## Add a New System Call with args to list all processes

实验题目：添加系统调用列出所有进程

1. 实验目标

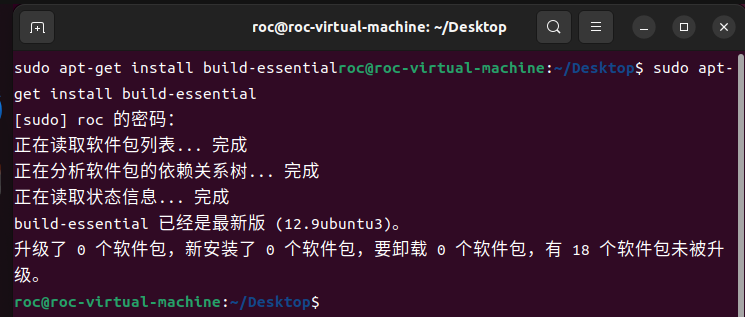
通过本实验，在Linux内核中添加一个新的系统调用，该系统调用能够返回所有进程的信息给用户。

2. 实验环境

操作系统：Linux

内核版本：Linux Kernel 4.x

工具：GCC Software Collection



3. 实验步骤

3.1 修改内核头文件

在include/linux/syscalls.h中添加新系统调用的定义：

asmlinkage long sys\_alcall(int cmd, char\* buf);

3.2 修改内核实现

在kernel/sys.c中修改系统调用的实现，以返回所有进程的信息：

SYSCALL\_DEFINE2(alcall, int, cmd, char\*, buf)

{

struct task\_struct \*p;

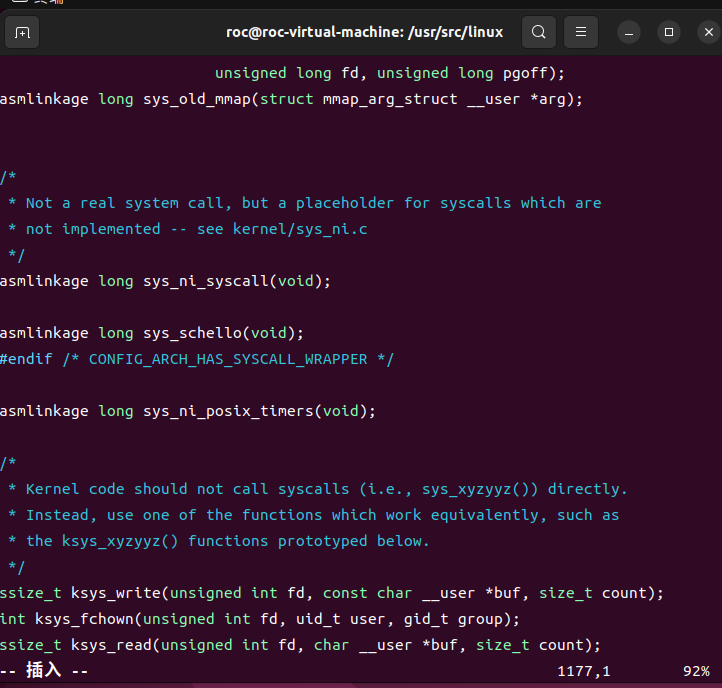
printk("Hello new system call alcall (%d,%x)!\n", cmd, buf);

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

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

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

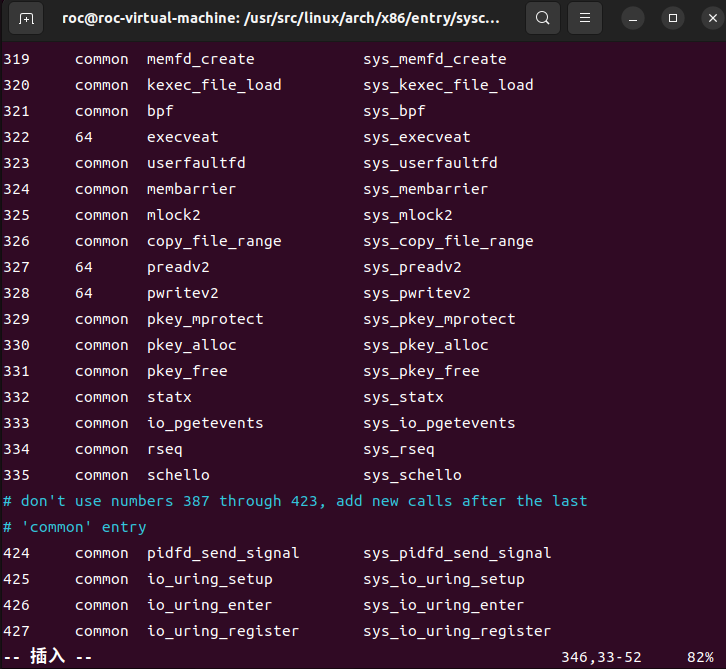
return 0;

