# 《操作系统》课第五次实验报告

|  |  |
| --- | --- |
| 学院: | 软件学院 |
| 姓名: | 郁万祥 |
| 学号: | 2013852 |
| 邮箱: | yuwanxiang0114@163.com |
| 时间: | 2022.10.14 |

## 开篇感言

经过上次系统调用的实验学习，这次再次添加新的系统调用的过程比较熟练了，所以此次实验的难点就是，如何借助 task\_struct 对系统的进程信息进行展示，这主要借助于 <linux/sched.h> 文件当中对于task\_struct结构中含有的信息的定义，我们进行直接的调用输出即可。

## 实验题目

Add a new system call to list all processes

## 实验目标

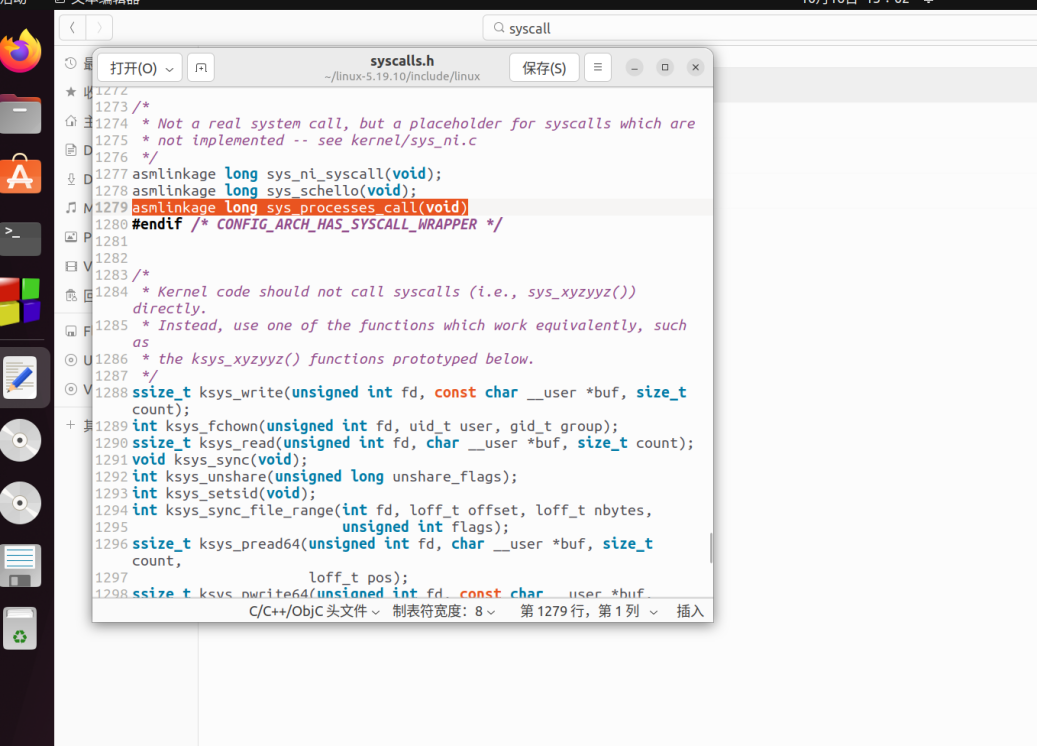
1. 添加新的系统调用
2. 系统调用实现罗列出进程的有关信息

## 原理方法

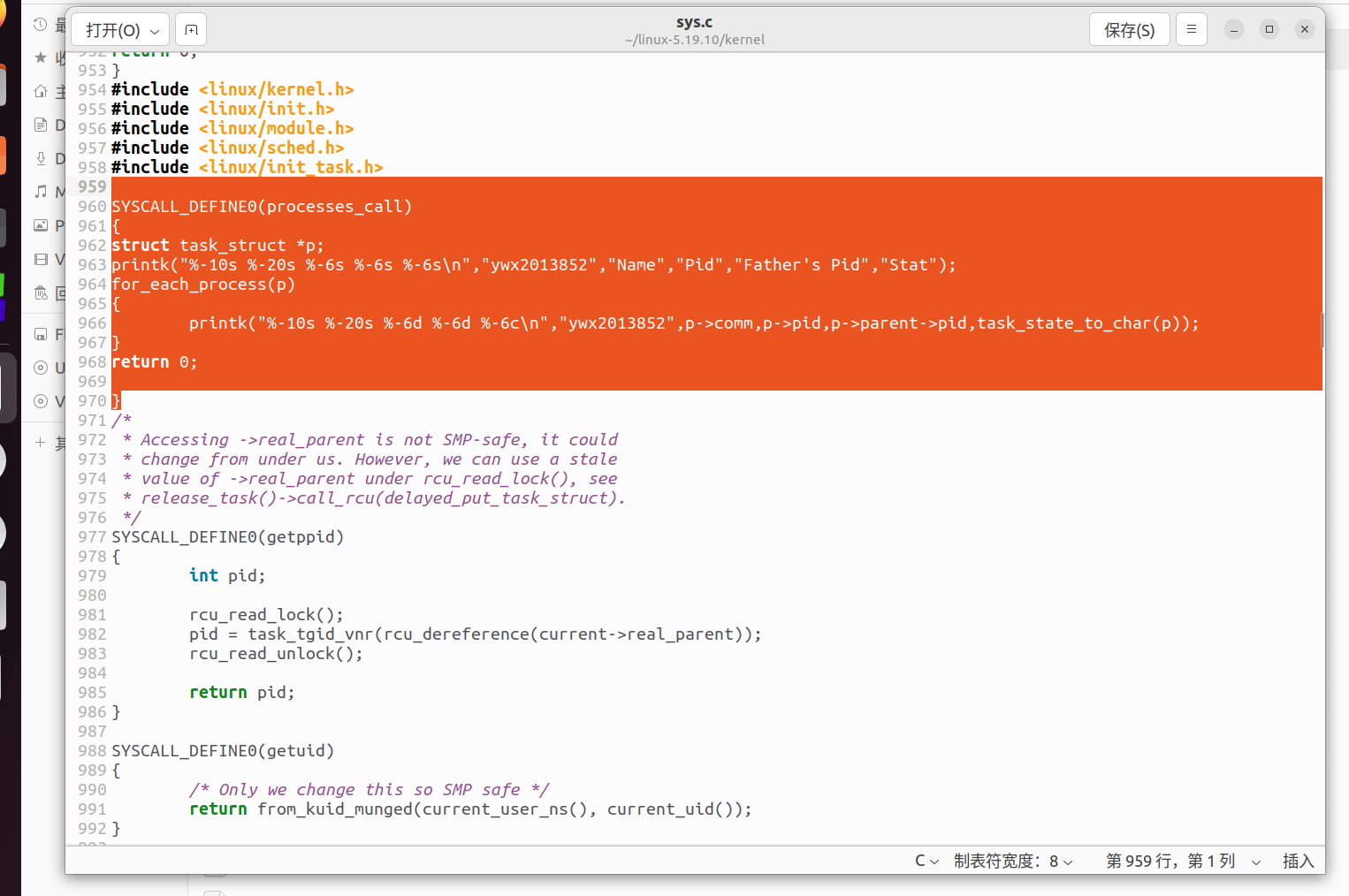
每个进程在内核中都有一个进程控制块(PCB)来维护进程相关的信息,Linux内核的进程控制块是task\_struct结构体。task\_struct是Linux内核的一种数据结构，它会被装载到RAM中并且包含着进程的信息。

