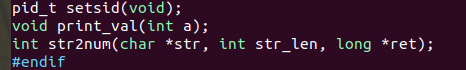
# EXP2.1

## 实验过程

1. **在linux-0.11/hdc/usr/include/unistd.h下定义系统调用号，并声明系统调用函数的形式。**

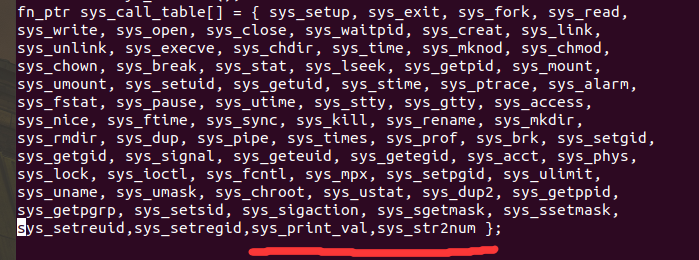
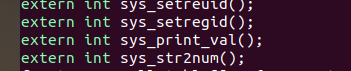
**C:\Users\Lucifer.dark\AppData\Roaming\Tencent\QQ\Temp\3I0_N%AF[ZC_F0J4RG8MLVX.png**



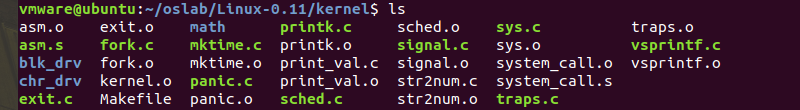
1. **在linux-0.11/kernel/system\_call.s中修改系统调用的个数，以使此系统调用被调用时，可以识别到。**

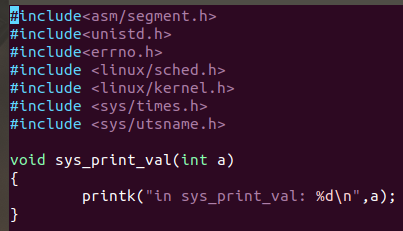
**C:\Users\Lucifer.dark\AppData\Roaming\Tencent\QQ\Temp\4S)1`_4]D[1$QGZ~4M@CZOG.png**

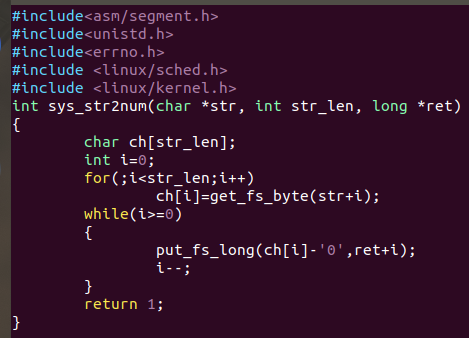
1. **在linux0.11/include/linux/sys.h中添加extern头，再在sys\_call\_table[]中加入系统调用的‘地址’。**

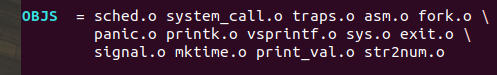


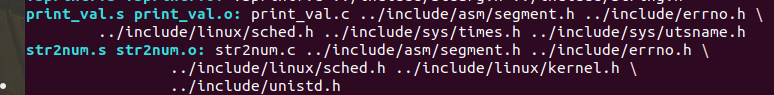
1. **在linux-0.11/kernel中实现该系统调用，并修改Makefile文件。**

