

# 操作系统实验报告一

计科 1601 修宇婷 16281051

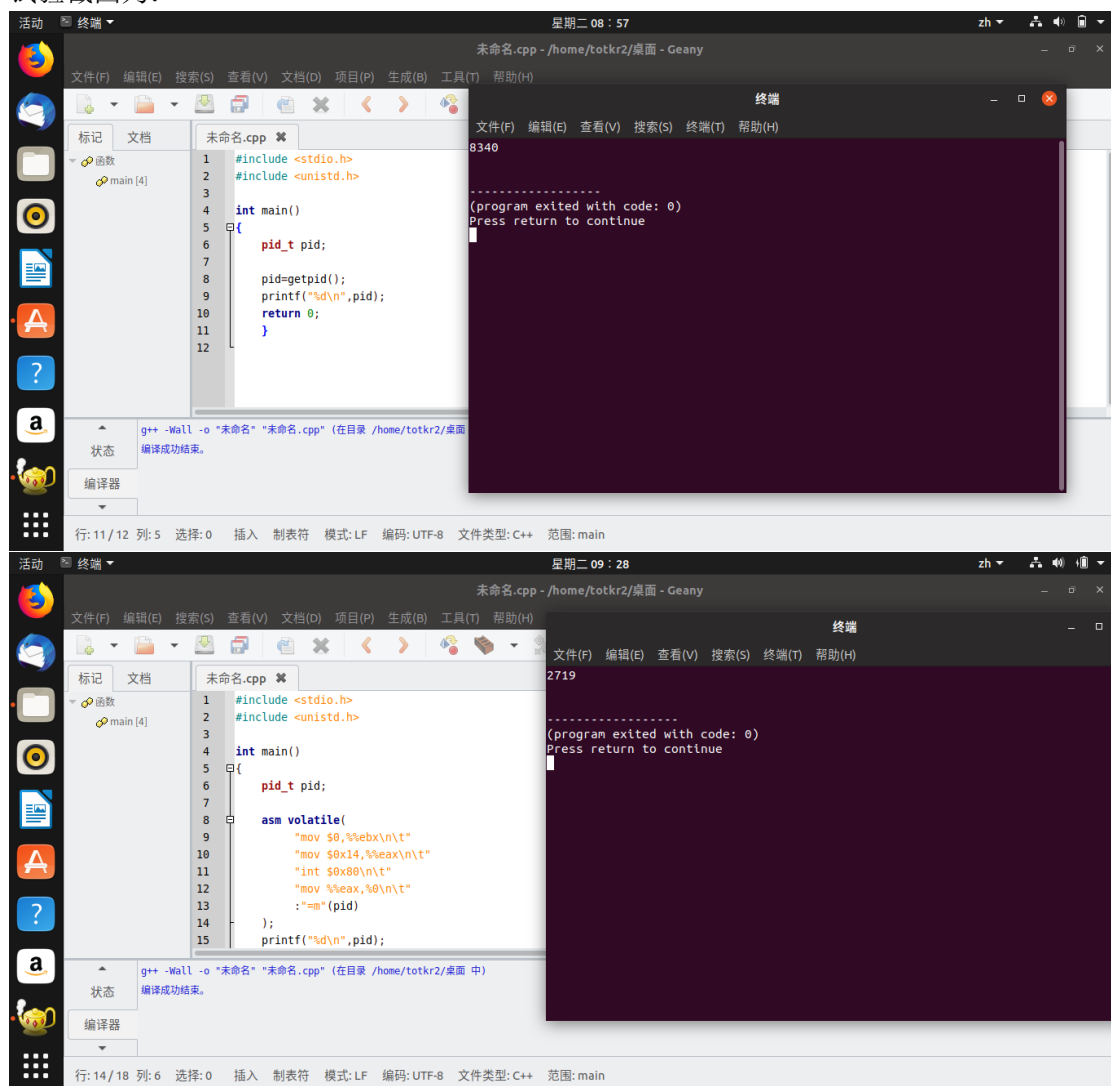
## 一、（系统调用实验）了解系统调用不同的封装形式。

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？）。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。

