实验六 系统调用

一、实验目的

1、理解、掌握Linux系统调用的实现框架、用户界面、参数传递、进入/返回过程

二、实验原理

1、系统调用的本质是调用内核函数，以内核态运行程序

三、实验步骤

在系统调用中添加一个不用传递参数的系统调用，实现一个简单的系统调用的添加（mysyscall）。执行这个系统调用，在屏幕上显示输出“My Student No. is XXXXX，and My Name is XXXXX\*\*\* !”。

一般系统没有安装源代码，需要kernel-devel安装

[root@MiWiFi-R1D-srv yum.repos.d]# yum install kernel-devel

（1）源代码的官网：http://vault.centos.org进入官网后，依次是进入 7.4/，进入os/，进入Source/，进入SPackages/，找到 kernel-3.10.0-1127.el7.src).rpm

**运行命令rpm -ivh kernel-3.10.0-1127.el7.src (1).rpm。此时，对应的内核源文件并没有展开到相应的目录下面，而是将这个rpm包展开到/root/rpmbuild/SOURCE目录下面。**

**解压/root/rpmbuild/SOURCE对应目录下面的linux-2.6.32-279.el6.tar.bz2，即可得到内核头文件**

**内核文件一般存储路径/usr/src/kernels/**

**或者下载：**

**[root@MiWiFi-R1D-srvker]#**

**wget https://cdn.kernel.org/pub/linux/kernel/v4.x/linux-4.14.225.tar.xz**

（2）进入源代码，添加系统调用号（vim syscall\_64.tbl）（内核所在目录）

arch/x86/entry/syscalls/syscall\_64.tbl



添加两个没有使用(unuse)的系统调用号，其他格式可以参考文件中的其他系统调用

（3）vim编辑include/linux/syscalls.h文件，添加原型声明。

syscalls.h里在最后一行 #endif前输入声明并保存

asmlinkage long sys\_mycall(void);

（4）vim编辑kernel/sys.c文件，实现具体的系统调用

**函数内容：**

asmlinkage int sys\_mycall(int number)

{

 printk("My Student No. is XXXXX，and My Name is XXXXX\*\*\* !");

 return number;

}

（5）编译内核，参照系统内核升级实验五

（6）编写调用的程序, gcc-o mytest mytest.c(编译C程序)之后 ./mytest 。

mytest.c 内容如下

#include <stdio.h>

#include <stdlib.h>

int main() {

syscall(549,1);

return 0;}

（7）查看系统条用dmesg -c