一．

在kernel/sys.c添加如下代码

SYSCALL\_DEFINE0(schello)

{

**struct** task\_struct \*p;

printk("Hello new system call schello!  2112966陈高楠 \n");

printk("Hello new system call schello!  2112966陈高楠 \n");

printk("%-20s %-6s %-6s %-6s\n","Name","Pid","PPid","Stat");

**for** (p = &init\_task; (p = next\_task(p)) != &init\_task;)

printk("%-20s  %-6d  %-6d  %-6ld\n",p->comm,p->pid,p->parent->pid,p->\_\_state);

**return** 0;

}

二．

接着开始编译

cd /usr/src/linux

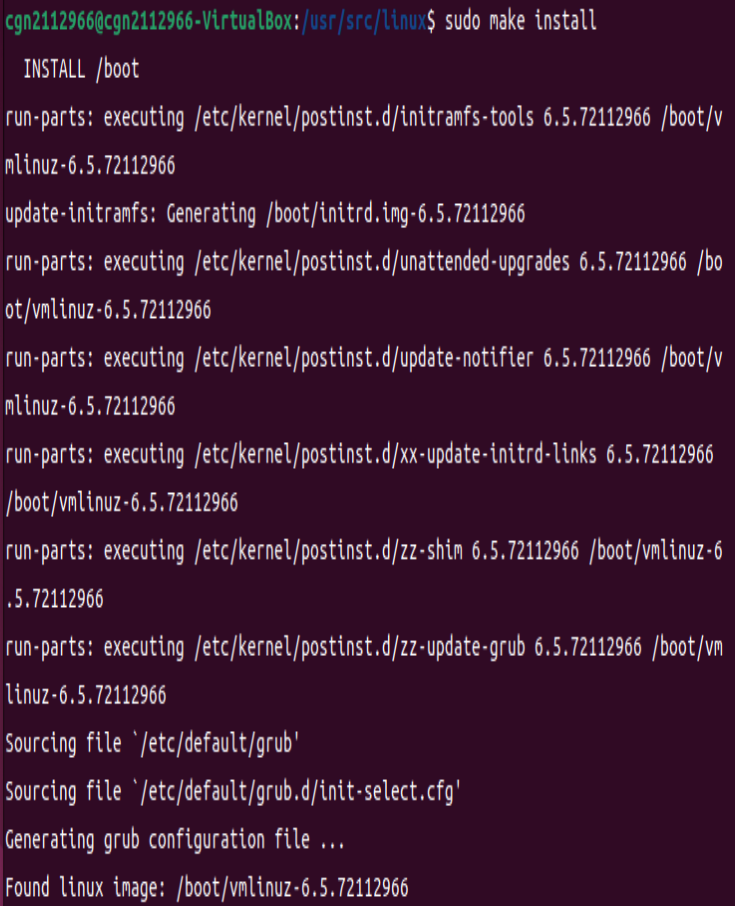
make clean

make -j4

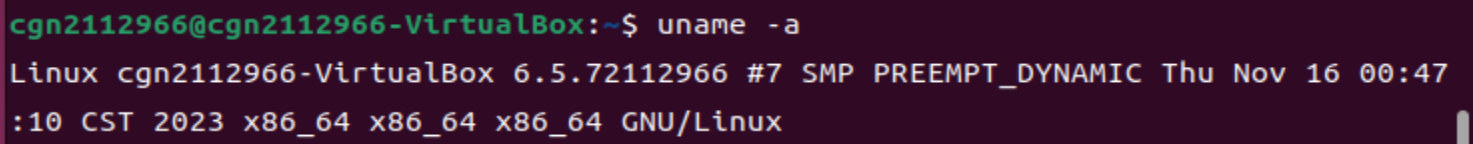
编译完后运行

sudo make modules\_install

sudo make install



然后重启查看



三．

接着编写用户态测试程序testschello.c

#include <unistd.h>

#include <sys/syscall.h>

#include <sys/types.h>

#include <stdio.h>

#define \_\_NR\_schello 335

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

{

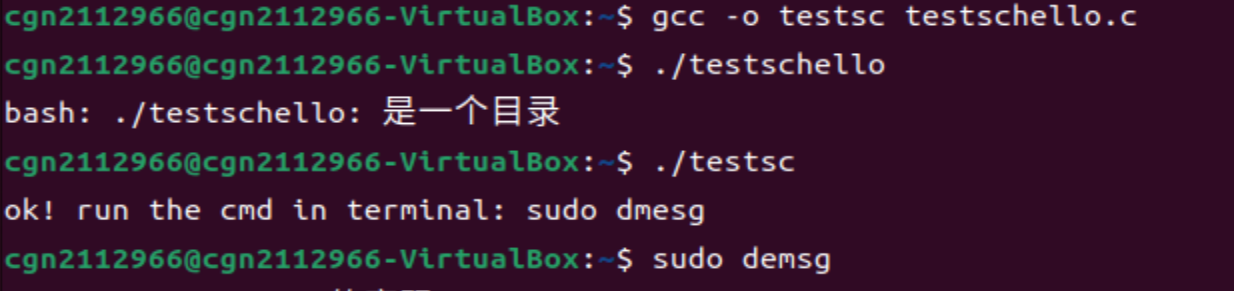
syscall(\_\_NR\_schello);

