# 《操作系统》课第四次实验报告

|  |  |
| --- | --- |
| 学院: | 软件学院 |
| 姓名: | 郁万祥 |
| 学号: | 2013852 |
| 邮箱: | yuwanxiang0114@163.com |
| 时间: | 2022.10.7 |

## 开篇感言

经过了几节实验课的摧残，终于渐渐对ubuntu以及linux内核的本质熟络起来。1、对于ubuntu中的命令行操作其实有点类似于函数的调用，通过内置的一些库直接进行某些操作。2、linux内核更像是用C语言编写的库，定义了一些可以控制、调配系统硬件设施的功能，并且提供了相应的接口，使得用户可以直接使用。理解了这些，回过头来来看OS的整体原理脉络，是要比刚开始清晰许多了的。

## 实验题目

添加系统调用 Add a New System Call

## 实验目标

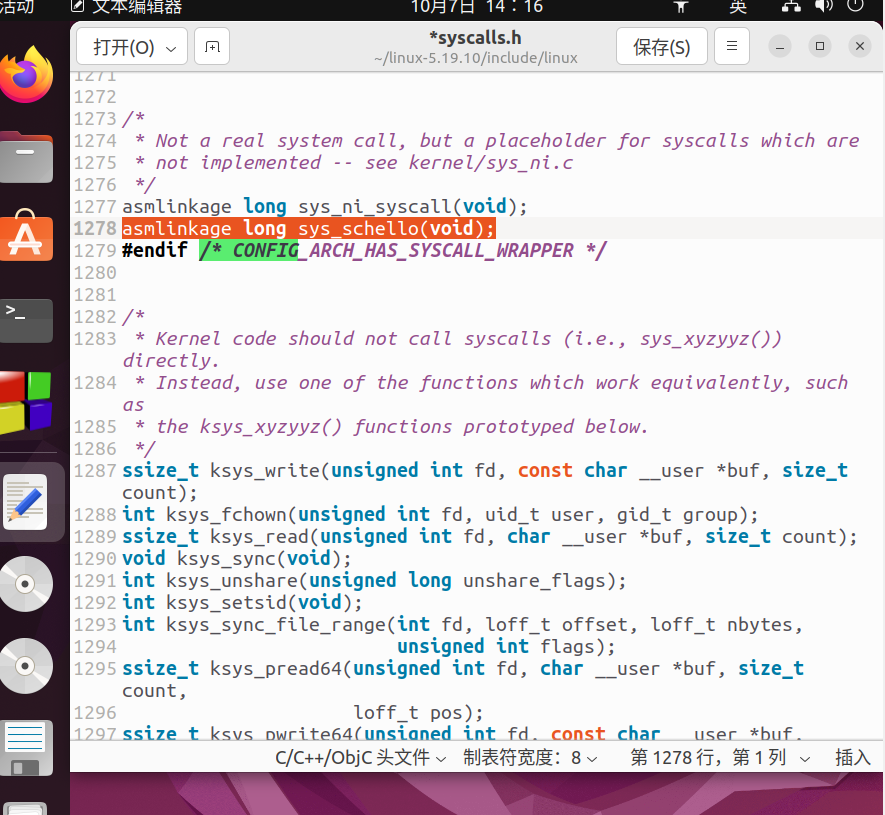
添加系统调用并且使用syscall进行调用

## 原理方法

**一般情况下，用户进程是不能访问内核的。它既不能访问内核所在的内存空间，也不能调用内核中的函数。系统调用是一个例外。其原理是(1)进程先用适当的值填充寄存器，(2)然后调用一个特殊的指令，(3)这个指令会让用户程序跳转到一个事先定义好的内核中的一个位置。（4）进程可以跳转到的固定的内核位置。这个过程检查系统调用号，这个号码告诉内核进程请求哪种服务。然后，它查看系统调用表(sys\_call\_table)找到所调用的内核函数入口地址。接着，就调用函数，等返回后，做一些系统检查，最后返回到进程。**

## 具体步骤

1. 在syscall.h头文件当中添加asmlinkage long sys\_schello(void);



