****

计算机操作系统实验1

向Linux内核增加一个系统调用

姓名： 涂煜洋

学号： 71118119

东南大学计算机科学与工程学院、软件学院

School of Computer Science & Engineering

College of Software Engineering

Southeast University

2020年5月

1. **实验目的及内容**

**1.实验目的：**

通过实验，熟悉 Linux 操作系统的使用，掌握构建与启动 Linux 内核的方法； 掌握用户程序如何利用系统调用与操作系统内核实现通信的方法，加深对系统调 用机制的理解；进一步掌握如何向操作系统内核增加新的系统调用的方法，以扩展操作系统的功能。

**2.实验内容：**

1. Linux 环境下的 C 或C++编译和调试工具的使用。

2. 向 Linux 内核增加新的系统调用，系统调用名称和功能自行定义，但必须 实现如下输出功能：“My Student No. is ×××，and My Name is ×××”。

3. Linux 新内核的编译、安装和配置。

4. 编写应用程序以测试新的系统调用并输出测试结果。

**二、实验的基本知识与步骤**

**1.Linux系统调用：**

**1.1** **系统调用表与系统调用例程**：

Linux中的系统调用通过**系统调用号**和**系统调用表**来定位相应系统调用例程的入口，其中系统调用表定义在文件中。

每个系统调用都对应于一个内核服务例程来实现其功能，而例程代码位于 文件中。而服务例程的原型则声明于 文件中。

**1.2 Linux系统调用设计：**

Linux中的系统调用函数是通过在文件中的宏定义的，接下来展示Linux相关宏定义的意义：

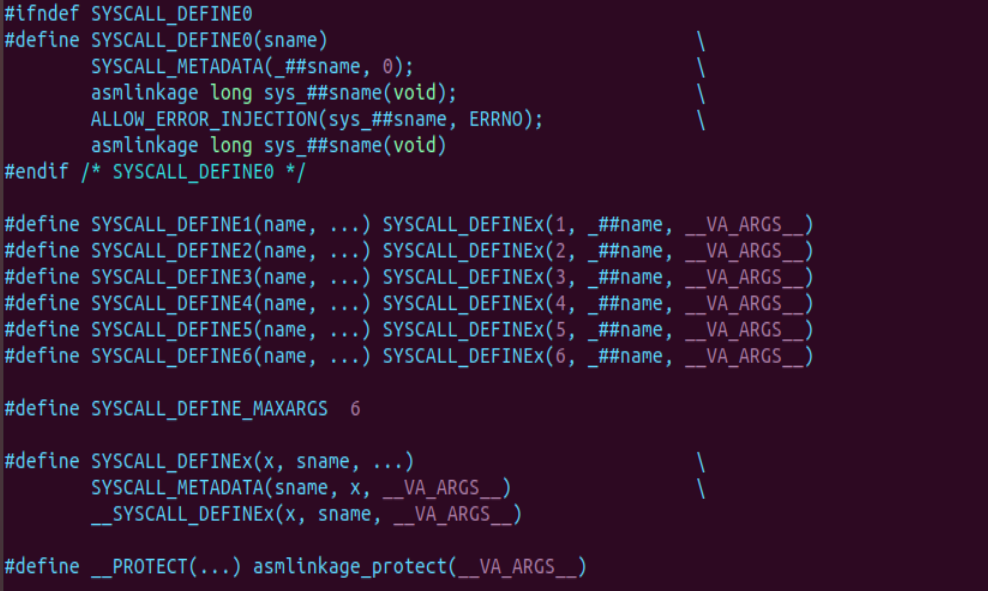


图1.2-1 SYSCALL\_DEFINE宏定义

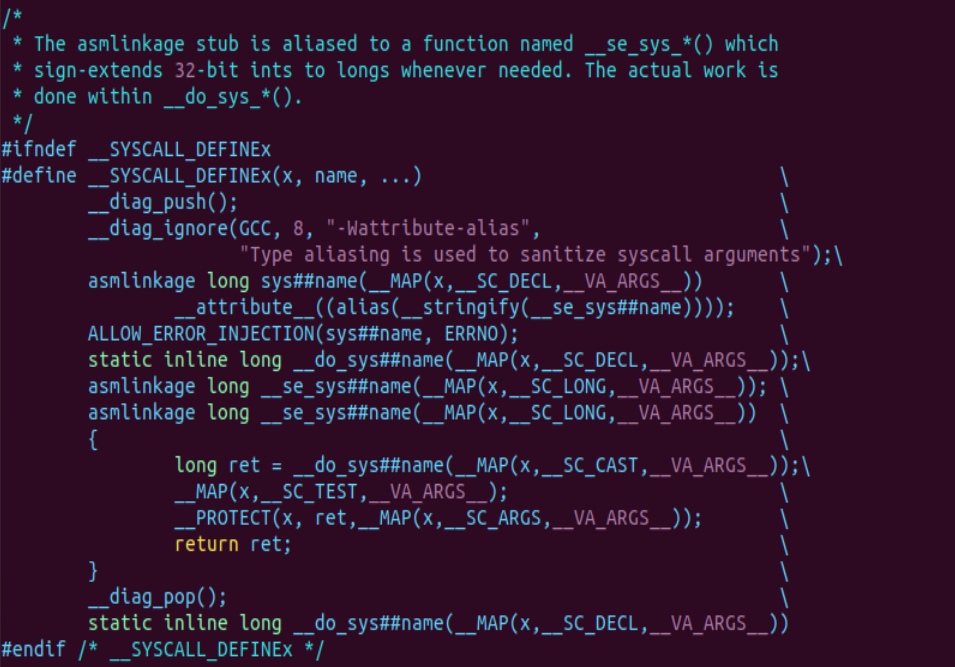


图1.2-2 \_\_SYSCALL\_DEFINEx宏定义

