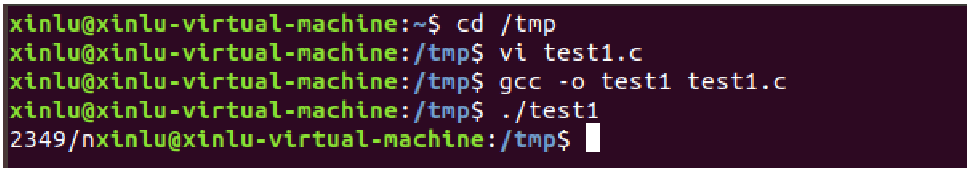
操作系统实验一：操作系统初步

16281259 鲁鑫

一、系统调用实验

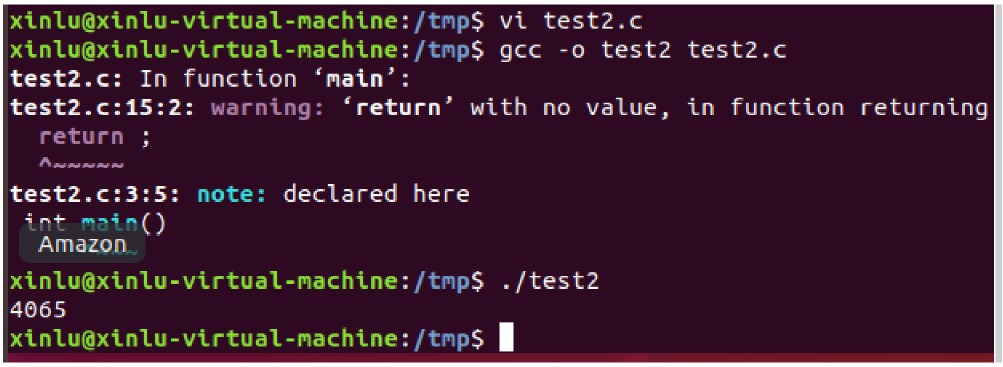
1. API 接口函数 getpid()直接调用



```
xinlu@xinlu-virtual-machine:~$ cd /tmp
xinlu@xinlu-virtual-machine:/tmp$ vi test1.c
xinlu@xinlu-virtual-machine:/tmp$ gcc -o test1 test1.c
xinlu@xinlu-virtual-machine:/tmp$ ./test1
2349/nxinlu@xinlu-virtual-machine:/tmp$ 
```

可以看到 getpid 的系统调用号是 2349

汇编中断调用



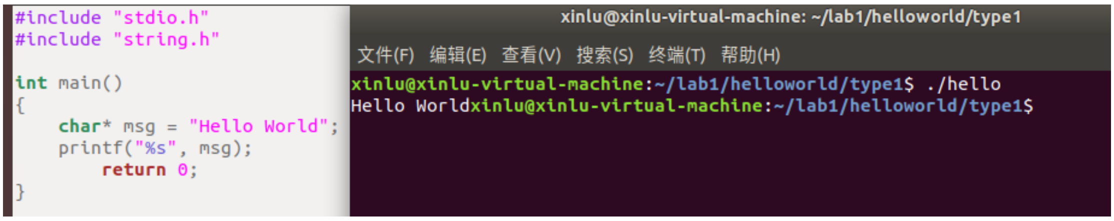
```
xinlu@xinlu-virtual-machine:/tmp$ vi test2.c
xinlu@xinlu-virtual-machine:/tmp$ gcc -o test2 test2.c
test2.c: In function ‘main’:
test2.c:15:2: warning: ‘return’ with no value, in function returning
  return ;
  ^~~~~~
test2.c:3:5: note: declared here
 int main()

xinlu@xinlu-virtual-machine:/tmp$ ./test2
4065
xinlu@xinlu-virtual-machine:/tmp$ 
```

