

《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

0. 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。

1. 实验题目

添加系统调用 Add a New System Call



2. 实验目标

添加系统调用并且使用 syscall 进行调用

3. 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。

4. 具体步骤

1、在 syscall.h 头文件当中添加 asmlinkage long sys_schello(void);

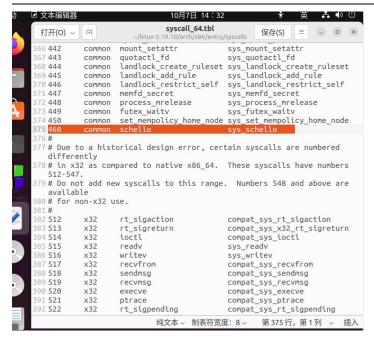


2、在 kernal/sys.c 中添加自己的服务函数

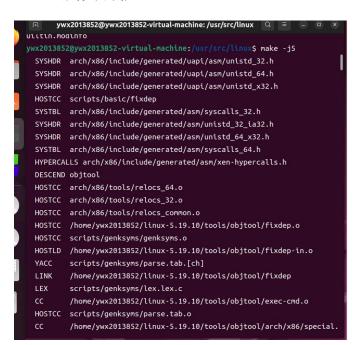
```
SYSCALL_DEFINEO(schello)
printk("Hello new system call schello!ywx2013852\n");
return 0;
                                   10月7日 15・22
     打开(O) ~ 用
                                                       保存(S)
                                                               Ξ
                                                                   _ 0
     944/* Inread ID - the internal Kernel pla
     945 SYSCALL_DEFINEO(gettid)
    946 {
    947
                return task_pid_vnr(current);
     948]
     949 SYSCALL_DEFINEO(schello)
     951 printk("Hello new system call schello!ywx2013852\n");
        return 0;
     955 * Accessing ->real_parent is not SMP-safe, it could
     956 * change from under us. However, we can use a stale
     957 * value of ->real_parent under rcu_read_lock(), see
    958 * release_task()->call_rcu(delayed_put_task_struct).
    959 */
    960 SYSCALL_DEFINEO(getppid)
    961 {
                int pid;
    962
    963
    964
                rcu_read_lock();
                pid = task_tgid_vnr(rcu_dereference(current->real_parent));
rcu_read_unlock();
     965
     966
                return pid;
     969 }
    971 SYSCALL_DEFINEO(getuid)
    972 {
                /* Only we change this so SMP safe */
    973
    974
                return from kuid munaed(current user ns(), current uid()):
                                   C ~ 制表符宽度: 8 ~
                                                       第949行,第1列 ~ 插入
     CC
              drivers/input/touchscreen/elants_i2c.o
```

3、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。





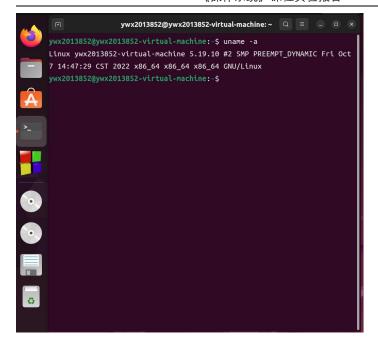
4、重新编译内核



5、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:



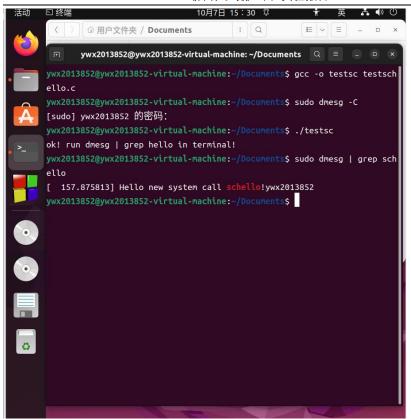


系统调用 c 文件编译:

```
ywx2013852@ywx2013852-virtual-machine: ~/Documents 🔍 😑 😑
ywx2013852@ywx2013852-virtual-machine:-/Documents$ gcc -o testsc testsch
ello.c
ywx2013852@ywx2013852-virtual-machine:-/Documents$ sudo dmesg -c
[sudo] ywx2013852 的密码:
    0.000000] Linux version 5.19.10 (ywx2013852@ywx2013852-virtual-mach
ine) (gcc (Ubuntu 11.2.0-19ubuntu1) 11.2.0, GNU ld (GNU Binutils for Ubu
ntu) 2.38) #2 SMP PREEMPT_DYNAMIC Fri Oct 7 14:47:29 CST 2022
    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-5.19.10 root=UUID=
ada09b56-eb7f-4dd2-bb25-5eff5969f6be ro find_preseed=/preseed.cfg auto n
oprompt priority=critical locale=en_US quiet splash
    0.000000] KERNEL supported cpus:
    0.000000] Intel GenuineIntel
0.000000] AMD AuthenticAMD
    0.000000] Hygon HygonGenuine
    0.000000] Centaur CentaurHauls
0.000000] zhaoxin Shanghai
    0.000000] x86/fpu: Supporting XSAVE feature 0x001: 'x87 floating po
int registers'
    0.000000] x86/fpu: Supporting XSAVE feature 0x002: 'SSE registers'
     0.000000] x86/fpu: Supporting XSAVE feature 0x004: 'AVX registers'
    0.0000000] x86/fpu: xstate_offset[2]: 576, xstate_sizes[2]: 256
    0.000000] x86/fpu: Enabled xstate features 0x7, context size is 832
bytes, using 'compacted' format.
[ 0.000000] signal: max sigframe size: 1776
```

调用结果:

《操作系统》课程实验报告



5. 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。



6. 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;
}
```