1. 在kernal/sys.c中添加自己的服务函数

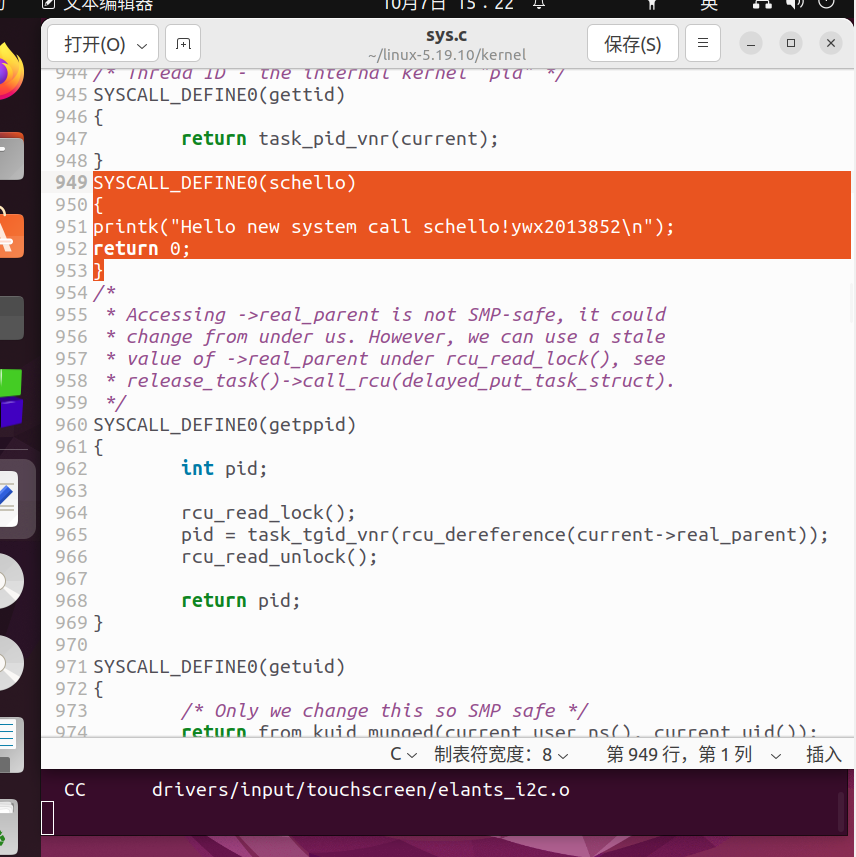
SYSCALL\_DEFINE0(schello)

{

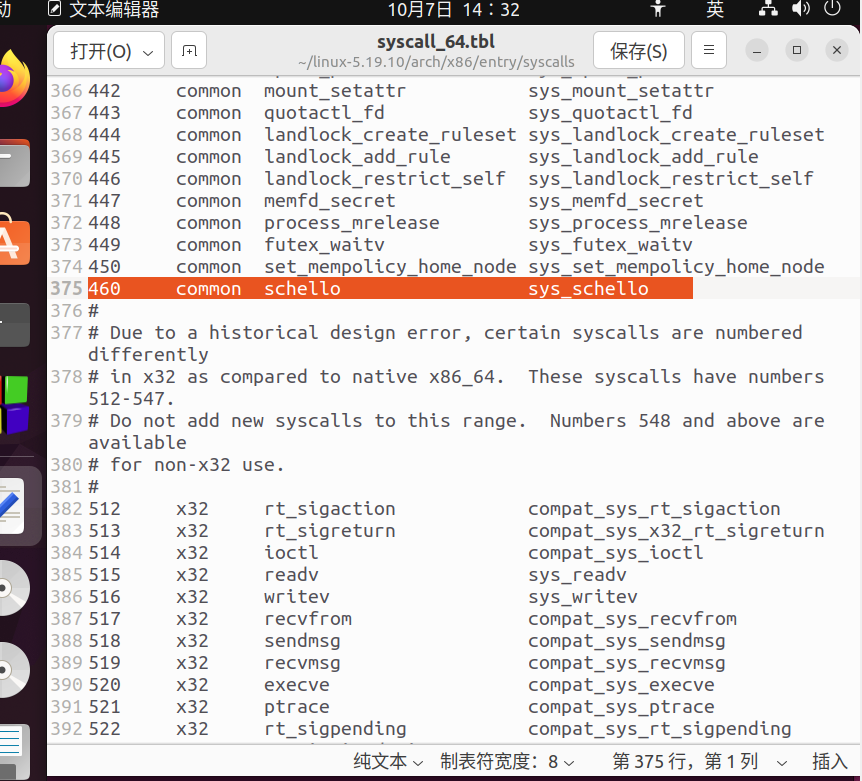
printk("Hello new system call schello!ywx2013852\n");