****

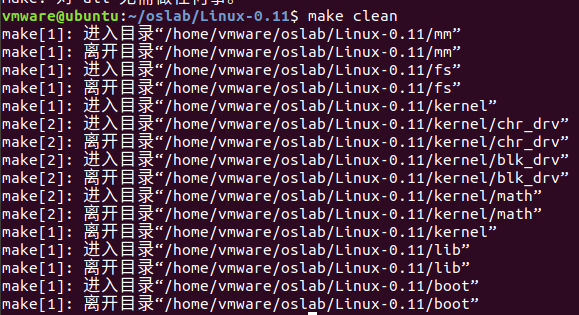


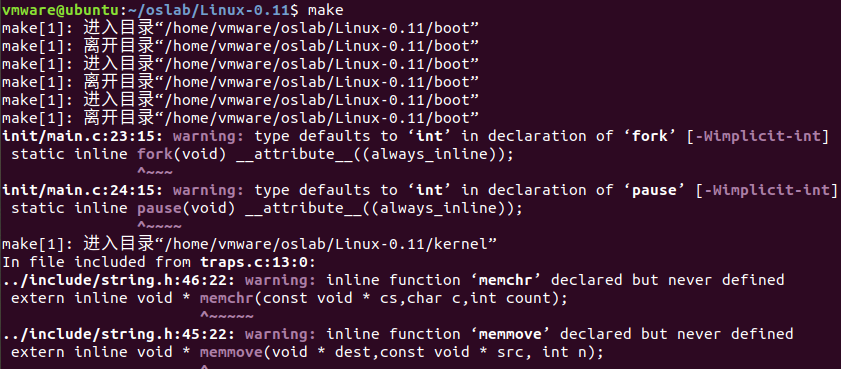


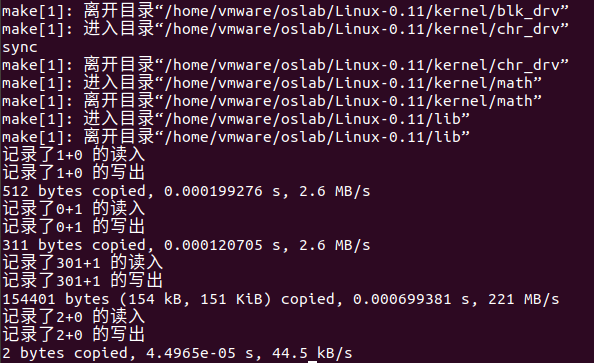




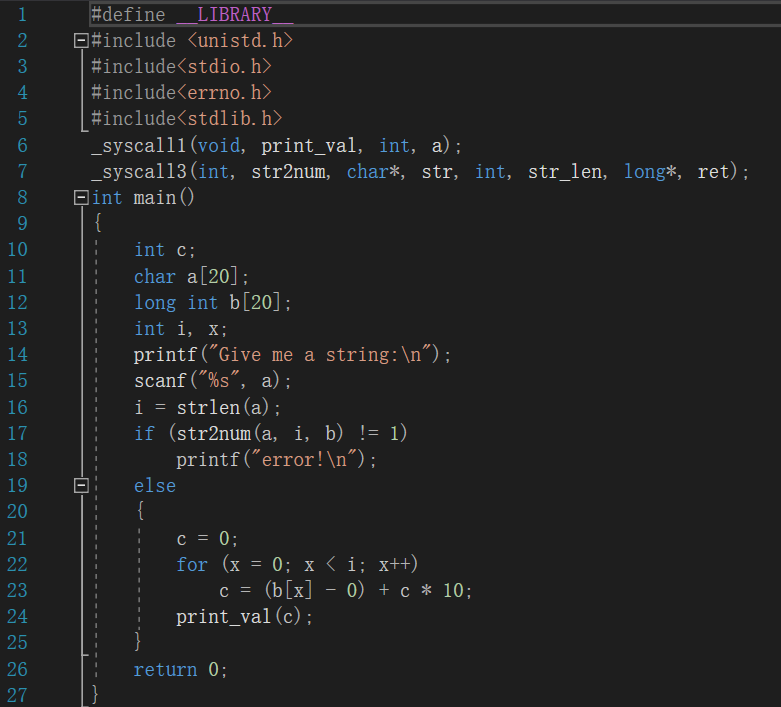
**⑤编译内核**



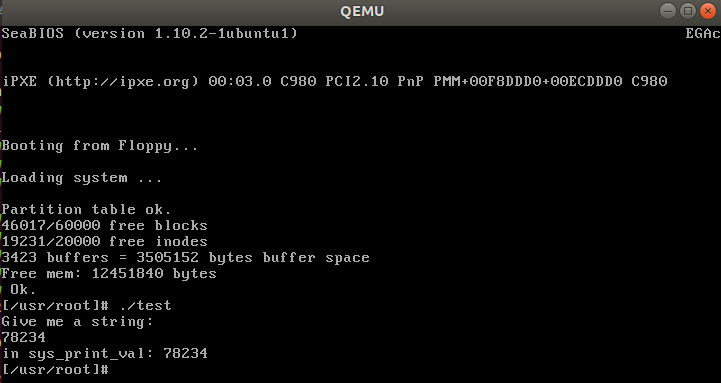




**⑥编写测试程序**



**⑦运行测试程序**



## 相关问题

1. **简要描述如何在Linux 0.11添加一个系统调用。**

**①在linux-0.11/include/unistd.h下定义系统调用号，并声明系统调用函数的形式。**

**②在linux-0.11/kernel/system\_call.s中修改系统调用的个数，以使此系统调用被调用时，可以识别到。**

**③在linux0.11/hdc/usr/include/sys.h中添加extern头，再在sys\_call\_table[]中加入系统调用的‘地址’。**

1. **在linux-0.11/kernel中实现该系统调用，并修改Makefile文件。**
2. **系统是如何通过系统调用号索引到具体的调用函数的?**

**①应用程序调用库函数（API）；**

**②API将系统调用号存入EAX，然后通过中断调用使系统进入内核态；**

**③中断处理函数system\_call把参数入栈(kernel/system\_call.s)**



**④根据索引在sys\_call\_table中查找函数地址(include/linux/sys.h)**

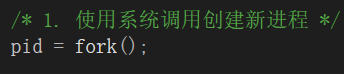
**⑤执行真正的系统调用函数。**

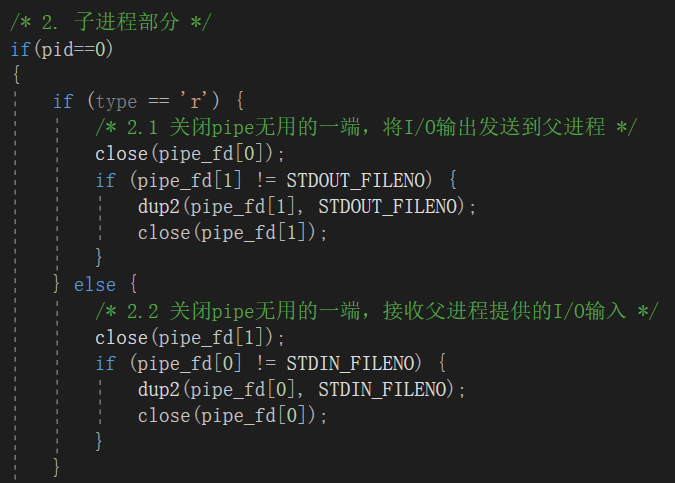
1. **在Linux 0.11中, 系统调用多支持几个参数? 有什么方法可以超过这个限制吗?**

