# 实验报告：添加一个新的系统调用来打印Hello

## 实验目的

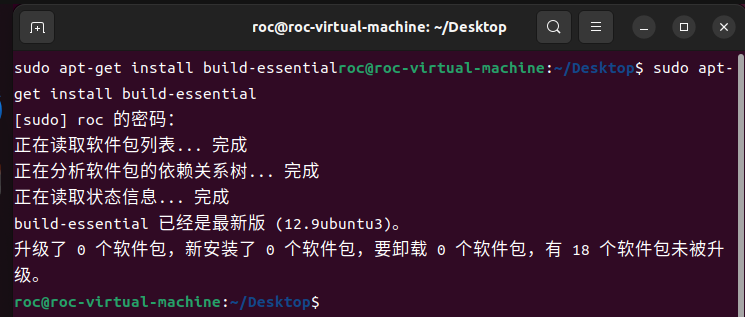
本实验旨在向Linux内核添加一个自定义的系统调用，该系统调用将用于打印"Hello"消息。这个实验还会涉及编写用户态测试程序来验证新系统调用的功能。

## 实验步骤

步骤 1: 安装GCC和必要的构建工具

为了编译内核和用户态测试程序，首先需要安装GCC和其他构建工具。使用以下命令来安装更新：

sudo apt-get install build-essential



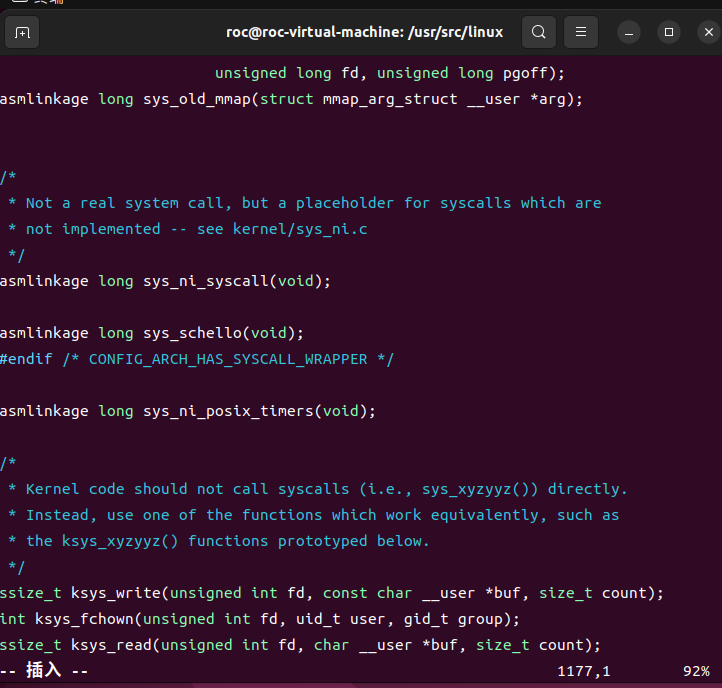
步骤 2: 创建自定义内核配置

首先，创建一个自定义内核配置。在本实验中，我们使用版本为6.5.x的Linux内核。之前的实验中已完成。

步骤 3: 修改系统调用头文件

在内核源代码中，打开文件 include/linux/syscalls.h。在 #endif /\* CONFIG\_ARCH\_HAS\_SYSCALL\_WRAPPER \*/ 之前，添加以下行：

asmlinkage long sys\_schello(void);



步骤 4: 添加新系统调用函数

打开内核源代码文件 kernel/sys.c。在函数 SYSCALL\_DEFINE0(gettid) 之后，添加以下代码：

SYSCALL\_DEFINE0(schello)