经过查阅资料得到通过如此复杂的宏来定义系统调用函数是出于历史原因，在Linux 2.6.28及以前版本内核中，IBM/S390、PowerPC、Sparc64以及MIPS 64位平台的ABI要求在系统调用时，用户空间程序将系统调用中32位的参数存放在64位的寄存器中要做到正确的符号扩展，但是用户空间程序却不能保证做到这点，这样就会可以通过向有漏洞的系统调用传送特制参数便可能导致系统崩溃或获得权限提升，这个漏洞被命名为 CVE-2009-0029。为解决这个漏洞，Linux内核先将所有参数当成long类型，之后再通过强制类型转换成原类型来避免可能的符号扩展错误，并通过宏定义的方式作用到所有的系统调用函数中。

而宏中实际上指明所需要的参数，而的实际意义为系统调用函数的名字。在之后的实验过程中，由于我选择向系统调用函数中传入一个返回参数，因此会选择宏。

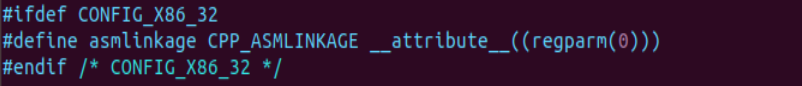
同时值得一提的是，在声明系统调用原型时，需要添加一个限定词，他被定义为如下宏（位于）：

图1.2-3 asmlinkage宏定义

这个宏表示限定函数不通过寄存器，而通过栈来传递参数。

**2.实验步骤：**

（注：实验步骤将均在shell中实现，且当前工作目录为 ）

**2.1 获取Linux源码：**

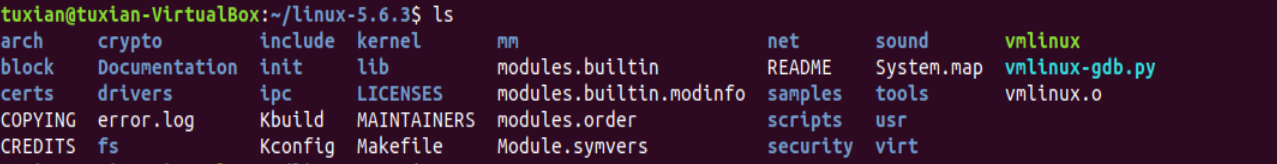
可以通过Linux官网的下载地址（<https://www.kernel.org/>）获取Linux源码。（我实验时所下载的版本为5.6.3）下载完毕后可以通过命令解压压缩包获取完成的源程序。

图2.1-1 linux源码目录（实验完成后）

**2.2 修改系统调用表：**

Linux内核的系统调用表位于文件中，可以通过命令:

访问这个文件。Linux系统调用表的源码通过：

的形式组织，其中

<number>为系统调用号，

<abi>为应用二进制接口（分为三种：64，x32，common），

<name>为系统调用名，

<entry point>为服务例程入口。

此处参考代码中的注释建议，将新增的系统调用号设置为335（位于最后一个系统调用号334之后），并按照上述格式添加：

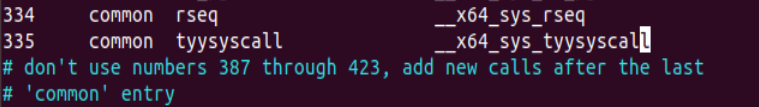


图2.2-1 新增的系统调用表项

可以看到系统调用名被设置为。

**2.3 添加系统调用声明：**

Linux系统调用函数声明于文件中，可以通过命令：

来访问这个文件。在文件的末尾添加上自己的系统调用声明即可：

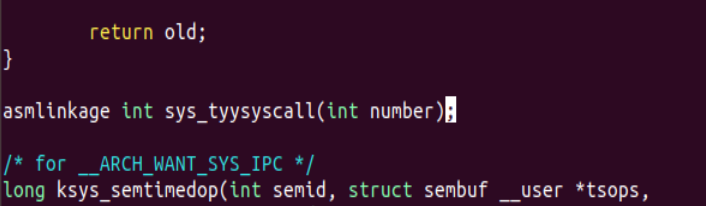
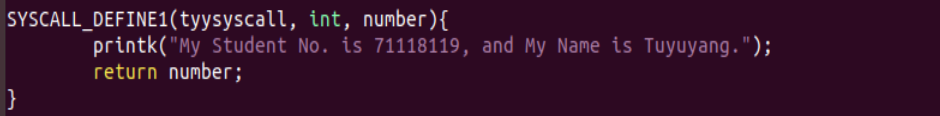


