**实验一: 操作系统初步**

(注意：本次所有实验都在Linux中完成)

注：使用虚拟机时无法截图，因此实验报告中图片均为拍摄照片。

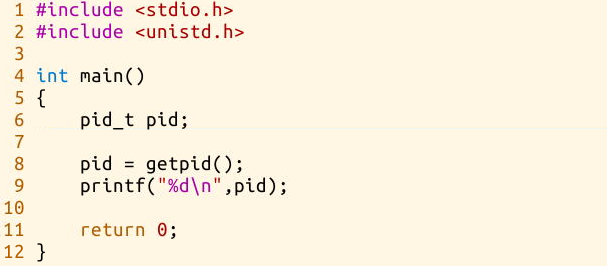
**一、（系统调用实验）了解系统调用不同的封装形式。**

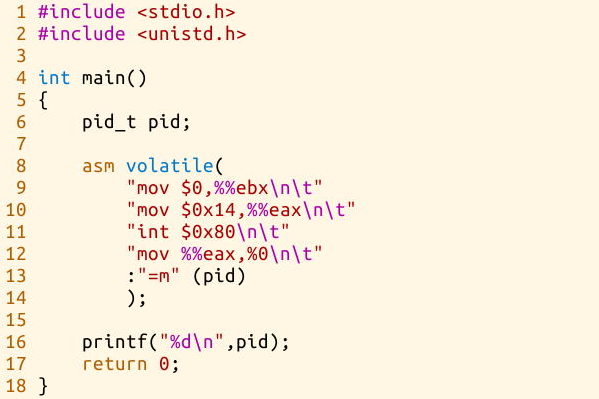
要求：

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？)。

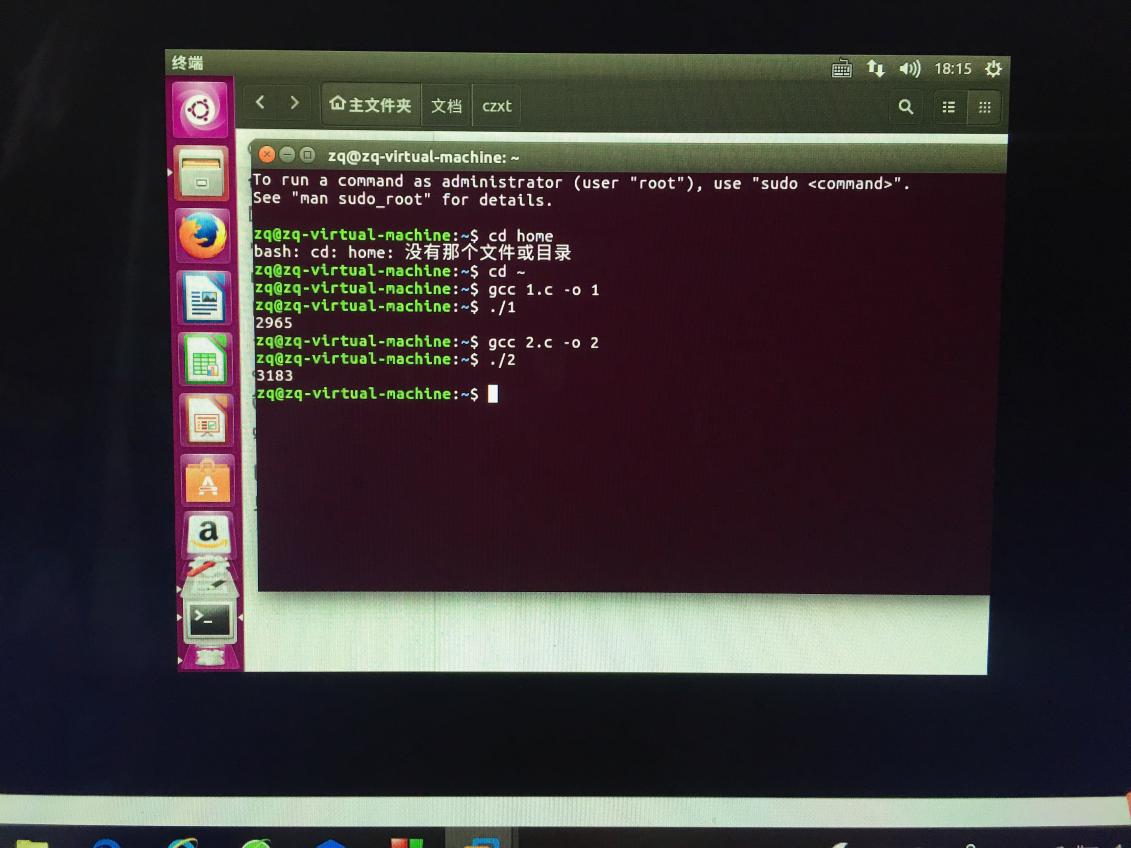




**实验步骤：**

在Linux上分别输入两个代码

运行结果如下：



因此，getpid的系统调用号为2965；Linux系统调用的中断向量号为80，系统调用号为3183。

1. 上机完成习题1.13。

命令 printf(“Hello World!\n”) 可归入一个{C标准函数、GNU C 函数库、Linux API}中哪一个或者哪几个？请分别用相应的linux系统调用的C函数形式和汇编代码两种形式来实现上述命令。

