

**<<操作系统>>第一次实验**

**操作系统初步**

**学院：计算机与信息技术**

**专业：信息安全（保密技术）**

**班级： 安全1601**

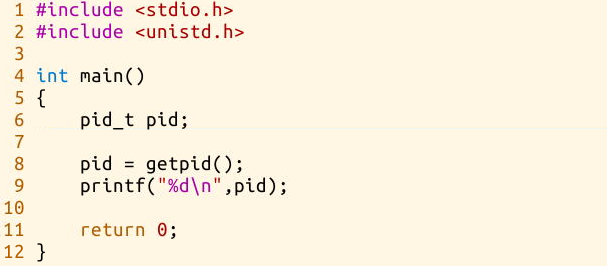
**姓名： 李超**

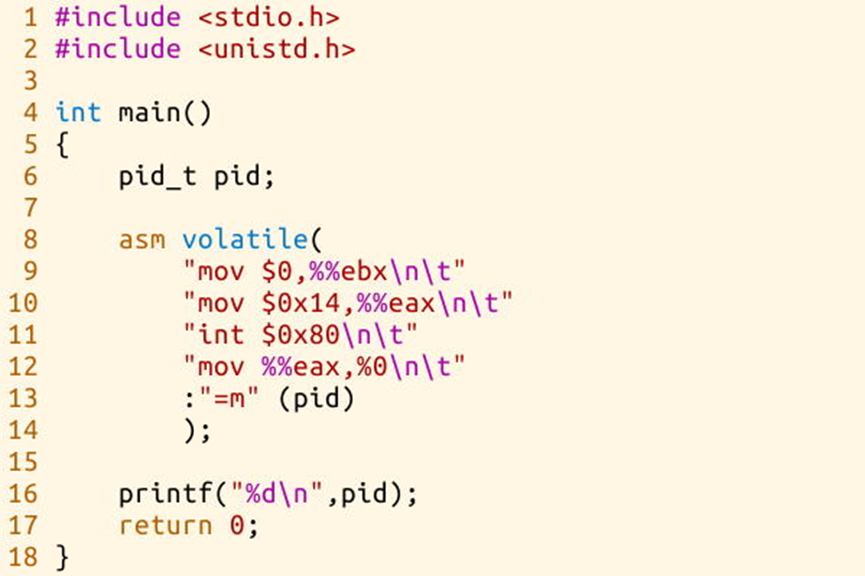
**学号： 16281194**

**2019年3月5日**

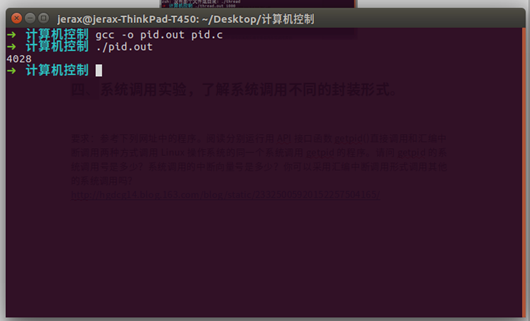
# 一、（系统调用实验）了解系统调用不同的封装形式。

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？)。