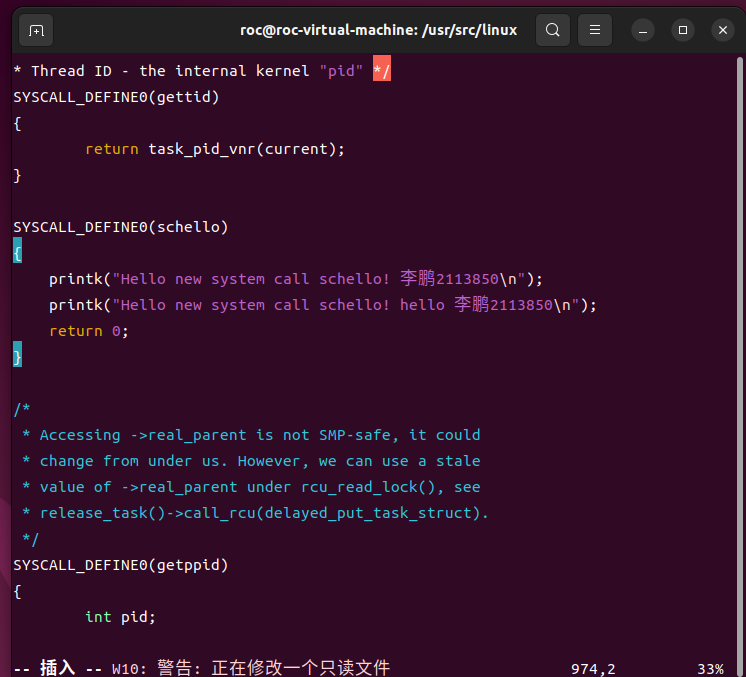
{

printk("Hello new system call schello! Your ID\n');

printk('Hello new system call schello! hello 学号\n');

return 0;

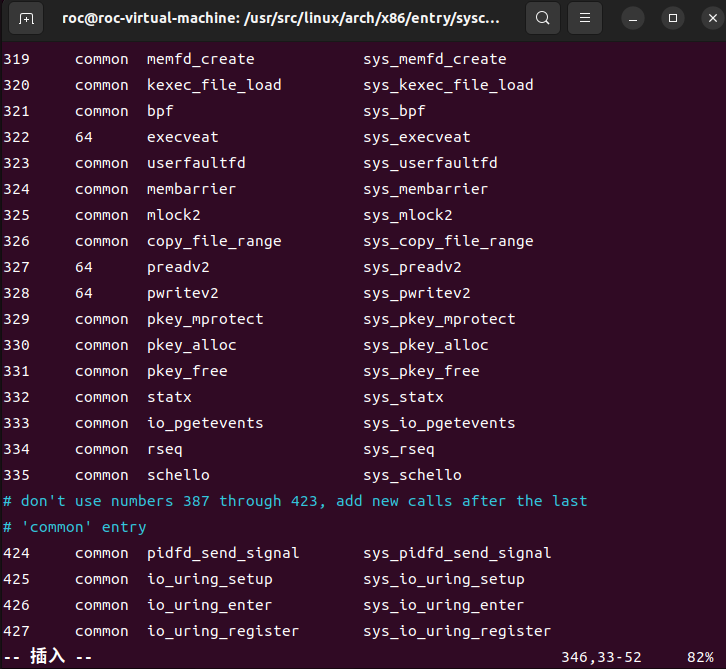
}



步骤 5: 更新系统调用表

对于64位操作系统，在内核源代码的文件 arch/x86/entry/syscalls/syscall\_64.tbl 中，找到行 334 common memfd\_secret sys\_memfd\_secret，在其下添加以下行：