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

(1)、源代码为 `getpid1.c` 和 `getpid2.c`

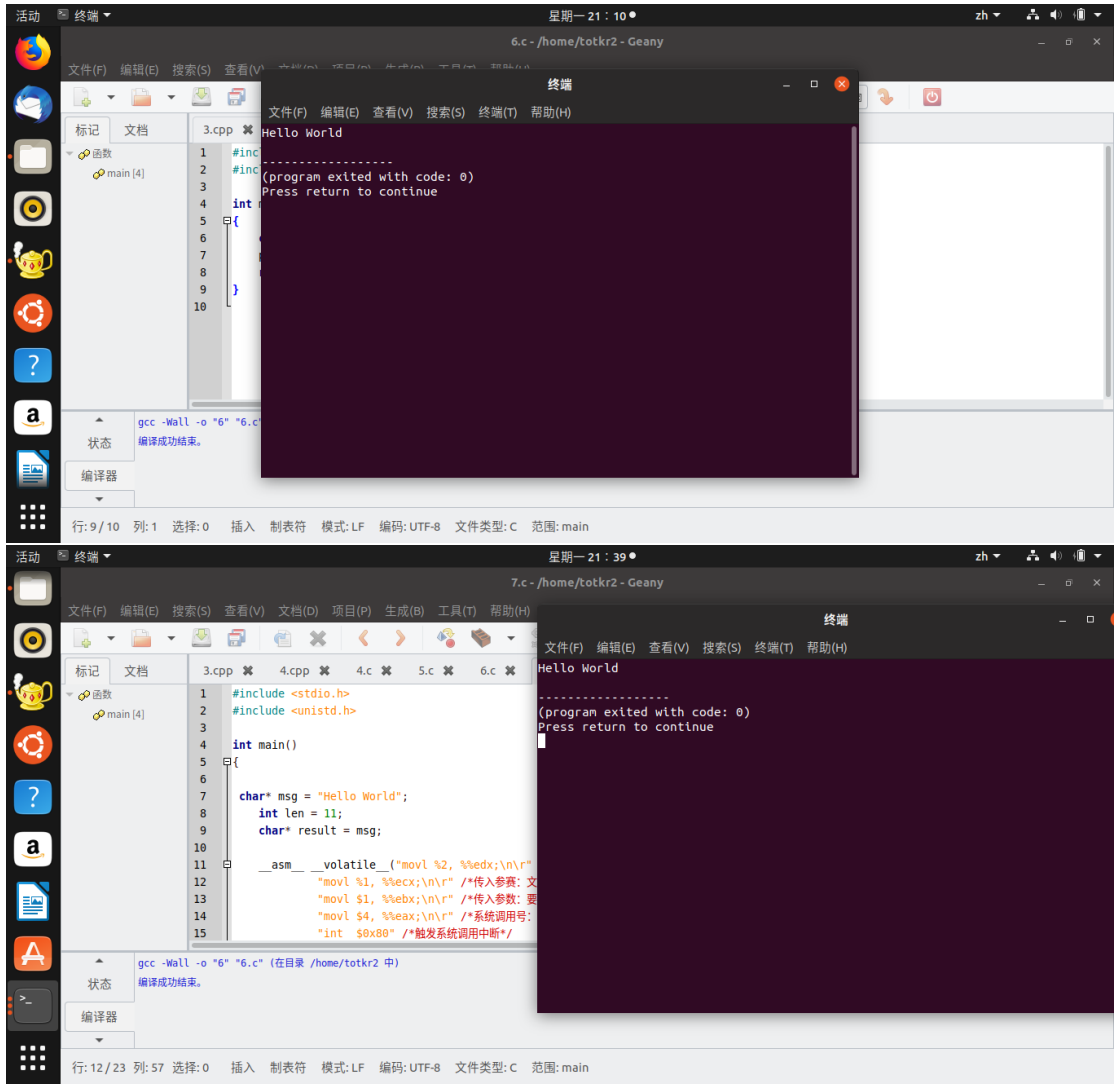
试验截图为：



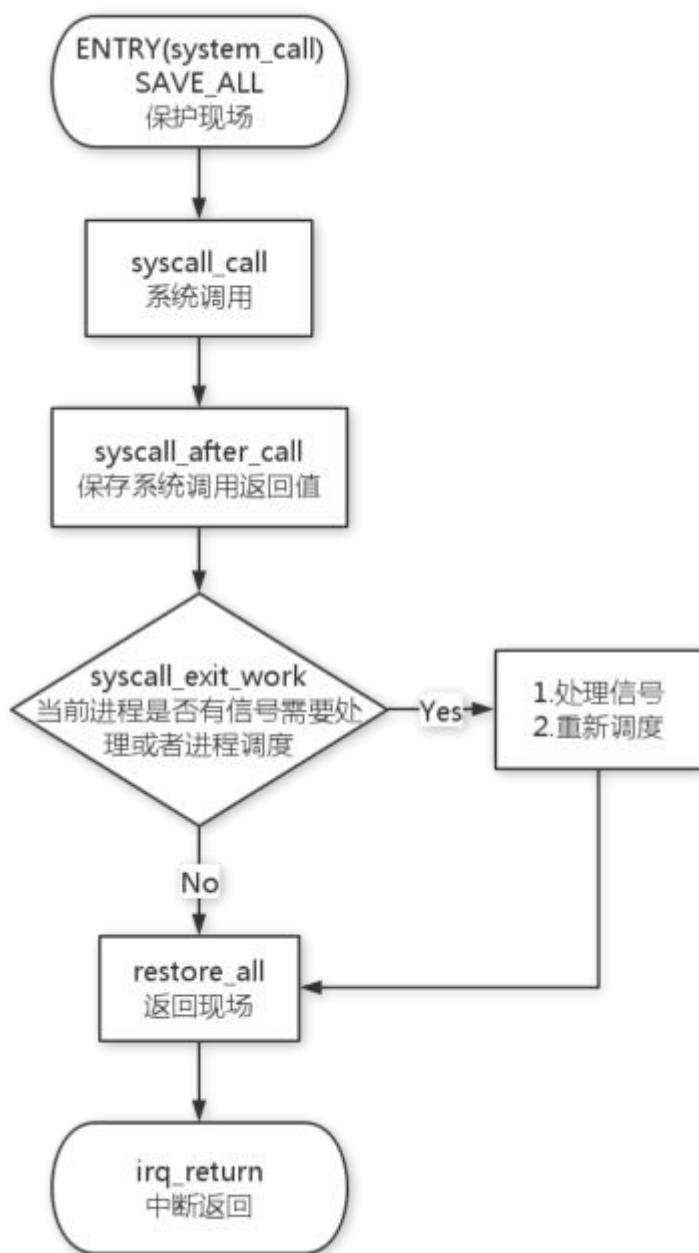
`Getpid()` 系统调用号为 `0x14`，系统调用中断向量号为 `0x80`

(2)、1.13 源代码为 helloworld1.c, helloworld2.c