return 0;

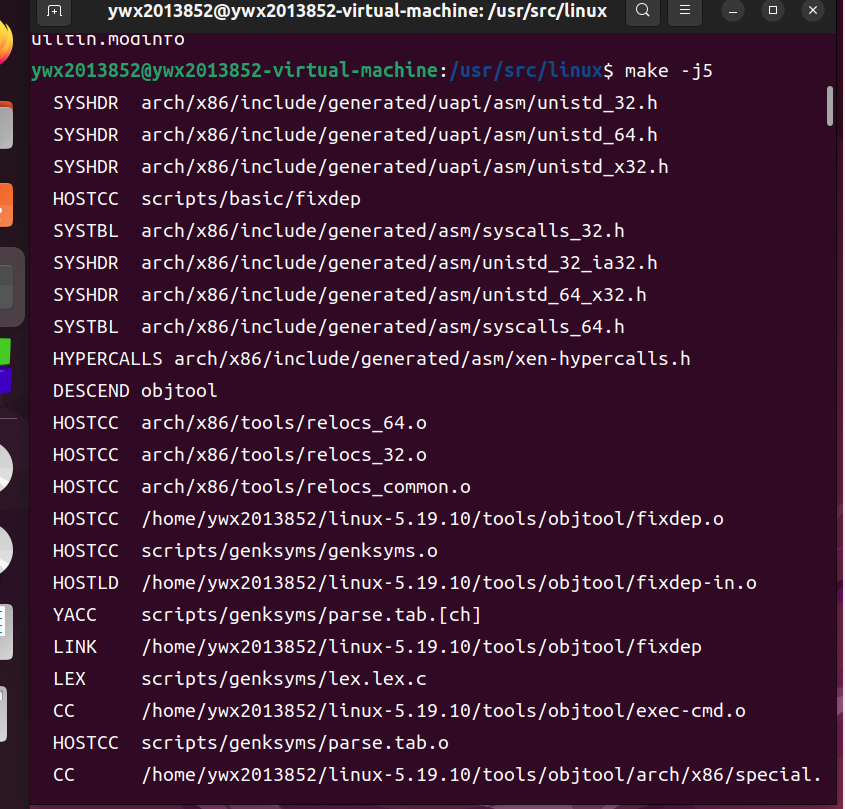
}



1. 在arch/x86/entry/syscalls/syscall\_64.tbl中添加系统调用号，添加系统调用号和系统调用函数的对应关系。

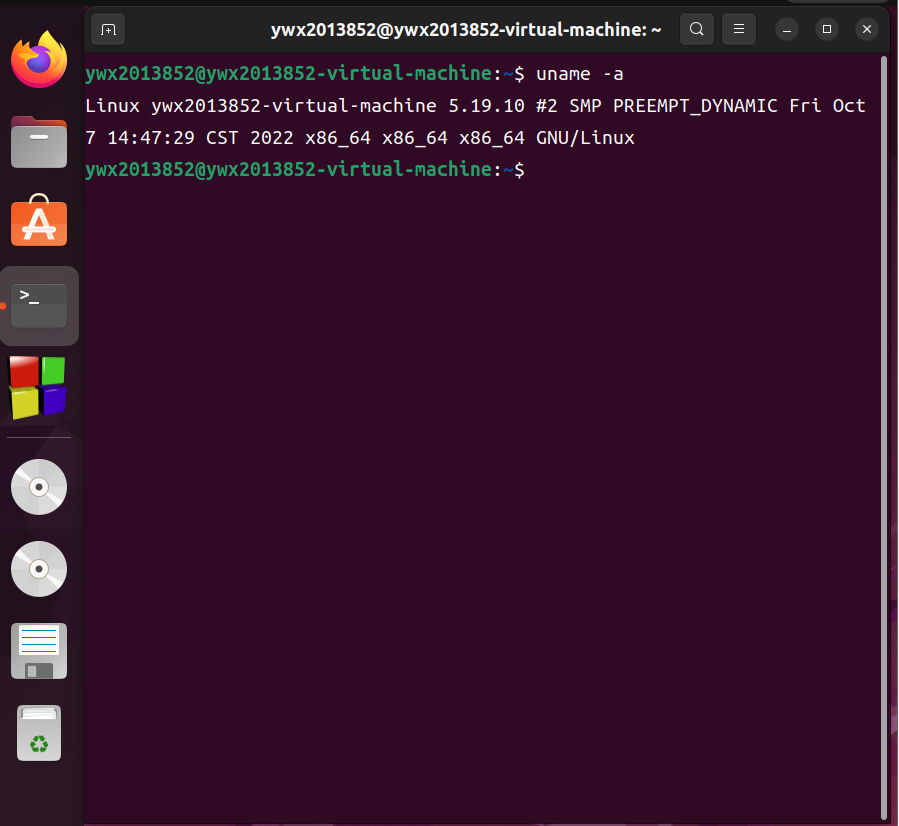


1. 重新编译内核

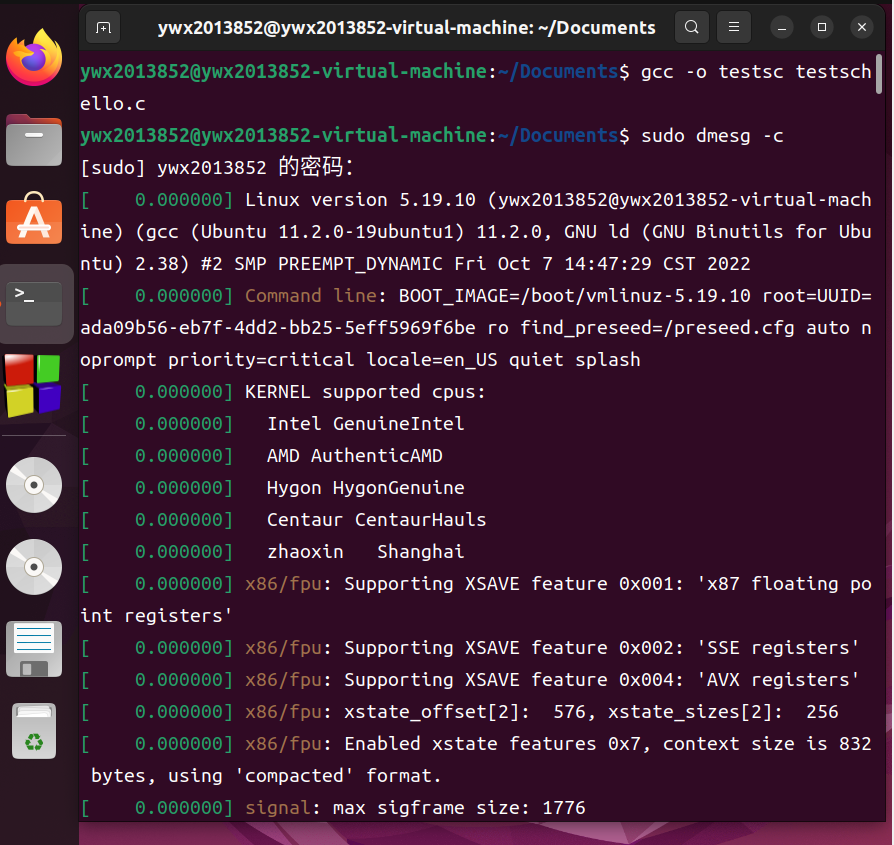


1. 重新启动后，检查内核版本，编写demo进行系统调用，进行测试：

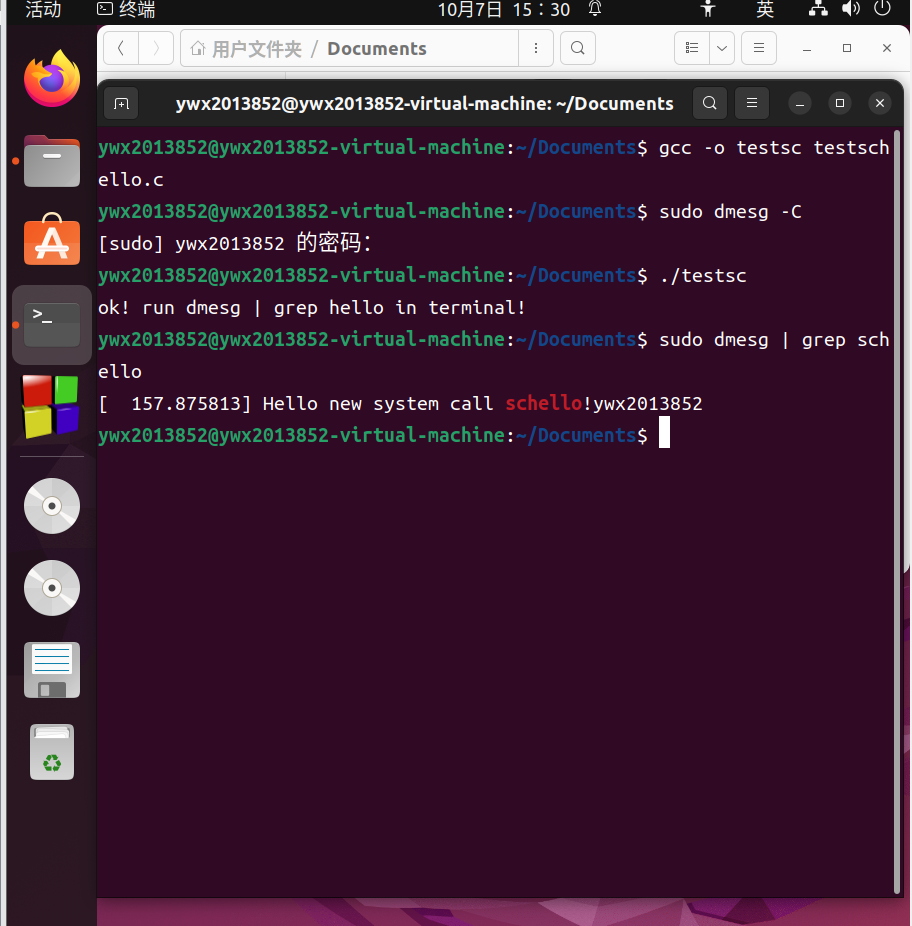
内核版本检查：



系统调用c文件编译：



调用结果：



## 总结心得

其实操作系统说跟到底还是系统软件程序，所谓linux内核，就是一些用C语言编写的能够调用内存等硬件设备的一些程序，所以，我们通过修改内核中的用C语言编写的程序，就可以增加或者删除（最好不要）一些我们希望能够达到直接访问内核的操作，其实这就是所谓的系统调用，那么系统调用有什么样的好处呢？系统调用大的好处：提供了统一的接口，比如读取数据，API就不必理会数据存储的物理介质。保护了OS的稳定，因为系统调用、异常和中断是外界进入内核的仅有通道，这就保证了不同进程对内核空间的操作是可知并可控的，这为OS多任务调度和虚拟内存实现提供了基础。

## 参考资料

源代码

testschello.c:

#include<unistd.h>

#include<sys/syscall.h>

#include<sys/types.h>

#include<stdio.h>

#define \_NR\_schello 460

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

{

syscall(\_NR\_schello);

printf("ok! run dmesg | grep hello in terminal!\n");

return 0;

}