**操作系统课程设计实验报告**

实验题目： Linux 内核模块编程

姓 名： 廖越强

学 号： 18041618

组 号： NULL

专 业： 网络工程

班 级： 网工二班

老师姓名： 张祯

日 期： 2020 年 3 月 14 日

目 录

[一 题目介绍 1](#_Toc515881940)

[二 实验思路 1](#_Toc515881941)

[三 遇到问题及解决方法 2](#_Toc515881942)

[四 核心代码及实验结果展示 3](#_Toc515881943)

[五 个人实验改进与总结 6](#_Toc515881944)

[5.1 个人实验改进 6](#_Toc515881945)

[5.2 个人实验总结 7](#_Toc515881946)

[六 参考文献 7](#_Toc515881947)

(大家注意，目录是自动生成的，页码从正文部分开始，当同学们把正文写完后，只需要右击目录，选择更新域，目录会自动更新)

# 一 题目介绍

**实验题目内容：**

1. 设计一个模块，要求列出系统中所有内核线程的程序名、PID 号、进程状态及进程优先级。
2. 设计一个带参数的模块，其参数为某个进程的 PID 号，该模块的功能是列出该进程的家族信息，包括父进程、兄弟进程和子进程的程序名、PID号。

**考察内容：**了解模块的基本概念、原理及实现技术。并且掌握基本的模块编程能力，能够自主设计模块功能并使用。

**问题关键：**了解模块结构后，只要能够获取到系统进程的相关信息（结构体），输出相关信息，就可以解决题目了

（大家注意，正文部分格式我已设定好，大家不要改，按这个格式书写就可以了）

**（正文部分一律用 宋体，五号字，1.25倍间距，首行缩进两格）**

**（根据实验设计指导书，描述实验题目内容，考察的知识点，问题的关键点）**

# 二 实验思路

**（再次强调，此处需要画人肉眼能清晰的流程图，不清晰的图不要放）**

**（图居中，并且下面的图注格式为： 图1. 名字 ）**

**（对于每个流程图需要配以文字详细阐述其内容，及其关键点，如实验中用到的每个信号量，为什么要用这个信号量，为什么用两个就够了，而不用三个）**

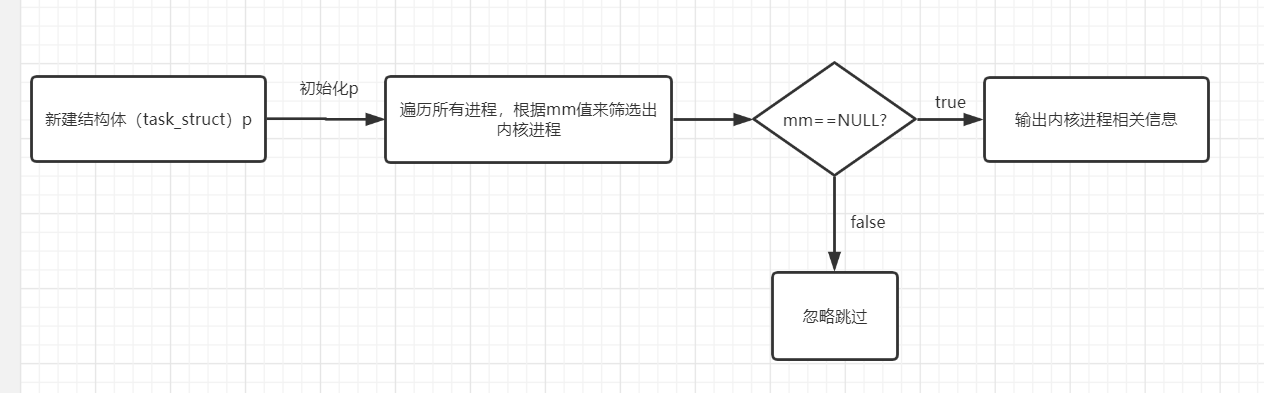


图1. 第一个实验（内核信息输出）

初始化p使用了&init\_task，&init\_task是kernel的第一个进程，0号进程。遍历采用内核提供的遍历方式for\_each\_process。成员mm是进程用户地址空间的描述符。内核线程的mm成员为空，可据此筛选出内核进程