## 具体步骤

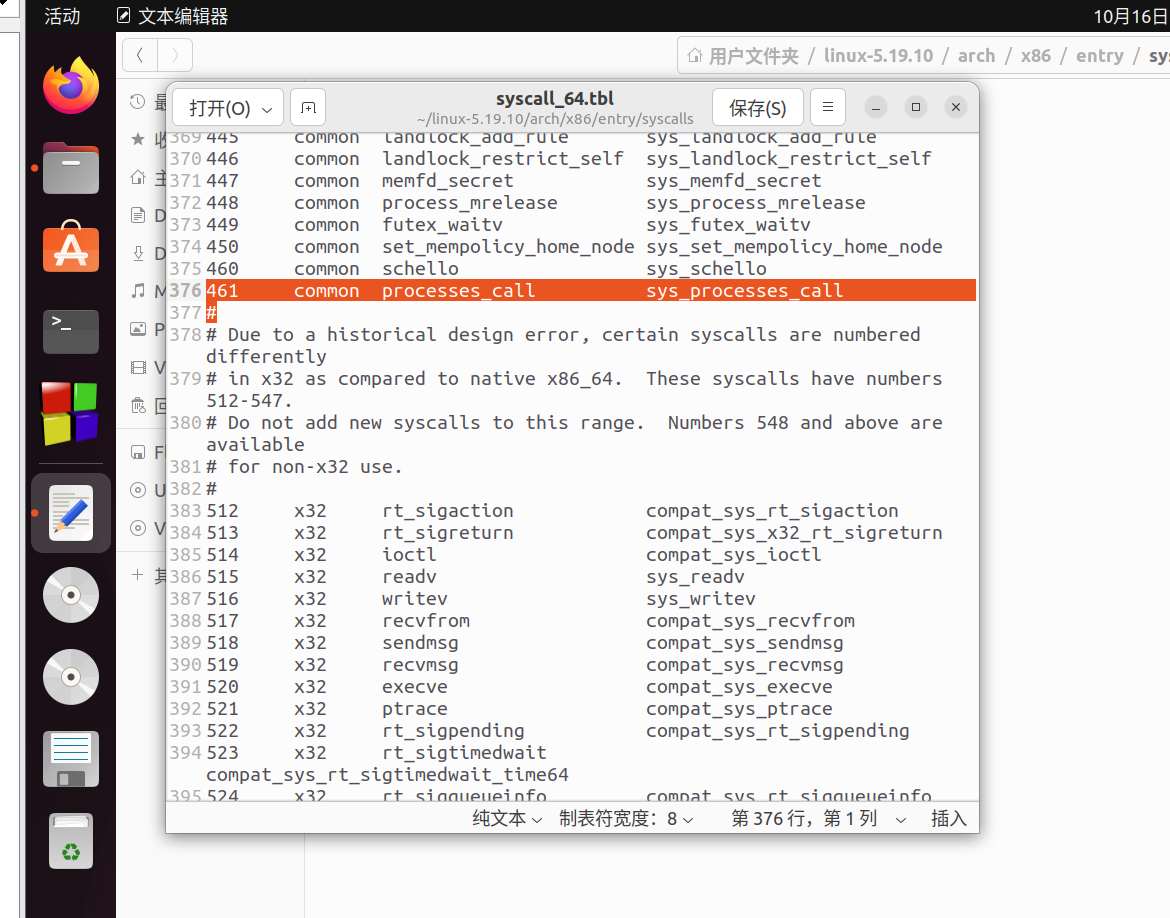
1、在syscall.h头文件当中添加asmlinkage long processes\_call(void);