截图为:



(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。

要求：

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`）

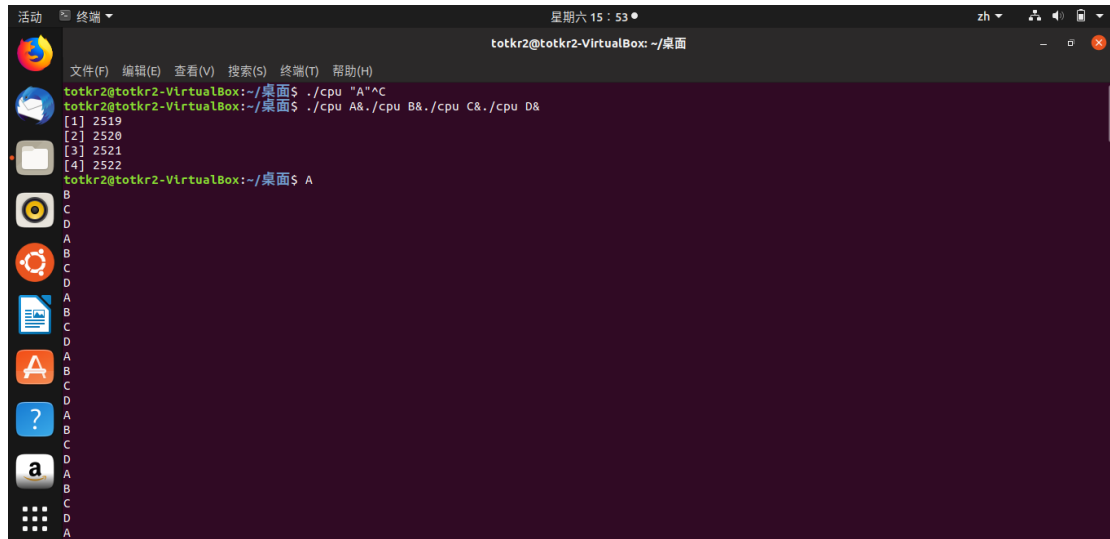
2、再次按下面的运行并观察结果：执行命令： `./cpu A & ; ./cpu B & ; ./cpu C & ; ./cpu`

