实验 1-2 报告

1336101 班　白广通 黄渝光

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？

答： 从 linux-0.11/include/unistd.h 中_syscalln 的宏展开我们可以知道，系统调用最多能够使用 ebx，ecx，edx 三个寄存器传递 3 个参数。

```
#define _syscall1(type,name,atype,a) \
type name(atype a) \
{ \
long __res; \
__asm__ volatile ("int $0x80" \
    : "=a" (__res) \
    : "0" (__NR_##name),"b" ((long)(a))); \
if (__res >= 0) \
    return (type) __res; \
errno = -__res; \
return -1; \
}
```

```
#define _syscall2(type,name,atype,a,btype,b) \
type name(atype a,btype b) \
{ \
long __res; \
__asm__ volatile ("int $0x80" \
    : "=a" (__res) \
    : "0" (__NR_##name),"b" ((long)(a)),"c" ((long)(b))); \
if (__res >= 0) \
    return (type) __res; \
errno = -__res; \
return -1; \
}
```

```
#define _syscall3(type,name,atype,a,btype,b,ctype,c) \
type name(atype a,btype b,ctype c) \
{ \
long __res; \
__asm__ volatile ("int $0x80" \
    : "=a" (__res) \
    : "0" (__NR_##name),"b" ((long)(a)),"c" ((long)(b)),"d" ((long)(c))); \
if (__res>=0) \
    return (type) __res; \
errno=-__res; \
return -1; \
}
```

解决限制的方法：将需要传递的参数依次保存在具有特定结构的区间（如栈）中，并将这个区间的首地址作为一个参数，区间中保存的参数的个数作为另一个参数通过寄存器传递给系统调用函数。然后通过寄存器间接寻址方式便可以访问所有参数。#当然，这么做的话，必须要验证参数的合法性。

2. 用文字简要描述向 Linux 0.11 添加一个系统调用 foo()的步骤。

a) 首先，修改/usr/include/unistd.h，添加#define __NR_foo num，其中 num 为接下来使用的系统调用号。（要直接在 linux-0.11 中修改或修改后放到 linux-0.11 中）

b) 其次，修改 include/linux/sys.h 在 sys_call_table 函数指针数组最后加入 sys_foo，并仿照上面给出其他系统调用格式加上对系统调用函数的声明:

```
extern rettype sys_foo();
```

c) 然后修改 kernel/system_call.s：

```
nr_system_calls = num
```

其中 num 为原值加 1， 即系统调用总数目加 1。

d) 接着在 kernel 中添加 foo.c 。若需要在内核态与用户态之间进行数据的交换，则需要包含 include/asm/segment.h，其中有 put_fs_XXX get_fs_XXX 函数。同时，在 foo.c 实现系统调用 sys_foo()

e) 最后修改 kernel 中的 Makefile，将 foo.c 与内核中的其它代码编译链接到一起

f) 实现系统调用的 API:

```
#define __LIBRARY__
#include <unistd.h>
```

_syscallN //宏展开系统调用，提供用户态下系统调用的接口（N 代表参数数目，用以具体确定宏的名称）。