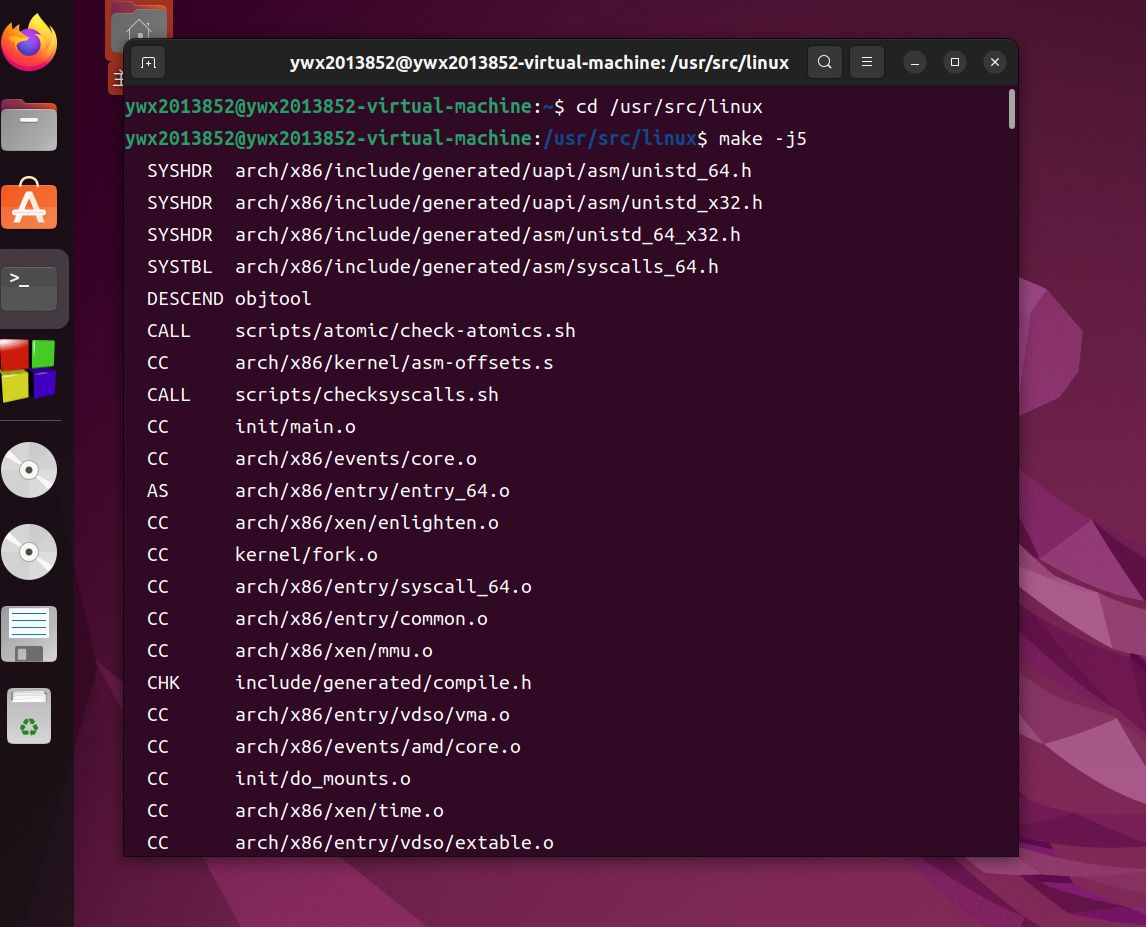
2、在kernal/sys.c中添加自己的服务函数



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。

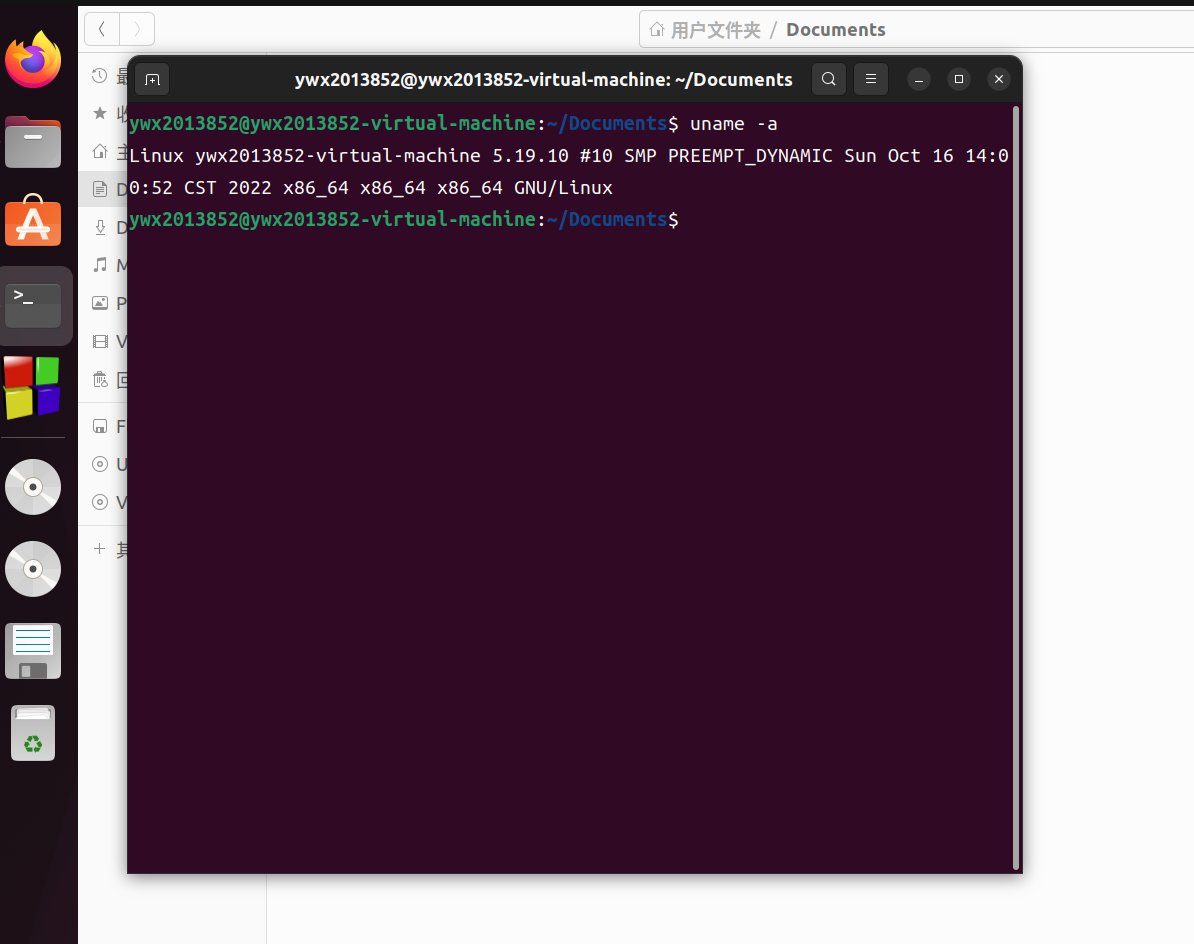


4、重新编译内核

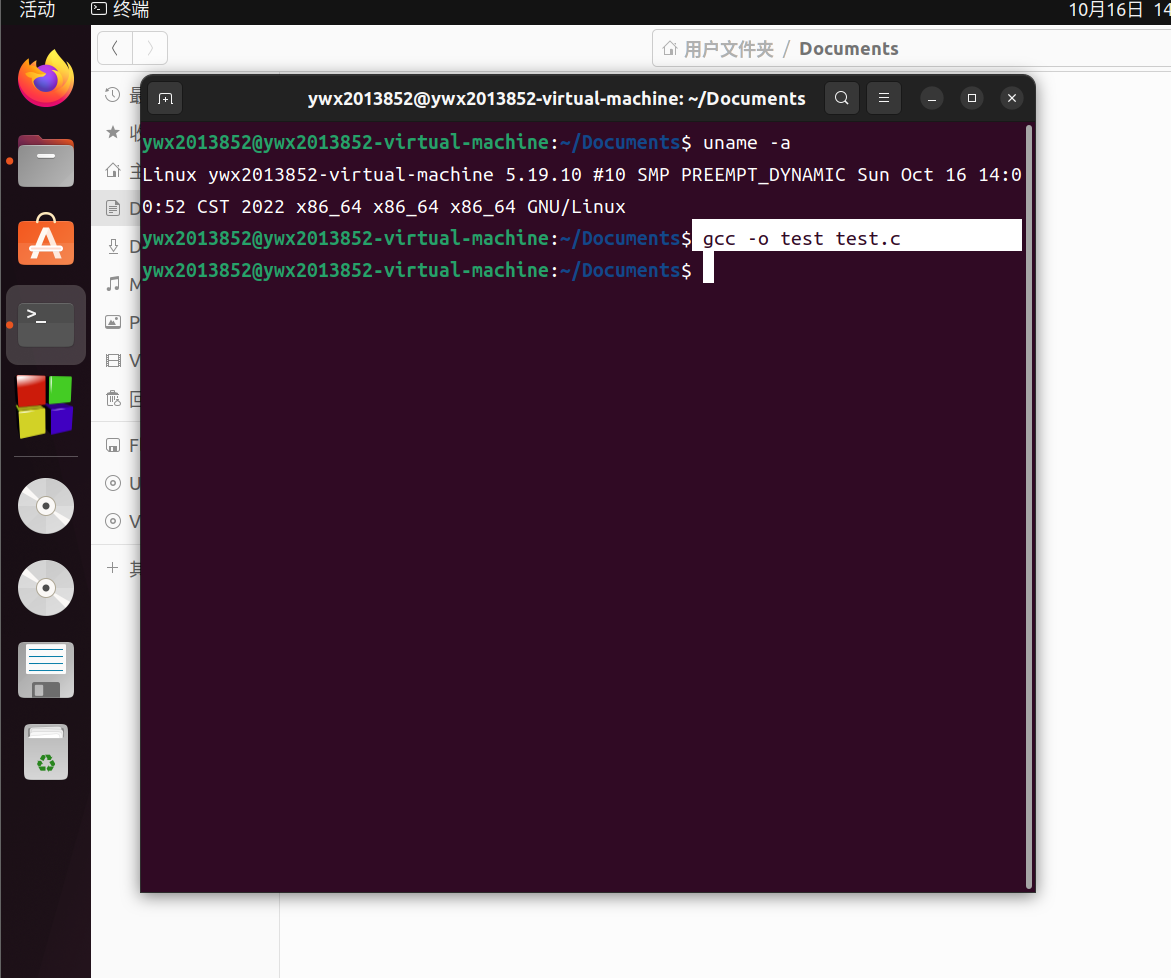


5、重新启动后，检查内核版本，编写demo进行系统调用，进行测试：

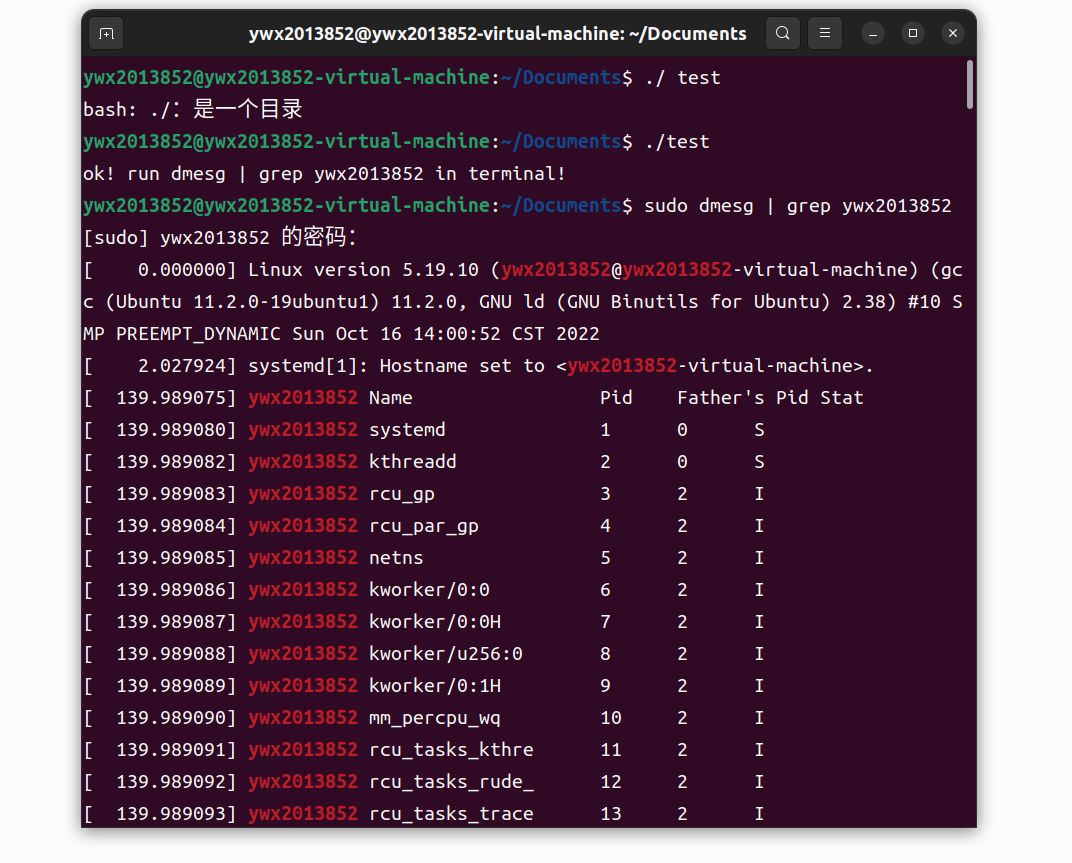
内核版本检查：



系统调用c文件编译：



调用结果：



## 总结心得

task\_struct 数据结果保存了linux当前的进程信息，并且，所有运行在系统中的进程都以task\_struct链表的形式存在在内核当中，对于task\_struct，本次使用到了：

1. 进程的状态 state
2. 进程的唯一表示 pid
3. 父进程 parent
4. 进程的名字 comm

因为进程结构体在内核当中，都是以链表形式存在，因此我们在以C语言编写的linux内核之下，借助于指针，通过遍历列表，就可以得到系统运行的所有进程，通过访问指针的成员，就可以得到上述的所有进程的信息。

## 参考资料

源代码

服务函数：

SYSCALL\_DEFINE0(processes\_call)

{

struct task\_struct \*p;

printk("%-10s %-20s %-6s %-6s %-6s\n","ywx2013852","Name","Pid","Father's Pid","Stat");

for\_each\_process(p)

{

printk("%-10s %-20s %-6d %-6d %-6c\n","ywx2013852",p->comm,p->pid,p->parent->pid,task\_state\_to\_char(p));

}

return 0;

}

调用文件：

#include<unistd.h>

#include<sys/syscall.h>

#include<sys/types.h>

#include<stdio.h>

#define \_NR\_processes\_call 461

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

{

syscall(\_NR\_processes\_call);

printf("ok! run dmesg | grep ywx2013852 in terminal!\n");

return 0;

}