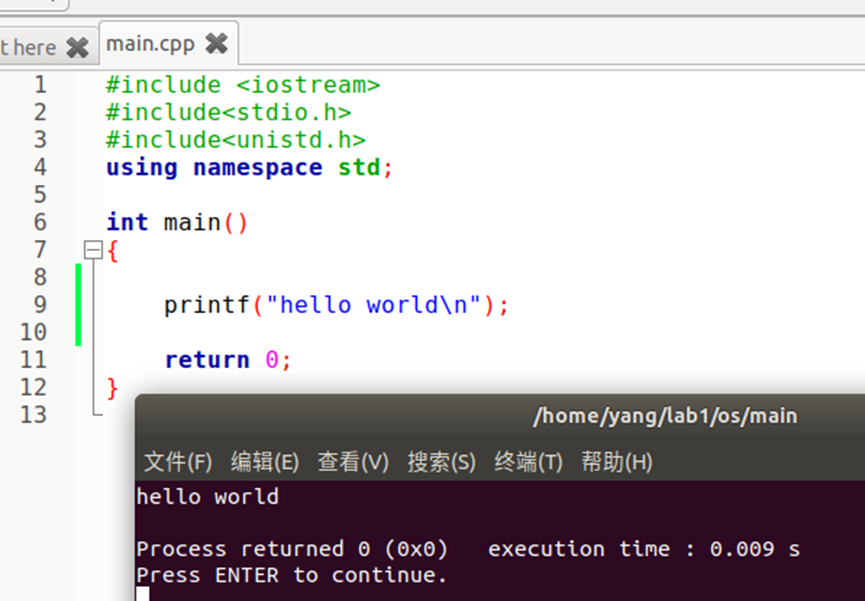


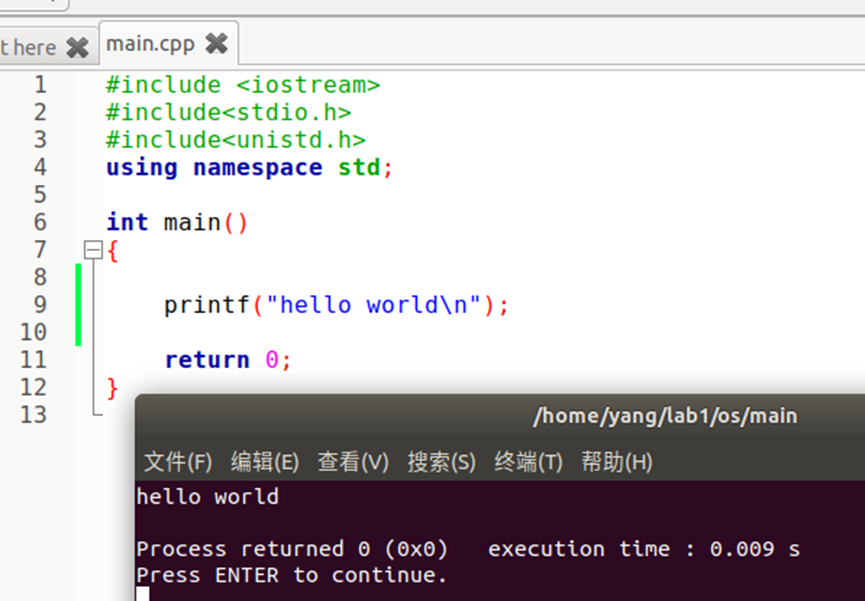
系统调用getpid的中断号通过查询syscall\_32.tbl表可知为20。系统调用其实就是内核空间里的函数的执行，但是出于提高系统安全性，提高用户程序的可移植性等目的，用户程序是不能直接调用或修改内核空间里的函数的，这个时候就用到了系统调用，它其实就是连接用户空间和内核空间的纽带，内核内函数的具体实现和系统调用函数之间的对应关系通过syscall\_32.tbl表来记录。



