈)。

并发程序并不是按照开始顺序执行，会因为某些原因导致执行程序的顺序发生改变。CPU会先执行可执行的程序。其中，spin(1)可以用sleep(1)代替。。spin(1)含义是循环执行空操作1秒钟。

两个程序如果共享同一个物理存储区域的话，再调用时，调用的是上一个程序执行后的结果。