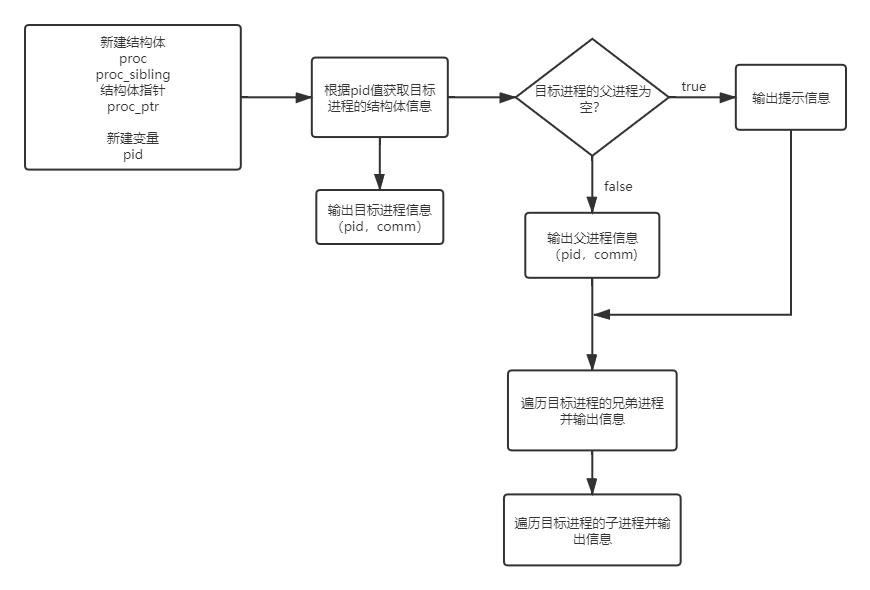


图2. 第二个实验，列出进程家族信息

proc:目标进程

proc\_sibling:目标进程的子进程,或者是目标进程的父进程的子进程（即目标进程的兄弟进程）

proc\_ptr:结构体指针,用于指向父进程/子进程头节点

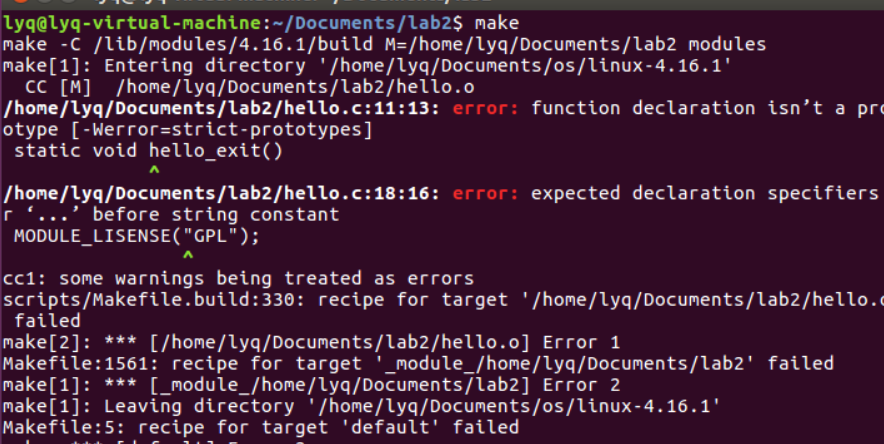
pid:目标进程号

遍历采用内核提供的遍历方式list\_for\_each(pos, head)。pos即为proc\_ptr，head为proc->children(子进程)或者proc->parent->children（兄弟进程）。对子进程结构体的获取则使用函数list\_entry(ptr, type, member)，并将返回的进程结构体信息存放在proc\_sibling中，最后输出信息pid和comm。