2、上机完成习题1.13。

Linux调用C函数形式



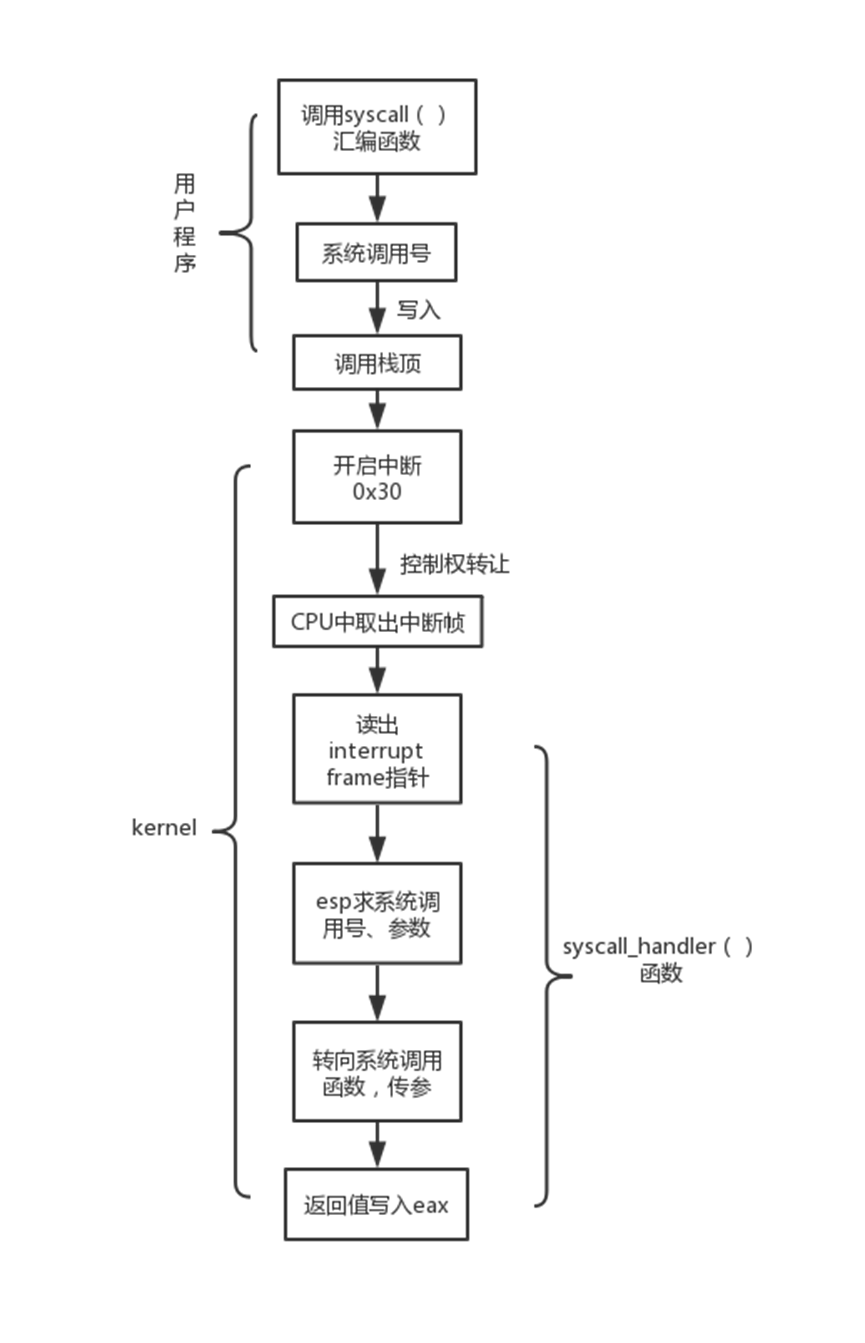


汇编形式



这段代码在64位系统中无法直接编译，编译32位程序需要加上”-m32”。但是直接输入会提示错误，需要安装适配库” sudo apt-get install gcc-multilib g++-multilib module-assistant”。

