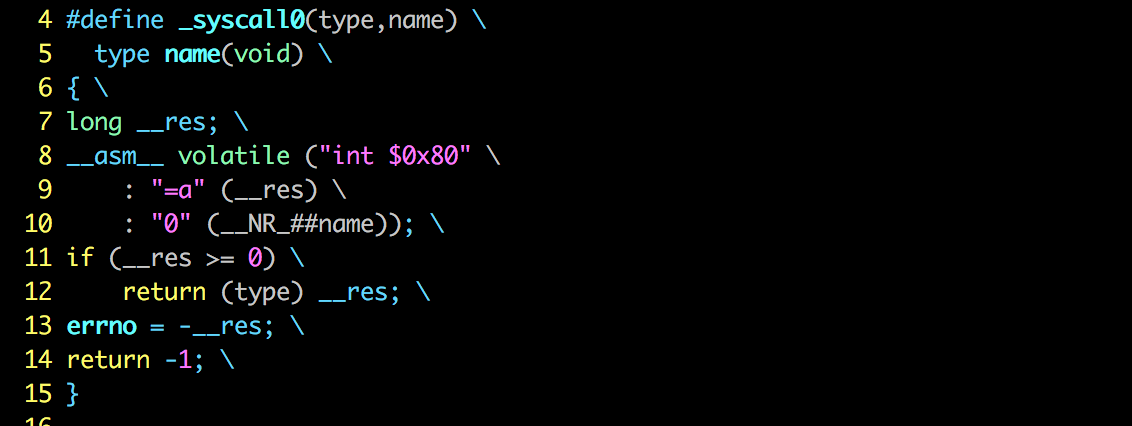
**Q:**

1. 从Linux 0.11现在的机制看，它的系统调用最多能传递几个参数？你能想出办法来扩大这个限制吗？

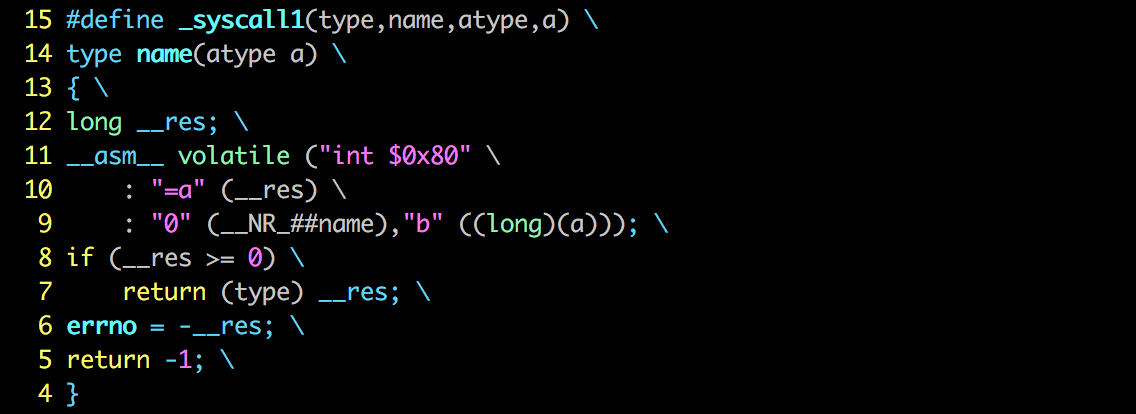
**A:**

从include/unistd.h中我们可以看到\_syscall的几个相关宏定义，截图如下

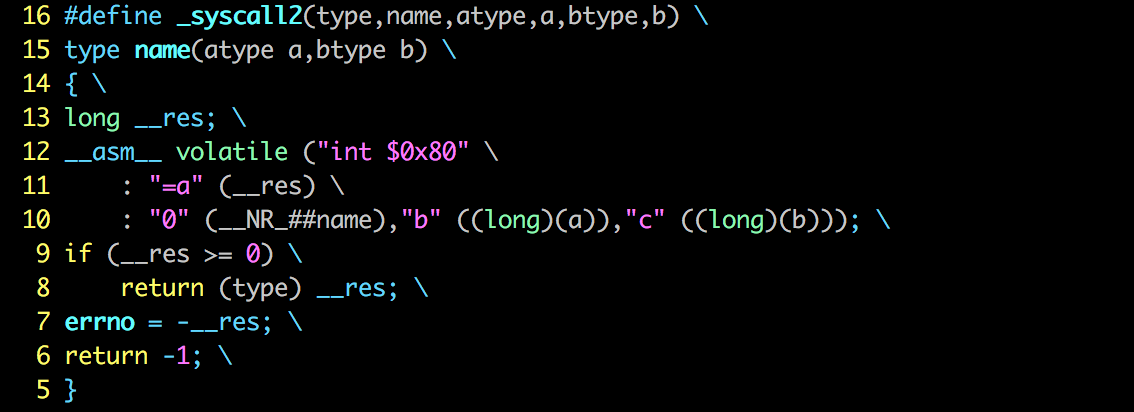
a）无参数版本



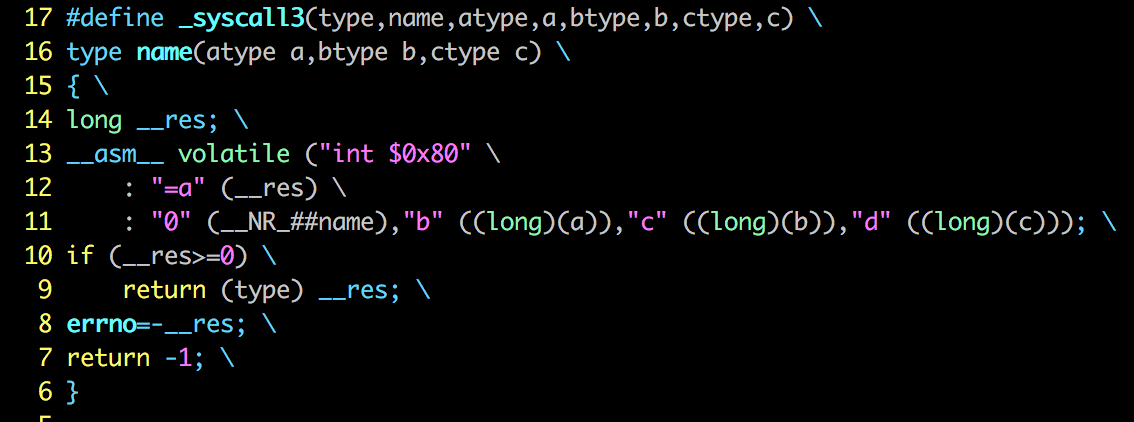
b）一个参数版本



c)两个参数版本



d)三个参数版本



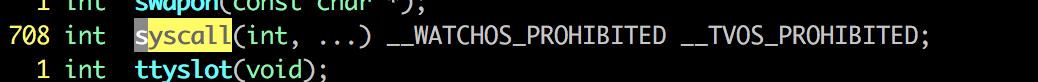
所以从Linux0.11 现在的机制来看，它的系统调用最多能传递3个参数。

扩大限制的方法：

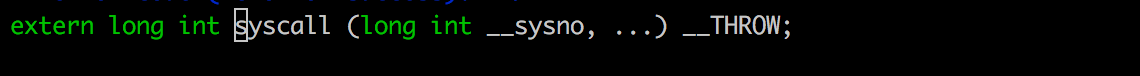
1. 当系统调用参数少于6个的时候，可以直接使用寄存器进行传递参数，在80x86寄存器的条件下，可以使用以下寄存器存放系统调用号和系统调用参数：eax(存放系统调用号)，ebx,ecx,edx,esi,edi,ebp。
2. 当系统调用参数多于6个时（受限于寄存器个数，不能直接寄存器进行传递参数）。这种情况下，用一个单独的寄存器指向进程地址空间中这些参数值所在的一个内存区。（编程者不用关心这个工作区，就像C调用一样，当调用封装例程时，参数自动被保存在栈中，封装例程找到合适的方式把参数传递给内核），这样做的话，当有一个参数指定的是地址，内核就必须检查它是否在这个进程的地址空间之内。

实际上现代的类Unix内核废除了\_syscall宏，而采用syscall函数，其接受一个可变参数。原理类似上述。

＃＃MacOS 中syscall的原型：



＃＃Ubuntu中syscall 的原型：



Q：

用文字简要描述向Linux 0.11添加一个系统调用foo()的步骤。

A:

1. 修改 include/linux/sys.h 在sys\_call\_table数组最后加入sys\_foo，并仿照上面给出其他系统调用格式加上extern rettype sys\_foo();
2. 修改include/unistd.h

#define \_\_NR\_foo num

num为接下来使用的系统调用号

1. 修改kernel/system\_call.s

nr\_system\_calls = num

num为在原值加1 即系统调用总数目加1

1. 在kernel中添加 foo.c

若需要支持内核态与用户态数据交互

则包含include/asm/segment.h，其中有put\_fs\_XXX get\_fs\_XXX函数

在foo.c实现系统调用sys\_foo()

1. 修改kernel的Makefile，将foo.c与内核其它代码编译链接到一起
2. 系统调用用户界面要

#define \_\_LIBRARY\_\_

#include <unistd.h>

\_syscallN宏展开系统调用，提供用户态的系统调用接口（参数数目确定具体宏）