

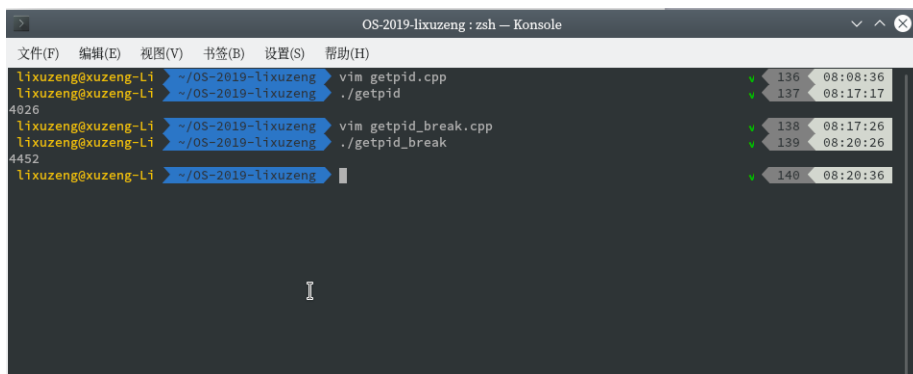
实验一：操作系统初步

李许增 16281042

操作环境：arch Linux

一、（系统调用实验）了解系统调用不同的封装形式。

1、程序运行结果：



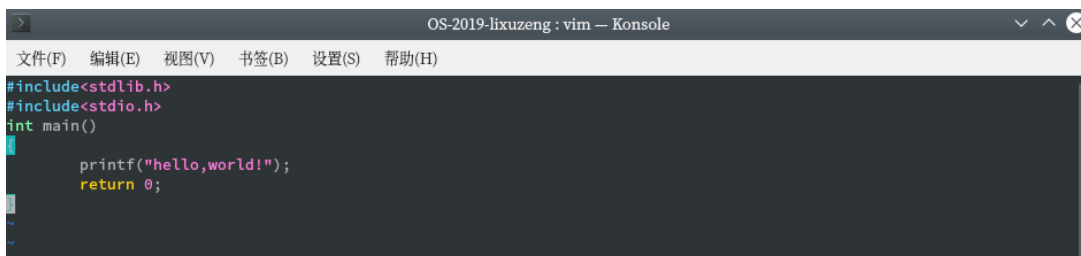
```
OS-2019-lixuzeng : zsh — Konsole
lixuzeng@lixuzeng-Li ~ /OS-2019-lixuzeng vim getpid.cpp
lixuzeng@lixuzeng-Li ~ /OS-2019-lixuzeng ./getpid
4026
lixuzeng@lixuzeng-Li ~ /OS-2019-lixuzeng vim getpid_break.cpp
lixuzeng@lixuzeng-Li ~ /OS-2019-lixuzeng ./getpid_break
4452
lixuzeng@lixuzeng-Li ~ /OS-2019-lixuzeng
```

getpid 系统调用号：0x14

Linux 系统调用的中断向量号：int 0x80

2、习题 1.13

（1）C 语言程序实现代码：



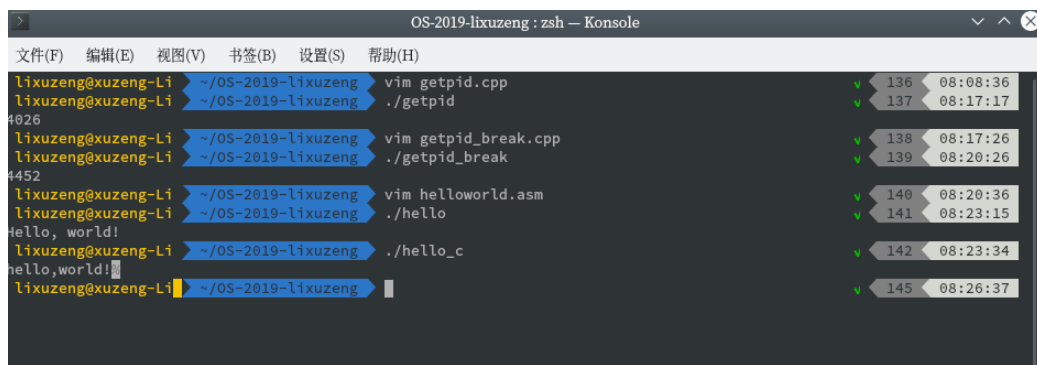
```
OS-2019-lixuzeng : vim — Konsole
#include<stdlib.h>
#include<stdio.h>
int main()
{
    printf("hello,world!");
    return 0;
}
```