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

**return** 0;

}

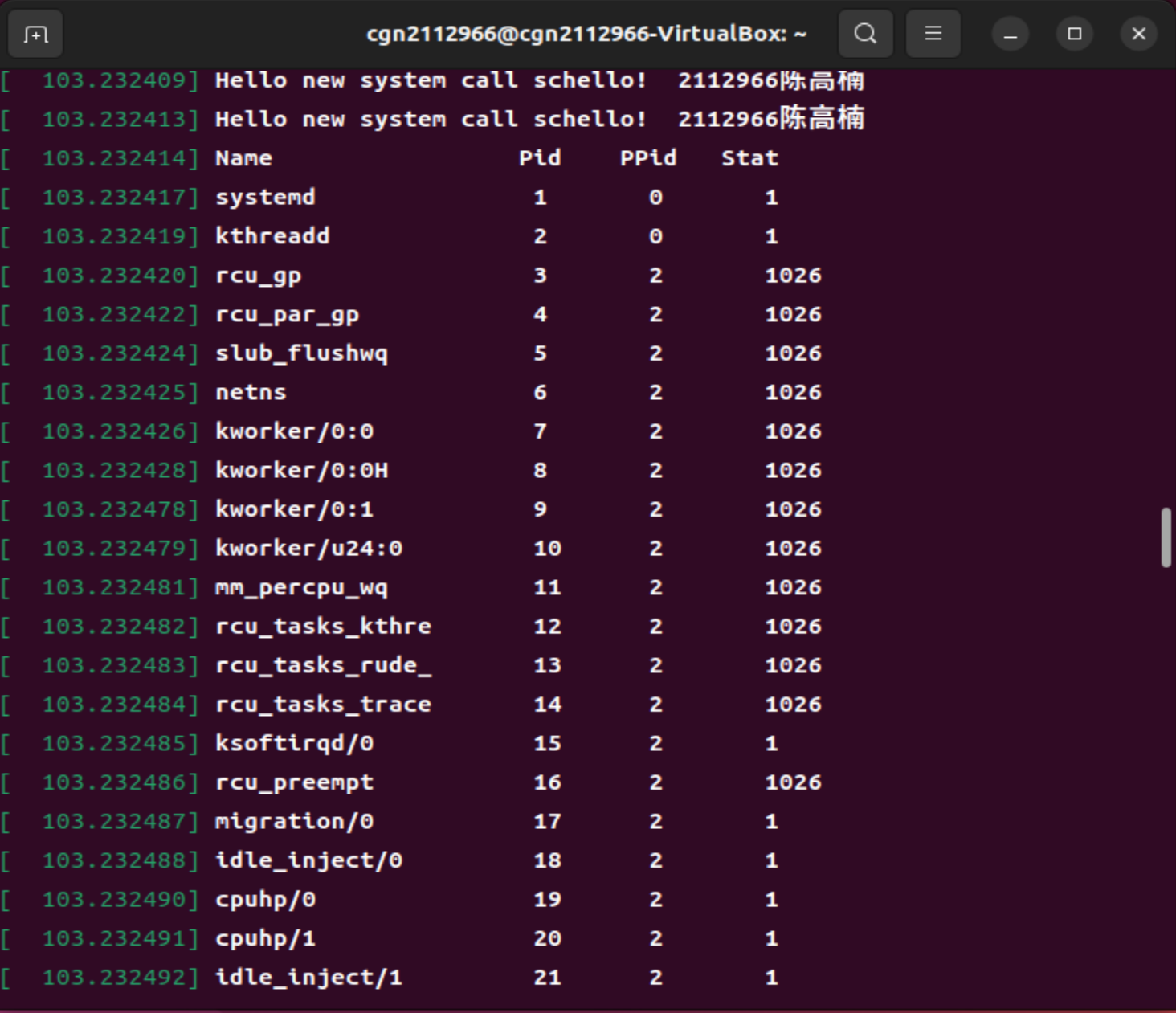
生成可执行文件



四．

运行结果如下

结果可查看进程



五．心得体会

通过这次实验，我深刻理解了Linux内核编程的基础概念和操作流程。在实验过程中，我首先在kernel/sys.c中添加了新的系统调用schello，通过编辑内核代码并亲自编译内核。我对操作系统有了更深的理解。