图2.3-1 新增系统调用声明

**2.4添加系统调用服务例程：**

Linux系统调用函数定义于文件中。可通过命令：

访问这个文件，类似的，只需要在文件的末尾添加上自己新增的系统调用函数即可。



注意，由于我此处选择传递一个返回参数，故选择宏，并且由于宏的特性，函数定义的格式与以往有所不同，表现在对所有类型与参数名之间也需要逗号隔开，其中第一个参数为系统调用函数的函数名。

**2.5 编译与安装Linux内核：**

**2.5.1 编译：**

由于之前使用过本虚拟机实现某些实验，故编译所需的相关依赖配置都已完整，这里仅给出编译的相关命令：

#这里选择默认配置，即之间选择exit退出即可。

#可选，用于获得CPU的相关信息，可以选择最大核数进行编译，可以加快速度。



图2.5-1 CPU相关信息

#参数可选，表示选择使用n核进行编译

**2.5.2安装：**

#用于进行模块的安装。

#安装内核

当内核安装完毕后即可重启系统，重启完成后机器的操作系统应该就是刚刚安装完成的操作系统了。可以通过命令：

来查询操作系统的版本以判断是否安装成功：



图2.5-2 操作系统版本查询

**三、测试实验结果**

可以通过创建测试代码（任意目录下）来测试系统调用的结果，这里选择一个简单的c文件来测试系统调用：

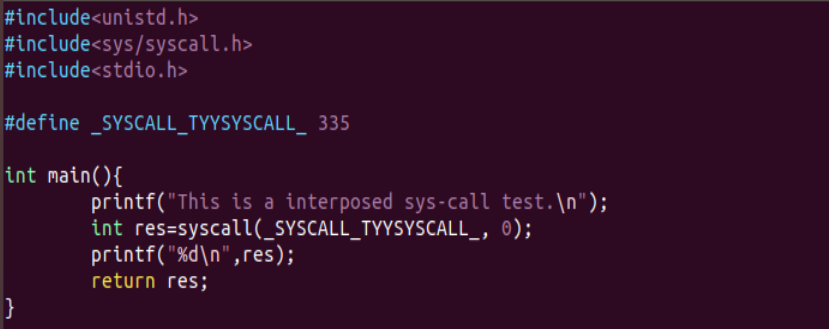


图3-1 系统调用测试代码

完成代码之后使用gcc进行编译，并执行测试结果：

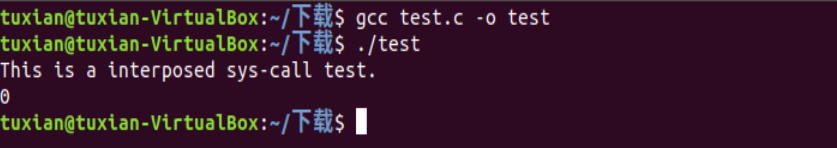


图3-2 代码编译与执行

使用命令查看日志可以看到系统调用函数输出的文字：

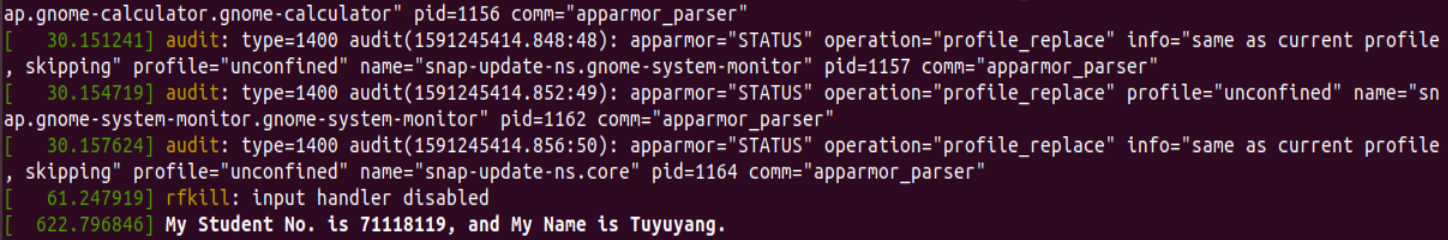


图3-3 系统调用输出结果

**四、实验感想**