系统调用（system call）是操作系统提供给程序调用的接口，程序调用系统调用完成所需的系统任务。

系统调用类似于我们平常使用的工具包。当你需要钳子时，你就从工具包中拿出钳子使用。当你需要老虎钳时，你就从工具包中拿出老虎钳使用。当你需要螺丝刀，你就从工具包中拿出螺丝刀使用……系统调用也是同样的原理，当你需要获取当前时间时，你可以调用获取当前时间的系统调用。当你需要创建文件时，你可以调用创建文件的系统调用。系统调用提供了许多不同的功能，你可以调用它们，让操作系统帮你完成所需的功能。

获取时间

#include <stdio.h>

#include <time.h>

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

{

long tm=time(NULL);

printf("time = %d \n",tm):

return 0;

}

1

2

3

4

5

6

7

8

9

10

在明白系统调用的用处后，应该对系统调用有更深入的了解。

操作系统在运行每个程序时，都会为运行的程序提供4G的内存空间。(32位系统下。这里的内存空间指的是虚拟内存空间，不是实际的物理内存空间。虚拟内存将在以后讲解。这里暂时可以认为操作系统提供了4G物理内存给程序。)4G内存空间中0-3G属于用户程序，3G以上属于操作系统内核。3G以上空间存放了 内核的代码与数据。这其中包含了一系列”特殊”的函数。。

这些”特殊”的函数是系统调用对应的内核函数。系统调用被调用后，操作系统将执行系统调用对应的内核函数，也就是说系统调用的实现是由内核函数完成的。系统调用实质上就是保存在内核中的函数。0-3G的内存空间保存了用户程序的代码与数据。用户程序可以执行特殊的机器指令，切入内核，执行系统调用对应的内核函数。一般该指令是int 0x80。内核执行完成后，将返回用户空间，继续执行下一条指令。

当应用程序需要请求内核服务(例如读文件read，创建进程fork，终止进程exit或加载新程序execve)时，就需要使用系统调用。系统调用对程序员来说就像使用函数调用一样，我们可以简单的理解其为一种提供程序到内核的像函数一样的接口。