335 common schello sys\_schello



步骤 6: 编译和安装内核

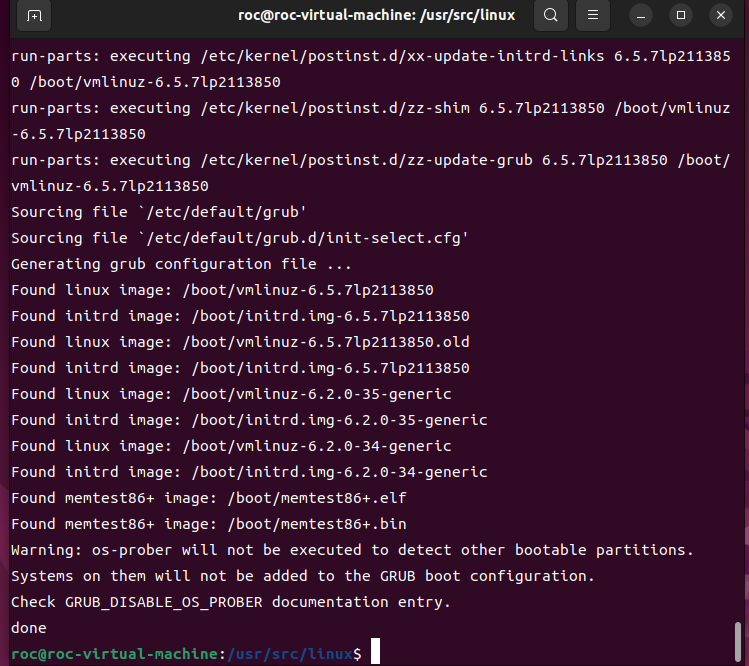
执行以下命令来清理旧的编译文件，编译内核，安装内核模块和内核本身：

make clean

make -j5

sudo make modules\_install

sudo make install



步骤 7: 重新启动系统

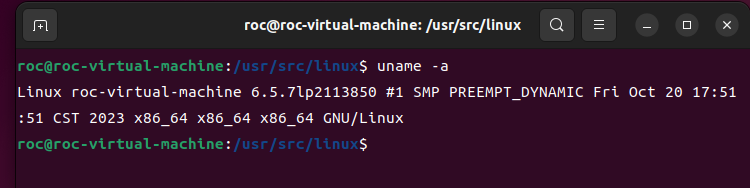
执行以下命令来重新启动系统，以应用新内核：

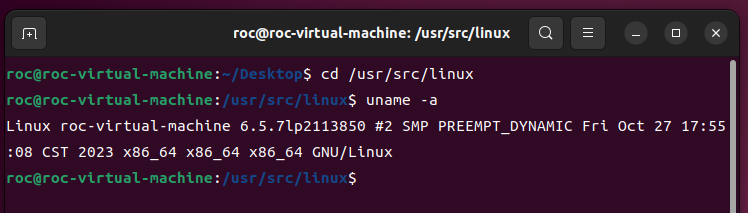
Reboot

步骤 8: 验证新系统调用

在系统重新启动后，使用以下命令来验证新系统调用是否成功添加到内核中：

uname -a





步骤 9: 编写用户态测试程序

创建一个用户态测试程序的C文件（例如 testschello.c），内容如下：

#include <unistd.h>

#include <sys/syscall.h>

#include <sys/types.h>

#include <stdio.h>

#define \_\_NR\_schello 335

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

{

syscall(\_\_NR\_schello);

printf("ok! run the cmd in terminal: sudo dmesg | grep schello\n");

return 0;

}

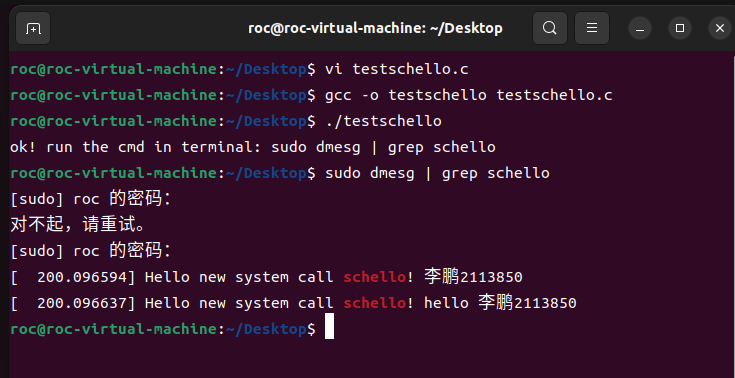
步骤 10: 编译和运行用户态测试程序

执行以下命令来编译用户态测试程序和运行它：

gcc -o testschello testschello.c

./testschello

sudo dmesg | grep schello



## 实验总结：

通过本实验，成功地向Linux内核中添加了一个新的系统调用，该系统调用用于打印"Hello"消息。还编写并运行了一个用户态测试程序来验证新系统调用的功能。这个实验展示了如何自定义内核并添加自定义系统调用。这对于深入理解Linux内核和进行系统编程非常有用。