（2）汇编实现代码：



```
OS-2019-lixuzeng : vim — Konsole
hello.asm
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
```

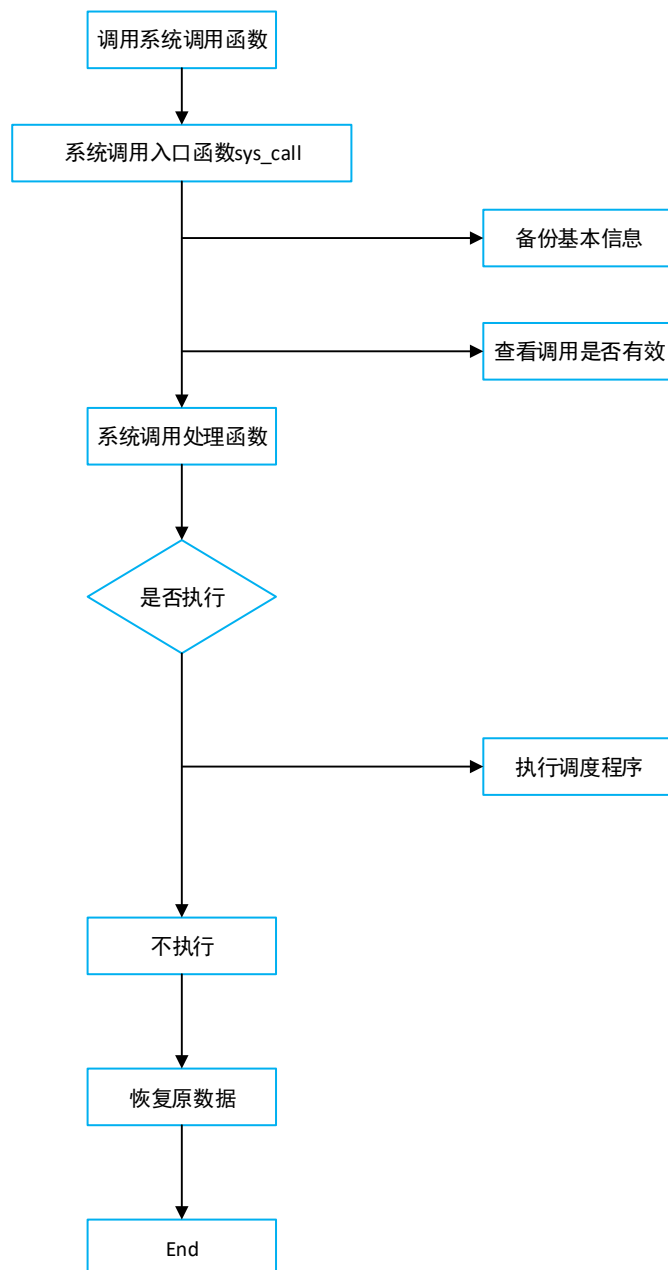
编译后运行结果：



```
OS-2019-lixuzeng: zsh — Konsole
文件(F) 编辑(E) 视图(V) 书签(B) 设置(S) 帮助(H)
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng vim getpid.cpp
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng ./getpid
4826
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng vim getpid_break.cpp
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng ./getpid_break
4452
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng vim helloworld.asm
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng ./hello
Hello, world!
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng ./hello_c
Hello, world!
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng
```