**实验步骤：**

1. 可归入C标准函数和GUN C函数库
2. **汇编**：

section .data

msg db "Hello, world!",0xA

len equ $ - msg

section .text

global \_start

\_start:

mov eax,4

mov ebx,1

mov ecx,msg

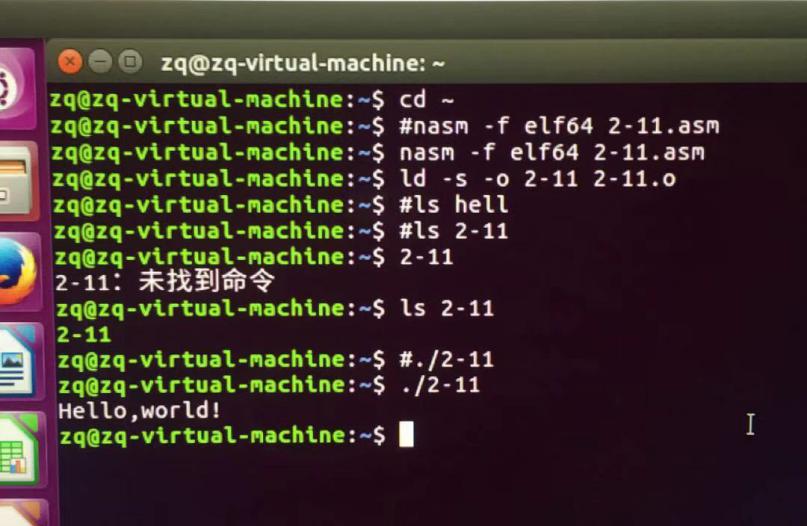
mov edx,len

int 0x80

mov eax,1

xor ebx,ebx

int 0x80



**Linux：**

#include <stdio.h>

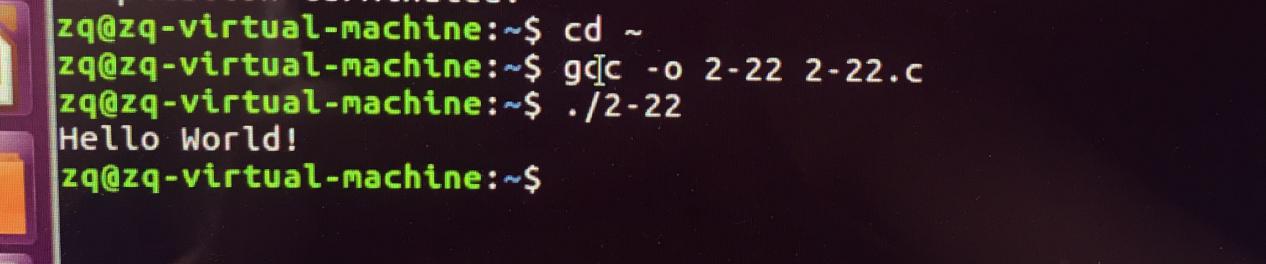
int main()

{

printf(“Hello World!\n”);

return 0;

}



3、阅读pintos操作系统源代码，画出系统调用实现的流程图。

<http://hgdcg14.blog.163.com/blog/static/23325005920152257504165/>

**实验步骤：**

操作系统各个数据结构初始化

创建空闲进程

中断设置并初始化系统时钟

中断开放使系统进入并发执行阶段

返回

创建其他进程，调度器按调度策略进行进程并发执行，如果无其他进程，则系统执行空闲进程

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

8

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);