可以看到 linux 系统调用的中断向量号是 4065

2. 上机完成习题 1.13.3

Linux 系统下的 c 语言实现



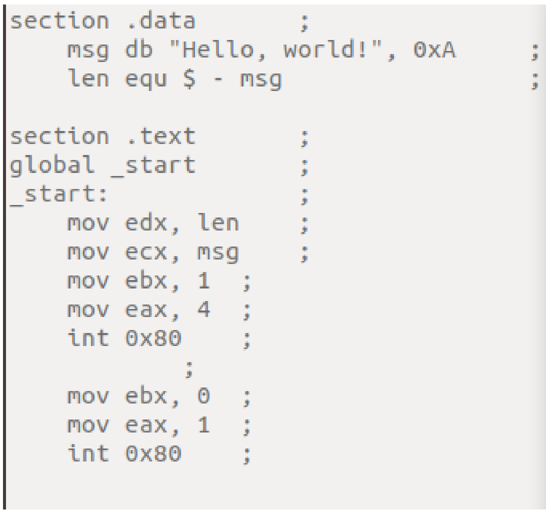
```
#include "stdio.h"
#include "string.h"

int main()
{
    char* msg = "Hello World";
    printf("%s", msg);
        return 0;
}
```

```
xinlu@xinlu-virtual-machine:~/lab1/helloworld/type1$ ./hello
Hello Worldxinlu@xinlu-virtual-machine:~/lab1/helloworld/type1$
```

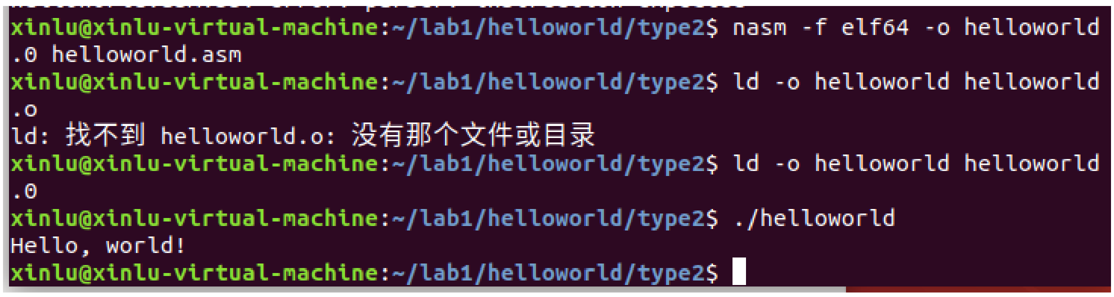
xinlu@xinlu-virtual-machine: ~/lab1/helloworld/type1
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

汇编代码实现



```
section .data        ;
    msg db "Hello, world!", 0xA      ;
    len equ $ - msg                  ;

section .text        ;
global _start        ;
_start:              ;
    mov edx, len     ;
    mov ecx, msg     ;
    mov ebx, 1   ;
    mov eax, 4   ;
    int 0x80     ;
            ;
    mov ebx, 0   ;
    mov eax, 1   ;
    int 0x80     ;
```



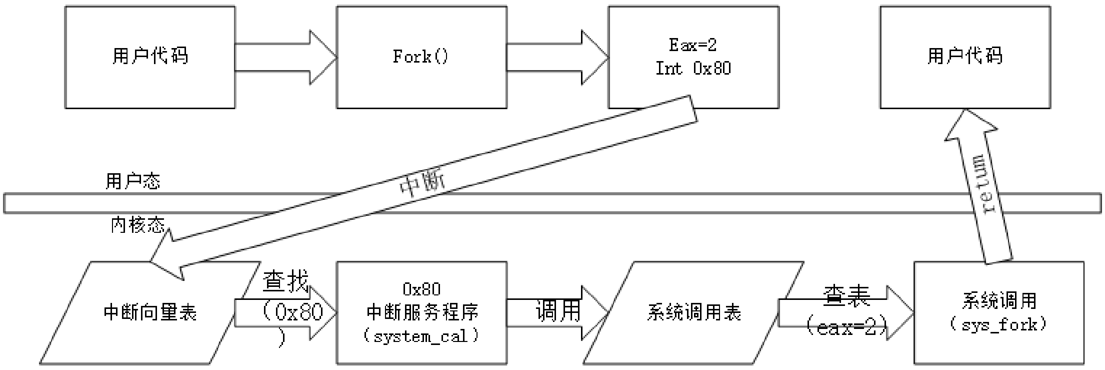
```
xinlu@xinlu-virtual-machine:~/lab1/helloworld/type2$ nasm -f elf64 -o helloworld
.0 helloworld.asm
xinlu@xinlu-virtual-machine:~/lab1/helloworld/type2$ ld -o helloworld helloworld
.o
ld: 找不到 helloworld.o: 没有那个文件或目录
xinlu@xinlu-virtual-machine:~/lab1/helloworld/type2$ ld -o helloworld helloworld
.0
xinlu@xinlu-virtual-machine:~/lab1/helloworld/type2$ ./helloworld
Hello, world!
xinlu@xinlu-virtual-machine:~/lab1/helloworld/type2$ 
```