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。

源代码见 `cpu.c`

试验截图：



这个程序所做的是调用 `Sleep()`，一个反复检查时间并一旦运行一秒钟就返回的函数。然后，它打印出用户在命令行上输入的字符串，并一直重复。，这是虚拟化 CPU，运行了 4 次，顺序是随机的

### 三、（内存分配实验）根据以下代码完成实验。

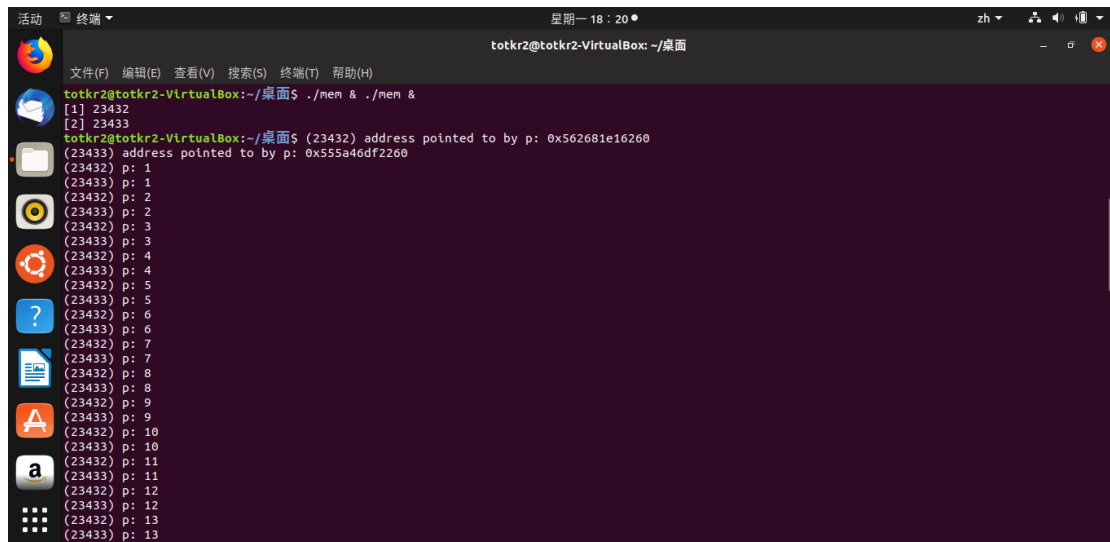
要求：

2、 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）、源代码为 `mem.c`

实验截图：



```
totkr2@totkr2-VirtualBox: ~/桌面$ ./mem & ./mem &
[1] 23432
[2] 23433
totkr2@totkr2-VirtualBox:~/桌面$ (23432) address pointed to by p: 0x562681e16260
(23432) address pointed to by p: 0x555a40df2260
(23432) p: 1
(23433) p: 1
(23432) p: 2
(23433) p: 2
(23432) p: 3
(23433) p: 3
(23432) p: 4
(23433) p: 4
(23432) p: 5
(23433) p: 5
(23432) p: 6
(23433) p: 6
(23432) p: 7
(23433) p: 7
(23432) p: 8
(23433) p: 8
(23432) p: 9
(23433) p: 9
(23432) p: 10
(23433) p: 10
(23432) p: 11
(23433) p: 11
(23432) p: 12
(23433) p: 12
(23432) p: 13
(23433) p: 13
```

(2)、分析：现代机器提供的物理内存模型非常简单。内存只是一个字节数组；要读取内存，必须指定一个地址才能访问存储在那里的数据；要写入（或更新）内存，还必须指定要写入给定地址的数据。