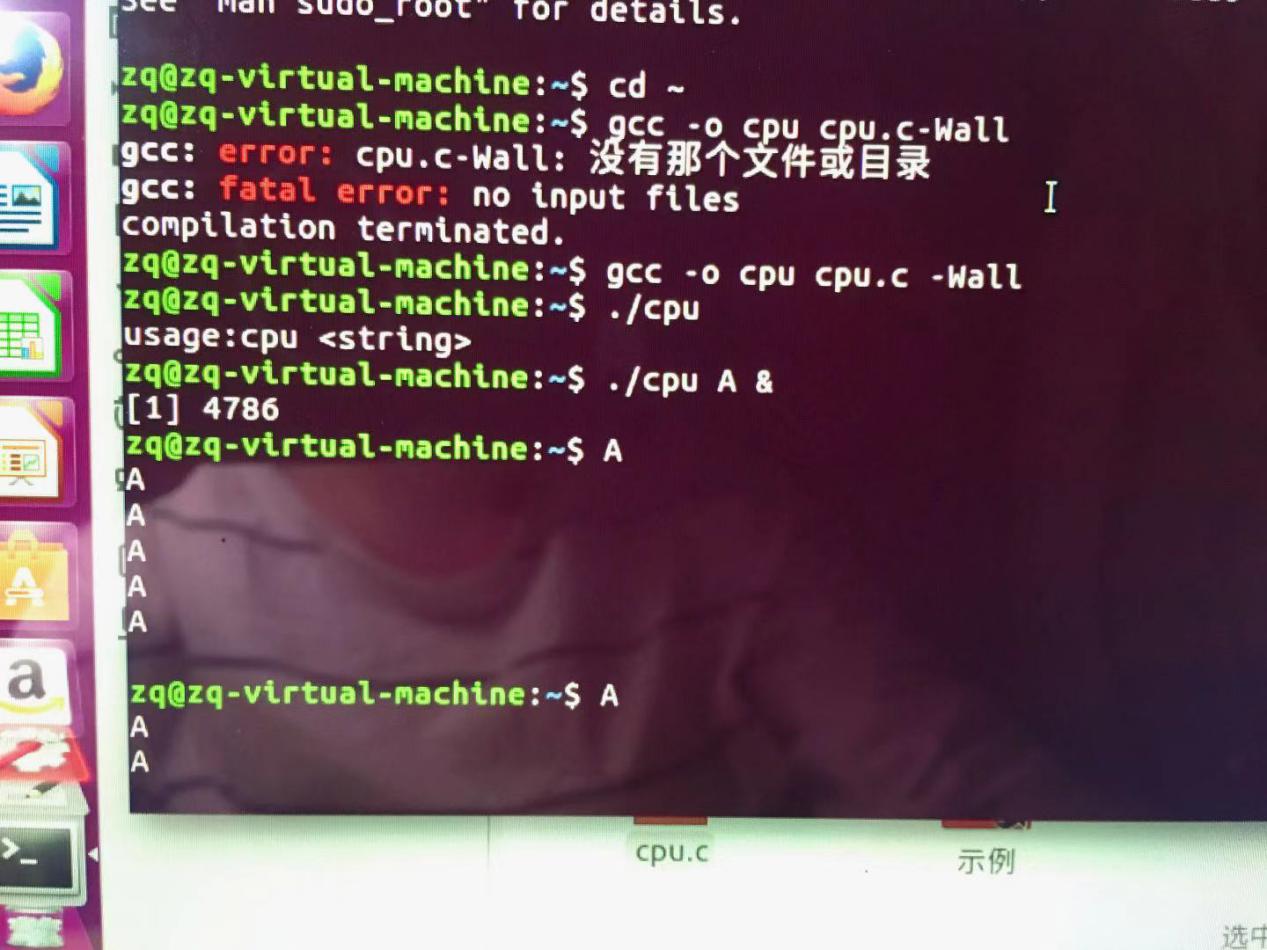
17 printf("%s\n", str);

18 }

1. return 0;