因为没有学过汇编，所以这段代码是从网上搜索的，参照网上的操作过程，nasm -f elf64 -o helloworld.0 helloworld.asm 命令为编译这段汇编代码；

Ld -o helloworld helloworld.0 为链接；

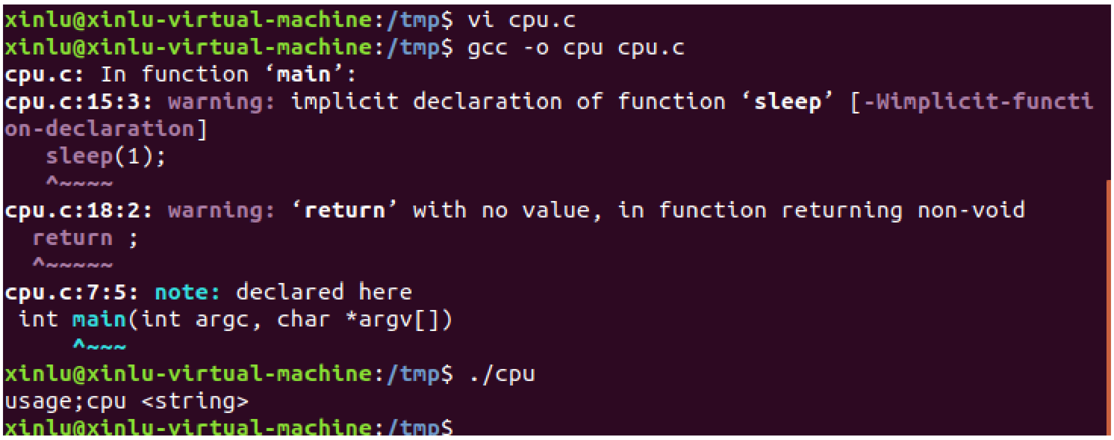
./helloworld 为运行。

3.画出系统调用实现的流程图



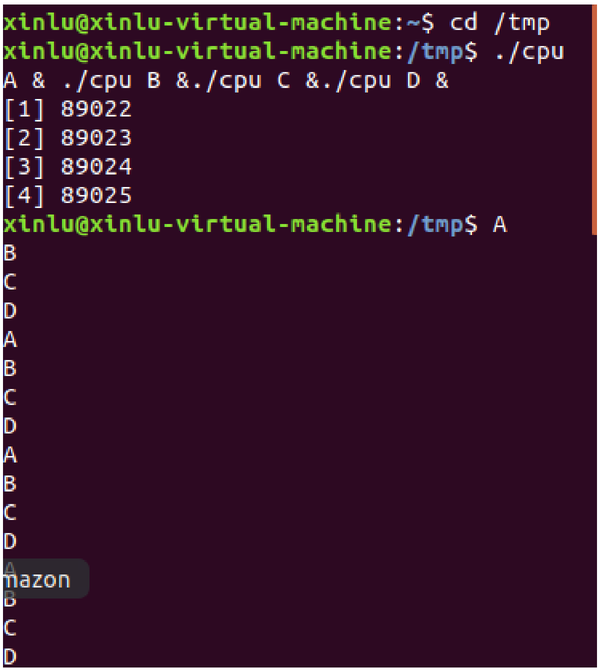
二、并发实验

1. 编译运行该程序（cpu.c），观察输出结果，说明程序功能



```
xinlu@xinlu-virtual-machine:/tmp$ vi cpu.c
xinlu@xinlu-virtual-machine:/tmp$ gcc -o cpu cpu.c
cpu.c: In function ‘main’:
cpu.c:15:3: warning: implicit declaration of function ‘sleep’ [-Wimplicit-functi
on-declaration]
   sleep(1);
   ^~~~~