Line	Time
136	08:08:36
137	08:17:17
138	08:17:26
139	08:20:26
140	08:20:36
141	08:23:15
142	08:23:34
145	08:26:37

3、系统调用实现流程图：



二、（并发实验）根据以下代码完成下面的实验。

程序编译运行结果：

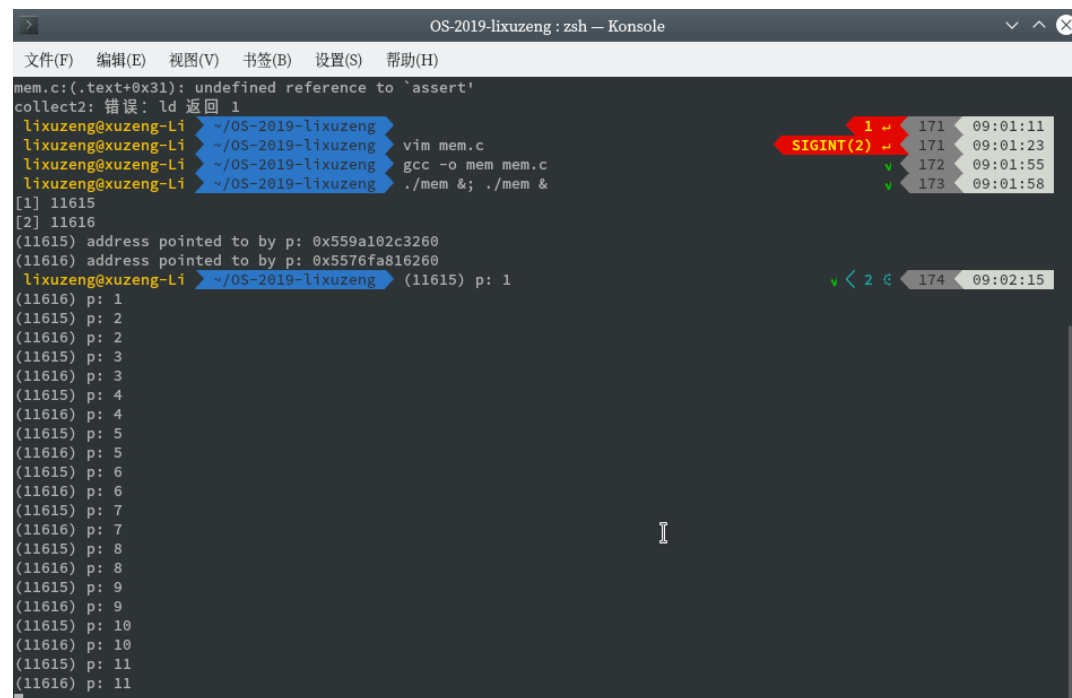


```
OS-2019-lixuzeng : zsh — Konsole
文件(F) 编辑(E) 视图(V) 书签(B) 设置(S) 帮助(H)
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng > ./cpu
usage: cpu <string>
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng > ./cpu 2 A
usage: cpu <string>
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng > vim cpu.c
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng > gcc -o cpu cpu.c
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng > ./cpu
usage: cpu <string>
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng > ./cpu A & ; ./cpu B & ; ./cpu C & ; ./cpu D &
[1] 10604
[2] 10605
[3] 10606
[4] 10607
lixuzeng@xuzeng-Li ~/OS-2019-lixuzeng > A
B
C
D
A
B
C
D
A
B
C
D
A
B
C
D
```

- 1、执行命令后提示缺少参数，程序通过 C 语言” printf” 函数实现调用 cpu 的功能
- 2、cpu 运行了一次，每次调用输出顺序不变都是 A, B, C, D，并且同时输出，证明多个程序运行是在同一时间间隔内，但是他们输出的顺序没有改变 cpu 在这个时间间隔内运行程序还是有顺序的，这个现象符合并发的概念。Linux 为分时操作系统，所以程序执行有时间先后。