**直接能传递的参数至多有 3 个。在 Linux-0.11 中，程序使用 ebx、ecx、edx 这三个通用寄存器保存参数，可以直接向系统调用服务过程传递至多三个参数 (不包括放在 eax 寄存器中的系统调用号)。如果使用指向用户数据空间的指针，将指针信息通过寄存器传递给系统调用服务，然后系统调用就可以通过该指针访问用户数据空间中预置的更多数据，就可以达到传递更多参数的目的。**

# EXP2.2

**首先添加os\_popen函数的代码**

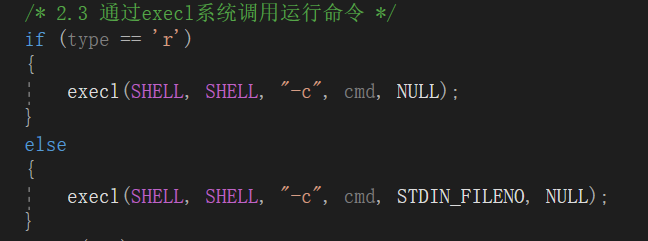
使用fork()函数创建子进程。



子进程其本身无子进程，返回的pid为0

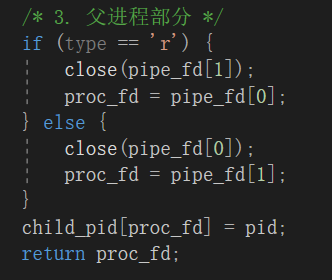
当type为r时，此时应该输出其内容，关闭管道读端pipe\_fd[0]，并将其对应写端文件描述符传给标准输出STDOUT\_FILENO。

当前类型为w时，此时应向其输入内容，关闭管道写端pipe\_fd[1]，并将其对应读端文件描述符传给标准输入STDIN\_FILENO。



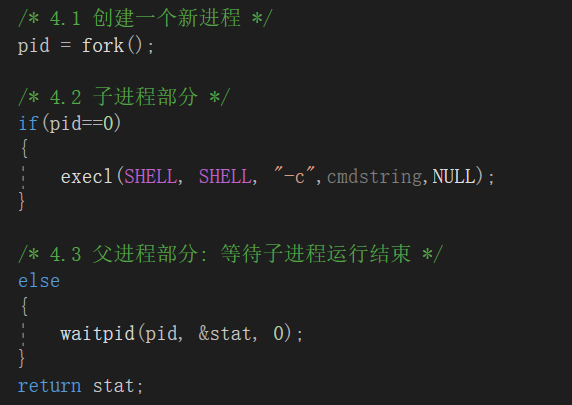
当为r类型时，调用SHELL并执行相应命令。

当为w类型时，调用SHELL执行相应命令，并传入相关输入。

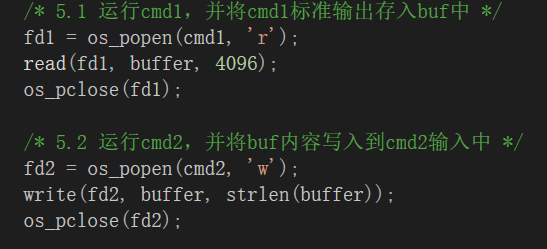


与子进程相反，类型为r时关闭写端，函数返回读端；类型为w时关闭读端，函数返回写端。

**然后添加os\_system函数。**

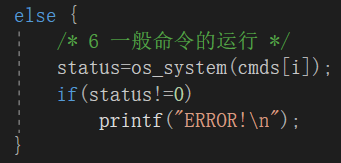


类似的，先创建子进程，然后子进程使用excel调用SHELL，执行相关命令，父进程等待子进程结束，返回相关状态。



实现管道功能，将管道命令的第一个命令的输出利用系统调用read读入buffer,并关闭相应管道；将第二个命令的输入利用系统调用write从buffer写入，并关闭相应管道。

一般的命令，使用os\_system实现，如果子进程运行错误，输出ERROR。



**再Linux-0.11中的实现效果：**