cpu.c:18:2: warning: ‘return’ with no value, in function returning non-void
  return ;
  ^~~~~~
cpu.c:7:5: note: declared here
 int main(int argc, char *argv[])
     ^~~~
xinlu@xinlu-virtual-machine:/tmp$ ./cpu
usage;cpu <string>
xinlu@xinlu-virtual-machine:/tmp$
```

2.

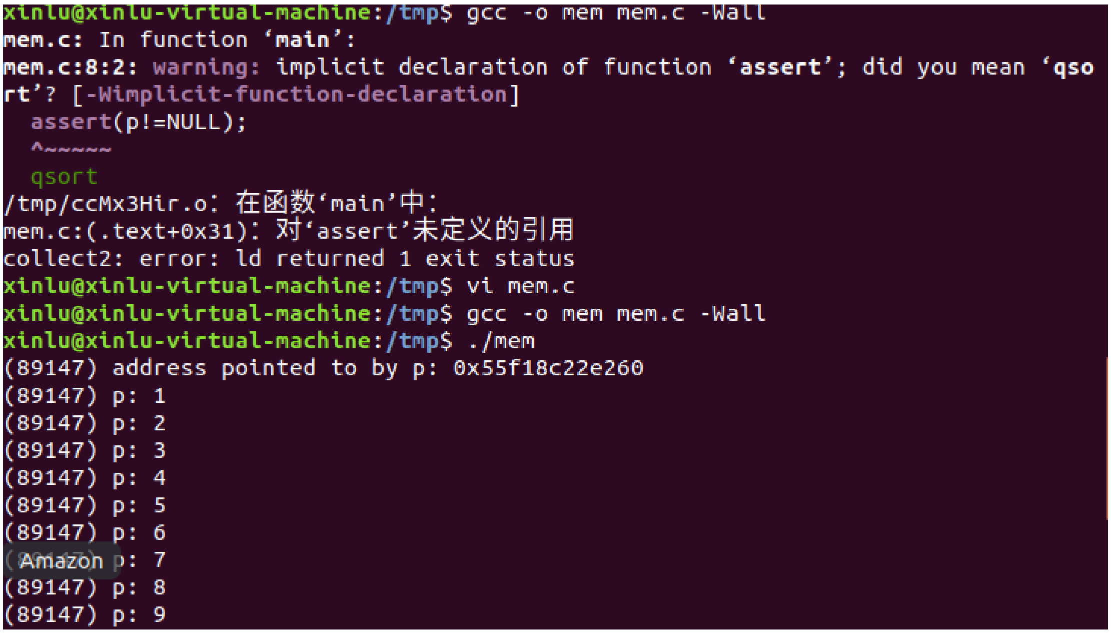


```
xinlu@xinlu-virtual-machine:~$ cd /tmp
xinlu@xinlu-virtual-machine:/tmp$ ./cpu
A & ./cpu B &./cpu C &./cpu D &
[1] 89022
[2] 89023
[3] 89024
[4] 89025
xinlu@xinlu-virtual-machine:/tmp$ A
B
C
D
A
B
C
D
A
B
C
D
A
mazon
B
C
D
```