3、阅读pintos操作系统源代码，画出系统调用实现的流程图。



# 二、（并发实验）根据以下代码完成下面的实验。

要求：

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 #include <stdio.h>**

**2 #include <stdlib.h>**

**3 #include <sys/time.h>**

**4 #include <assert.h>**

**5 #include "common.h"**

**6**

**7 int**

**8 main(int argc, char \*argv[])**

**9 {**

**10 if (argc != 2) {**

**11 fprintf(stderr, "usage: cpu <string>\n");**

**12 exit(1);**

**13 }**

**14 char \*str = argv[1];**

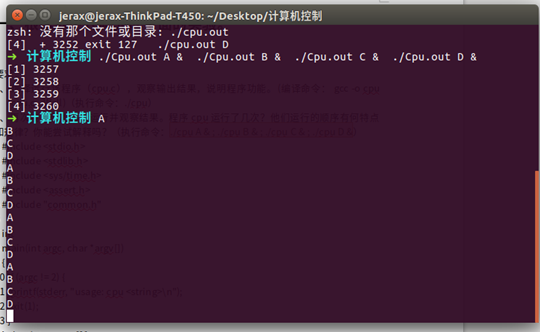
**15 while (1) {**

**16 spin(1);**

**17 printf("%s\n", str);**

**18 }**

1. **eturn 0;**



经过上面的实验，在cpu在一个周期内运行了4次但是由于测不准原理，无法准确的确认时间开销，但是可以大致计算吞吐量。

输出顺序与系统调度有关系，如果是单道或者轮转，那输出肯定是顺次的，但是由于现在的系统调度比较复杂，如果是随机调度，输出顺序就是不一定的。

# 三、（内存分配实验）根据以下代码完成实验。

要求：

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 #include <unistd.h>**

**2 #include <stdio.h>**

**3 #include <stdlib.h>**

**4 #include "common.h"**

**5**

**6 int**

**7 main(int argc, char \*argv[])**

**8 {**

**9 int \*p = malloc(sizeof(int)); // a1**

**10 assert(p != NULL);**

**11 printf("(%d) address pointed to by p: %p\n",**

**12 getpid(), p); // a2**

**13 \*p = 0; // a3**

**14 while (1) {**

**15 Spin(1);**

**16 \*p = \*p + 1;**

**17 printf("(%d) p: %d\n", getpid(), \*p); // a4**

**18 }**

**19 return 0;**



经过上面的实验两个程序分配的内存地址不同，但是在关闭了地址随机化，分配地址相同。但是具体物理地址不能完全确定，其中有对应关系。是随机的，可能不是物理连续的。

# 四、（共享的问题）根据以下代码完成实验。

要求：

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。）
3. 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。

**1 #include <stdio.h>**

**2 #include <stdlib.h>**

**3 #include "common.h"**

**4**

**5 volatile int counter = 0;**

**6 int loops;**

**7**

**8 void \*worker(void \*arg) {**

**9 int i;**

**10 for (i = 0; i < loops; i++) {**

**11 counter++;**

**12 }**

**13 return NULL;**

**14 }**

**15**

**16 int**

**17 main(int argc, char \*argv[])**

**18 {**

**19 if (argc != 2) {**

**20 fprintf(stderr, "usage: threads <value>\n");**

**21 exit(1);**

**22 }**

**23 loops = atoi(argv[1]);**

**24 pthread\_t p1, p2;**

**25 printf("Initial value : %d\n", counter);**

**26**

**27 Pthread\_create(&p1, NULL, worker, NULL);**

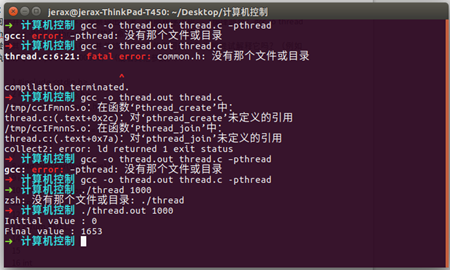
**28 Pthread\_create(&p2, NULL, worker, NULL);**

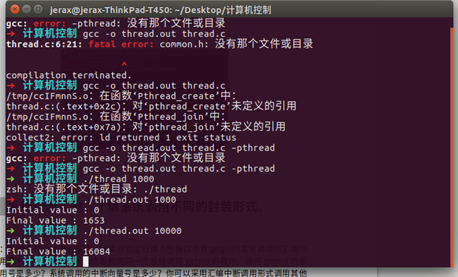
**29 Pthread\_join(p1, NULL);**

**30 Pthread\_join(p2, NULL);**

**31 printf("Final value : %d\n", counter);**

**32 return 0;**





在分别修改了参数之后，得到的数值不同于两倍，理论值应该是输入值得两倍，但是由于冲突，所以数值会小于两倍。