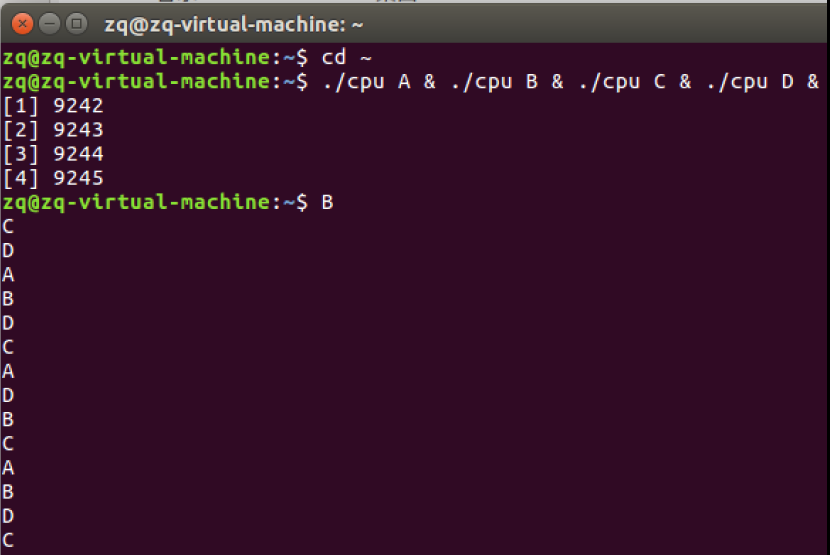
**实验步骤：**

（1）如图：



结果为：usage: cpu <string>。该程序的功能是：循环输出./cpu指令后的字母，如果没有字母，则输出usage：cpu <string>

（2）如图：



CPU一直循环输出字母，并且依次输出，每次顺序不一样。

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

7

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

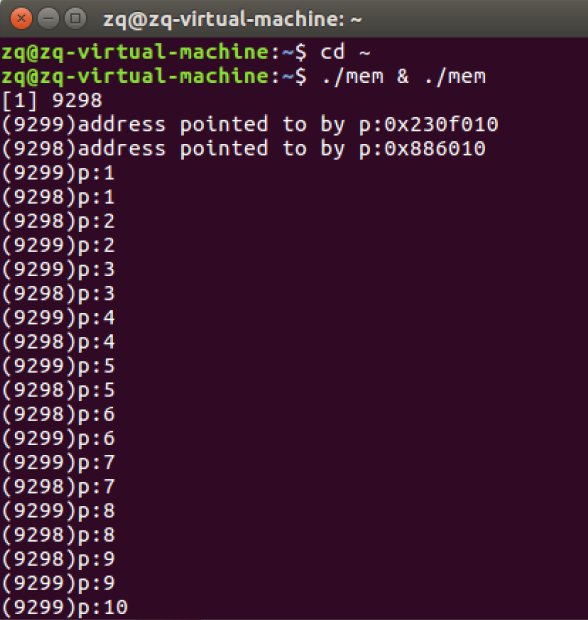
18 }

19 return 0;

**实验步骤：**

（1）如图：程序的功能是输出进程号和依次输出分配给p的地址

（2）两次分配的内存地址不同



**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);

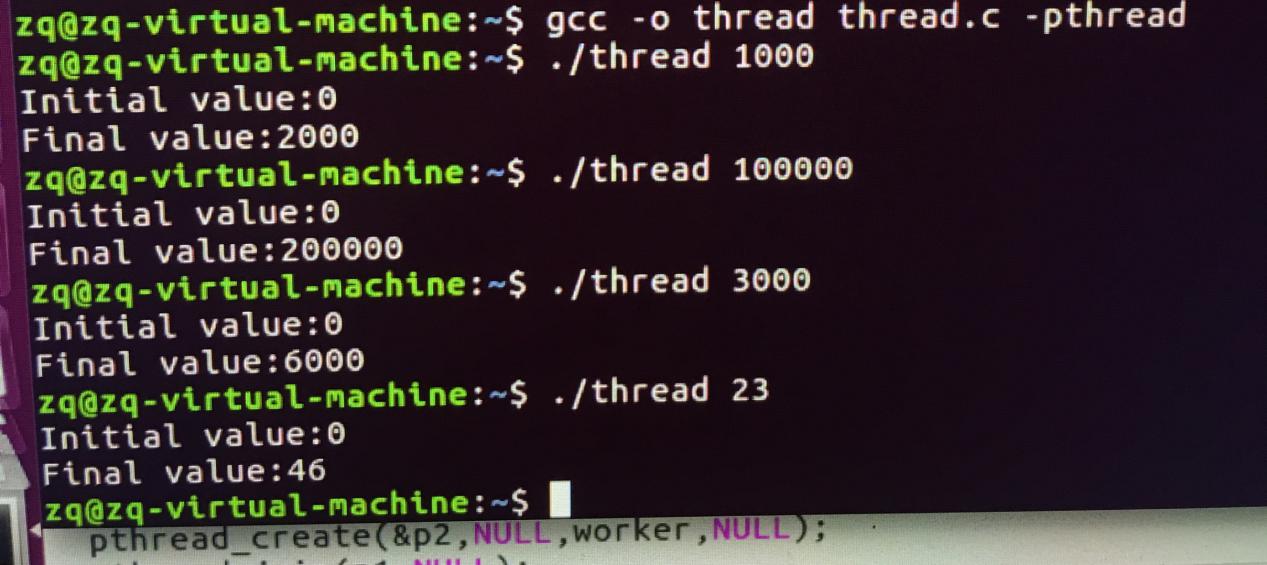
29 Pthread\_join(p1, NULL);

30 Pthread\_join(p2, NULL);

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

32 return 0;

**实验步骤：**



结果为2000，Initial value都为0，Final value均为输入的2倍。