程序 cpu 一直在运行。运行的顺序为谁 A B C D，也就是按照请求 cpu 的顺序依次进行。按照操作系统的并发性，在一段时间内，可以有多个程序运行。

三、内存分配实验

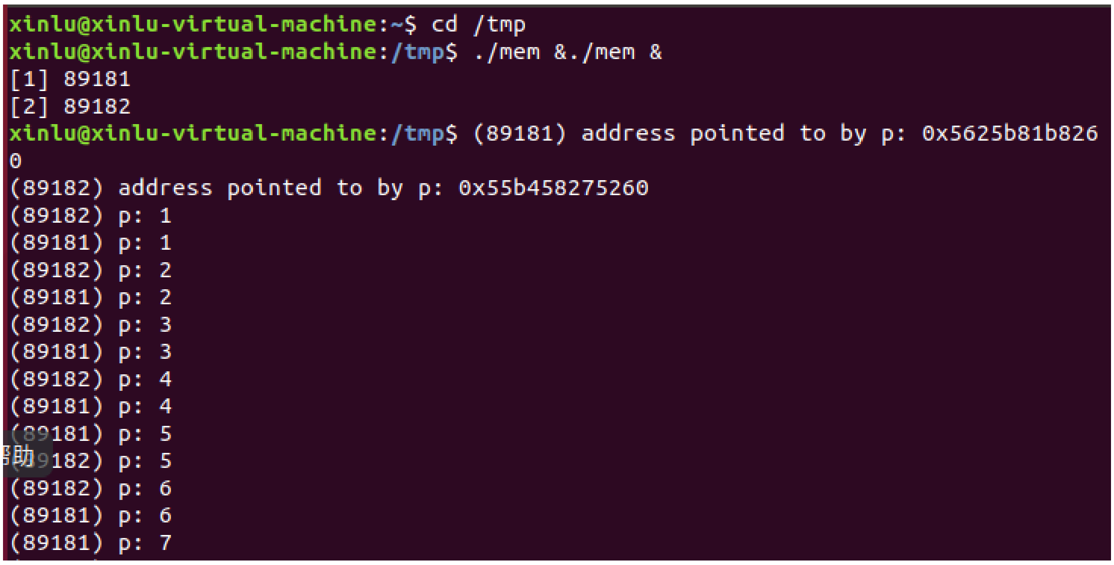
1. 阅读并编译运行该程序(mem.c)，观察输出结果，说明程序功能



```
xinlu@xinlu-virtual-machine:/tmp$ gcc -o mem mem.c -Wall
mem.c: In function ‘main’:
mem.c:8:2: warning: implicit declaration of function ‘assert’; did you mean ‘qso
rt’? [-Wimplicit-function-declaration]
  assert(p!=NULL);
  ^~~~~~
  qsort
/tmp/ccMx3Hir.o：在函数‘main’中：
mem.c:(.text+0x31)：对‘assert’未定义的引用
collect2: error: ld returned 1 exit status
xinlu@xinlu-virtual-machine:/tmp$ vi mem.c
xinlu@xinlu-virtual-machine:/tmp$ gcc -o mem mem.c -Wall
xinlu@xinlu-virtual-machine:/tmp$ ./mem
(89147) address pointed to by p: 0x55f18c22e260
(89147) p: 1
(89147) p: 2
(89147) p: 3
(89147) p: 4
(89147) p: 5
(89147) p: 6
(89147) p: 7
(89147) p: 8
(89147) p: 9
```