三、（内存分配实验）根据以下代码完成实验。

程序运行结果



```
OS-2019-lixuzeng : zsh — Konsole
文件(F) 编辑(E) 视图(V) 书签(B) 设置(S) 帮助(H)
mem.c:(.text+0x31): undefined reference to `assert'
collect2: 错误: ld 返回 1
lixuzeng@xuzeng-Li ~ /OS-2019-lixuzeng vim mem.c
lixuzeng@xuzeng-Li ~ /OS-2019-lixuzeng gcc -o mem mem.c
lixuzeng@xuzeng-Li ~ /OS-2019-lixuzeng ./mem & ./mem &
[1] 11615
[2] 11616
(11615) address pointed to by p: 0x559a102c3260
(11616) address pointed to by p: 0x5576fa816260
lixuzeng@xuzeng-Li ~ /OS-2019-lixuzeng (11615) p: 1
(11616) p: 1
(11615) p: 2
(11616) p: 2
(11615) p: 3
(11616) p: 3
(11615) p: 4
(11616) p: 4
(11615) p: 5
(11616) p: 5
(11615) p: 6
(11616) p: 6
(11615) p: 7
(11616) p: 7
(11615) p: 8
(11616) p: 8
(11615) p: 9
(11616) p: 9
(11615) p: 10
(11616) p: 10
(11615) p: 11
(11616) p: 11
```

- 1、程序观察程序运行时变量被分配的内存地址。
- 2、两次运行程序分配的内存地址不相同（因为变量 p 分配的地址不同），不共享同一块物理内存区域，两个程序分配的 pid 不同所以程序不在同一个内存区域上。

四、（共享的问题）根据以下代码完成实验。

运行结果：

```
OS-2019-lixuzeng : zsh -- Konsole
文件(F) 编辑(E) 视图(V) 书签(B) 设置(S) 帮助(H)
collect2: 错误: ld 返回 1
lixuzeng@xuzeng-Li ~ > gcc -o thread thread.c -lpthread
lixuzeng@xuzeng-Li ~ > ./thread 1000
Initial value : 0
Final value : 2000
lixuzeng@xuzeng-Li ~ > ./thread 100000
Initial value : 0
Final value : 115326
lixuzeng@xuzeng-Li ~ > ./thread 1002
Initial value : 0
Final value : 2004
lixuzeng@xuzeng-Li ~ > ./thread 1006
Initial value : 0
Final value : 2012
lixuzeng@xuzeng-Li ~ > ./thread 2009
Initial value : 0
Final value : 4018
lixuzeng@xuzeng-Li ~ > ./thread 10000
Initial value : 0
Final value : 16718
lixuzeng@xuzeng-Li ~ > ./thread 5000
Initial value : 0
Final value : 6124
lixuzeng@xuzeng-Li ~ > ./thread 3000
Initial value : 0
Final value : 5665
lixuzeng@xuzeng-Li ~ > ./thread 100
Initial value : 0
Final value : 200
lixuzeng@xuzeng-Li ~ > ./thread 1000
Initial value : 0
Final value : 2000
lixuzeng@xuzeng-Li ~ > ./thread 2500
Initial value : 0
Final value : 5000
lixuzeng@xuzeng-Li ~ > ./thread 2700
Initial value : 0
Final value : 5400
lixuzeng@xuzeng-Li ~ > ./thread 2900
Initial value : 0
Final value : 5800
lixuzeng@xuzeng-Li ~ > ./thread 3000
Initial value : 0
Final value : 6000
lixuzeng@xuzeng-Li ~ >
```

- 1、程序可以在执行过程中创建两条线程进行运行，并且调用共享的变量以观察多线程调用的结果
- 2、执行结果程序运行中开了两个线程所以 counter 的输出应该改为输入数的二倍加入输入为 n，则输出应该为 2n，带式实际结果出现为部分输出小于 2n 当然输出结果不可能大于 2n。结果出现小于 2n 的证明多线程调用初夏你出了问题，发生了冲突。
- 3、线程共享变量：loops； counter； 会导致意想不到的问题，事实证明多线程调用可能存在数据丢失或者数据出错的问题。