}

3.3 修改系统调用表

在arch/x86/entry/syscalls/syscall\_64.tbl中添加新系统调用的信息：

440 common alcall \_\_x64\_sys\_alcall



3.4 重新配置和编译内核

执行以下步骤：

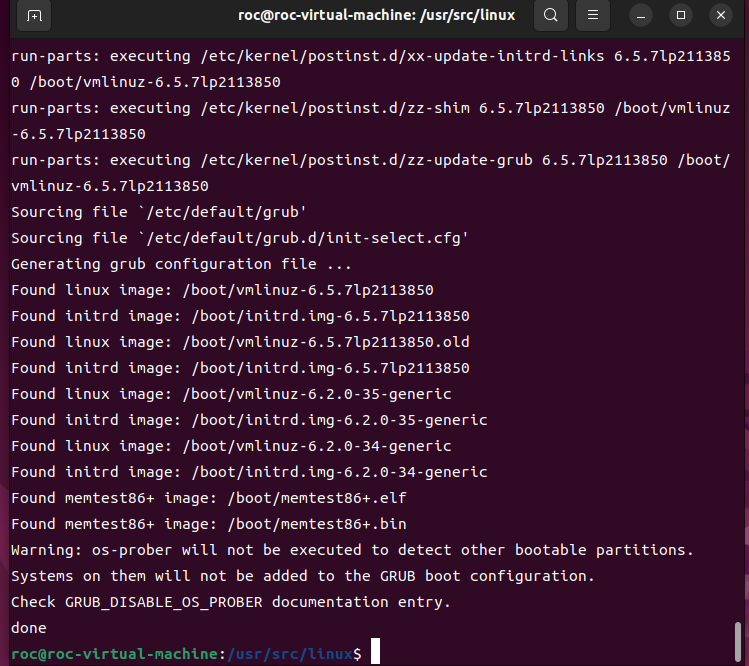
make clean

make -j5

sudo make modules\_install

sudo make install

Reboot



3.5 编写用户态测试程序

编写用户态测试程序 testalcall.c，用于测试新系统调用：

#include <unistd.h>

#include <sys/syscall.h>

#include <sys/types.h>

#include <stdio.h>

#define \_\_NR\_alcall 335

long alcall(int cmd, char\* buf) {

return syscall(\_\_NR\_alcall, cmd, buf);

}

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

int cmd;

char buf[256];

cmd = 9;

alcall(cmd, buf);

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

return 0;

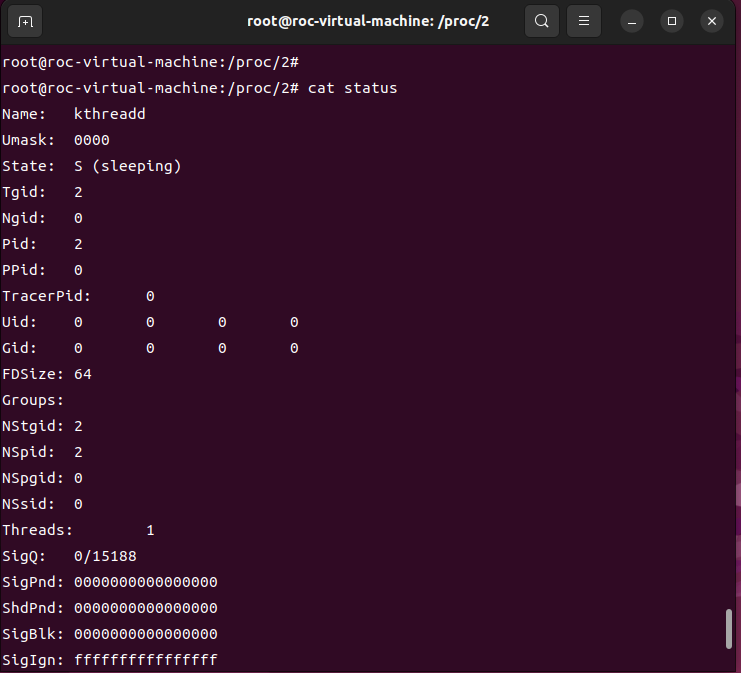
}

编译并执行测试程序：

gcc -o testalcall testalcall.c

./testalcall

$dmesg | grep alcall



4. 实验结果

通过运行测试程序，可以观察到新系统调用的输出信息，包括所有进程的名称、进程ID和状态。

5. 实验总结

通过本次实验，成功添加了一个带参数的新系统调用，实现了返回所有进程信息的功能。深入了解了内核编程、系统调用的添加和使用过程，提高了对内核的理解。