程序的功能是输出一个系统调用号对应的程序的运行结果，即循环输出一个指针对应系统分配的内存地址内存储的数据。

2. 再次按下面的命令运行并观察结果

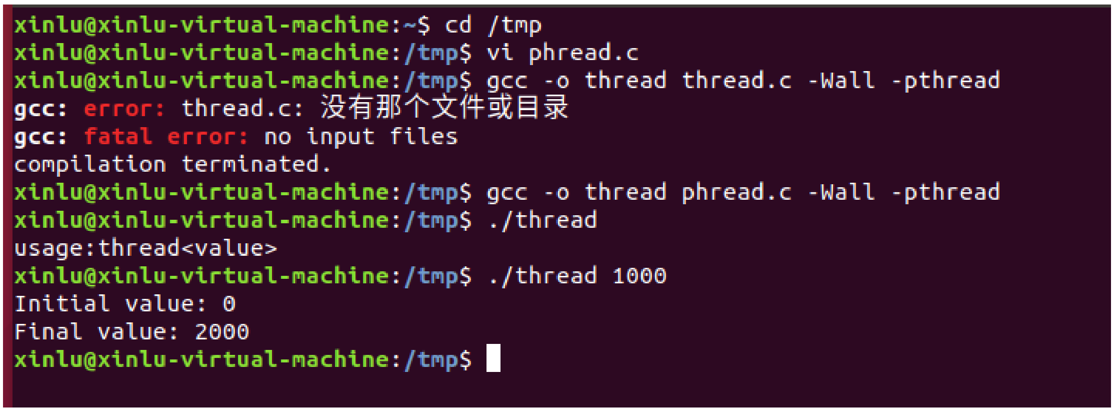


```
xinlu@xinlu-virtual-machine:~$ cd /tmp
xinlu@xinlu-virtual-machine:/tmp$ ./mem &./mem &
[1] 89181
[2] 89182
xinlu@xinlu-virtual-machine:/tmp$ (89181) address pointed to by p: 0x5625b81b826
0
(89182) address pointed to by p: 0x55b458275260
(89182) p: 1
(89181) p: 1
(89182) p: 2
(89181) p: 2
(89182) p: 3
(89181) p: 3
(89182) p: 4
(89181) p: 4
(89181) p: 5
(89182) p: 5
(89182) p: 6
(89181) p: 6
(89181) p: 7
```

两个分别运行的程序分配的内存地址不相同，但是共享一块物理内存区域，因为这两个程序是交替执行的，说明二者分配的物理内存区域是相同，系统按照两个程序对物理内存区域的请求依次调度。

四、共享的问题

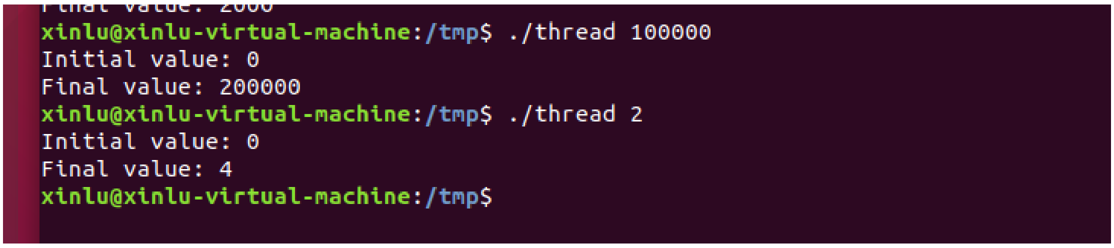
1. 阅读并编译运行该程序，观察输出结果，说明程序功能



```
xinlu@xinlu-virtual-machine:~$ cd /tmp
xinlu@xinlu-virtual-machine:/tmp$ vi phread.c
xinlu@xinlu-virtual-machine:/tmp$ gcc -o thread thread.c -Wall -pthread
gcc: error: thread.c: 没有那个文件或目录
gcc: fatal error: no input files
compilation terminated.
xinlu@xinlu-virtual-machine:/tmp$ gcc -o thread phread.c -Wall -pthread
xinlu@xinlu-virtual-machine:/tmp$ ./thread
usage:thread<value>
xinlu@xinlu-virtual-machine:/tmp$ ./thread 1000
Initial value: 0
Final value: 2000
xinlu@xinlu-virtual-machine:/tmp$
```

从运行结果可以看到程序的功能是把输入程序的值*2，然后输出。

2.



```
Final value: 2000
xinlu@xinlu-virtual-machine:/tmp$ ./thread 100000
Initial value: 0
Final value: 200000
xinlu@xinlu-virtual-machine:/tmp$ ./thread 2
Initial value: 0
Final value: 4
xinlu@xinlu-virtual-machine:/tmp$
```

3.loops,counter 变量是各个线程共享的，不会导致意想不到的问题。