程序运行时始终访问内存。一个程序在内存中保存它的数据结构，并在执行工作时通过各种指令进行访问，例如加载、存储、或其他访问内存的显式指令。不要忘记，程序的每个指令也都在内存中；因此，在每次取指令时访问内存。

先分配一些内存。然后，它打印出内存地址，然后将数字 0 放入新分配的内存的第一个位置。延迟一秒并递增存储在 p 中保存的地址的值。对于每个 print 语句，打印出正在运行的程序的进程标识符（PID）。

操作系统虚拟化内存。每个进程访问自己的私有虚拟地址空间，操作系统以某种方式映射到机器的物理内存。一个正在运行的程序中的内存引用不会影响其他进程的地址空间；就运行程序而言，它拥有所有的物理内存。然而，现实是物理内存是由操作系统管理的共享资源。

(3)、所以分配的内存空间不同，但是两个程序共享同一块物理内存区域

## 四、（共享的问题）根据以下代码完成实验。

要求：

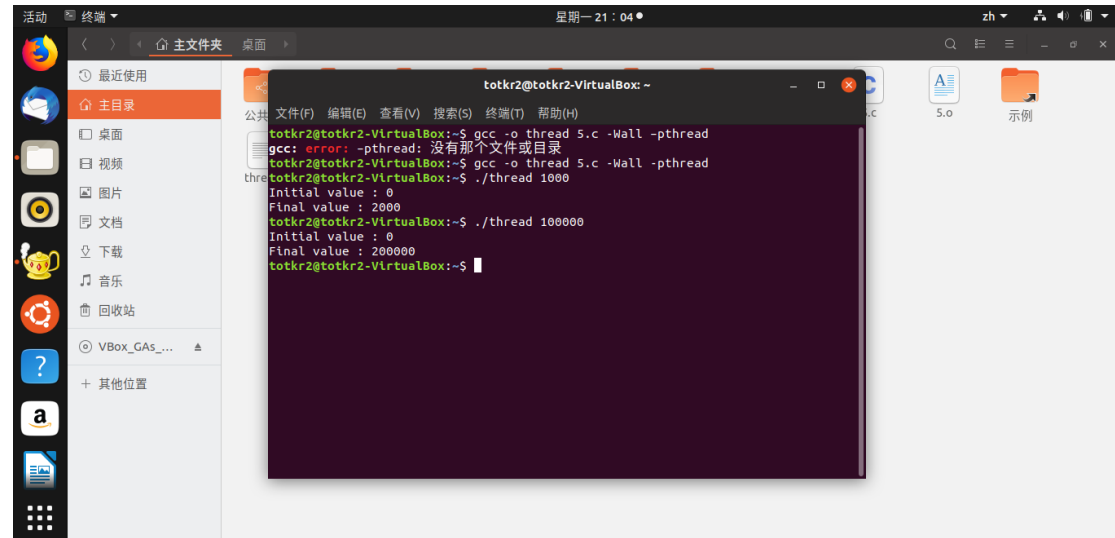
(1) 阅读并编译运行该程序，观察输出结果，说明程序功能。（编译命令：gcc -o thread thread.c -Wall -pthread）（执行命令 1：./thread 1000）

(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。）

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。

源代码为 thread.c

实验截图：



```
totkr2@totkr2-VirtualBox: ~  
totkr2@totkr2-VirtualBox:~$ gcc -o thread 5.c -Wall -pthread  
gcc: error: -pthread: 没有那个文件或目录  
totkr2@totkr2-VirtualBox:~$ gcc -o thread 5.c -Wall -pthread  
totkr2@totkr2-VirtualBox:~$ ./thread 1000  
Initial value : 0  
Final value : 2000  
totkr2@totkr2-VirtualBox:~$ ./thread 100000  
Initial value : 0  
Final value : 200000  
totkr2@totkr2-VirtualBox:~$
```

主程序使用 Pthread .create() 创建两个线程。 您可以将线程视为在与其他函数相同的内存空间中运行的函数，其中一次激活多个函数。 在这个例子中，每个线程开始在一个名为 worker() 的例程中运行，在该例程中，它只是循环递增一个统计循环次数的计数器。循环的值确定两个工作程序中的每一个将在循环中递增共享计数器的次数，所以，当循环的输入值设置为 N 时，我们期望程序的最终输出为 2N。