# 三 遇到问题及解决方法

**（此处请同学们务必认真规范书写，对于每个实验，写几条遇到的问题，以条目化形式呈现，如 1. 2. 3. ）**

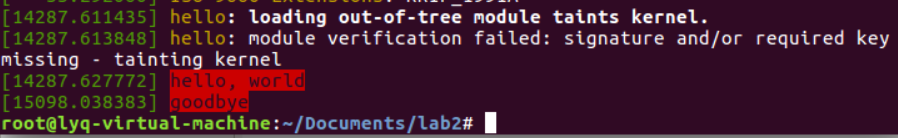
1. 编写加载函数和卸载函数时，没有形参的话，要写上void，否则会报错



1. 除了make外其他命令都需要在root权限下使用，否则会报错提示没有权限



1. 在使用dmesg查看信息时，发现了报错提示模块验证失败



module没有签名，可以在Makefile中加入CONFIG\_MODULE\_SIG=N关闭模块验证

# 四 核心代码及实验结果展示

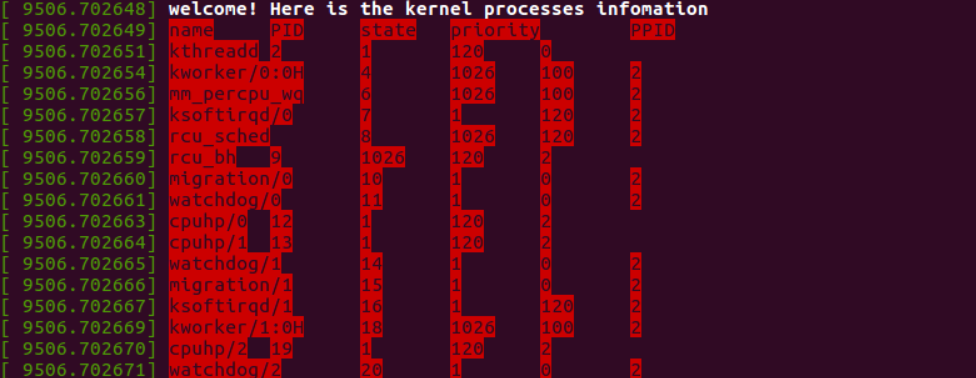
**（不需要粘贴所有的代码，把核心的代码粘贴即可，实验结果也需要截图放进来，每个图需要按照图片格式编写）**

**第一个实验：**内核信息输出

（1）核心代码：

1. **static** **int** function\_init(**void**)
2. {
3. **struct** task\_struct \*p;
4. p = NULL;
5. printk("welcome! Here is the kernel processes infomation\n");
6. printk(KERN\_ALERT"name\tPID\tstate\tpriority\tPPID\t");
7. for\_each\_process(p)
8. {
9. **if**(p->mm == NULL){ //the mm of kernel thread is NULL,so here only kernel threads will be output
10. printk(KERN\_ALERT"%s\t%d\t%ld\t%d\t%d\n", p->comm, p->pid,p->state, p->normal\_prio, p->parent->pid);
11. }
12. }
13. **return** 0;
14. }

（2）图片结果：



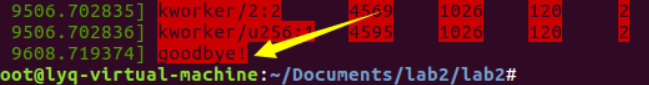


图1、第一个实验

**第二个实验：**列出进程家族信息

（1）核心代码：

1. **static** **int** proc\_family\_info\_init(**void**)
2. {
3. **struct** task\_struct \*proc;
4. **struct** list\_head \*proc\_ptr;
5. **struct** task\_struct \*proc\_sibling;
7. /\*current pid\*/
8. proc = pid\_task(find\_vpid(pid), PIDTYPE\_PID);
9. printk("current process--pid:%d name:%s\n", proc->pid, proc->comm);
11. **if**(proc->parent == NULL){
12. printk("no parent process\n");
13. }
14. **else** {
15. printk("parent process--pid:%d name:%s\n", proc->parent->pid, proc->parent->comm);
16. }
18. /\*brother process\*/
19. list\_for\_each(proc\_ptr, &proc->parent->children){
20. proc\_sibling = list\_entry(proc\_ptr, **struct** task\_struct, sibling);
21. printk("sibling process--pid:%d name:%s\n", proc\_sibling->pid, proc\_sibling->comm);
22. }
24. /\*children process\*/
25. list\_for\_each(proc\_ptr, &proc->children){
26. proc\_sibling = list\_entry(proc\_ptr, **struct** task\_struct, sibling);
27. printk("children process--pid:%d name:%s\n", proc\_sibling->pid, proc\_sibling->comm);
28. }
29. **return** 0;
30. }
31. 图片结果：

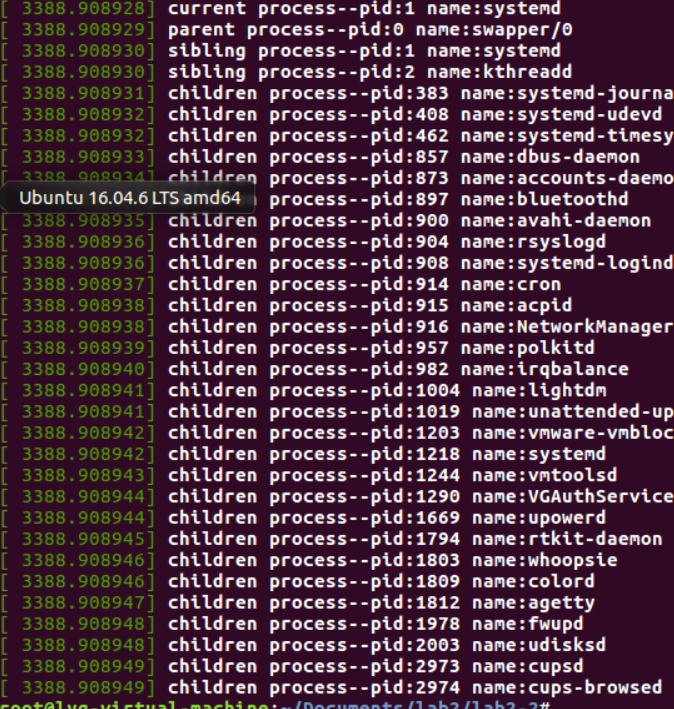




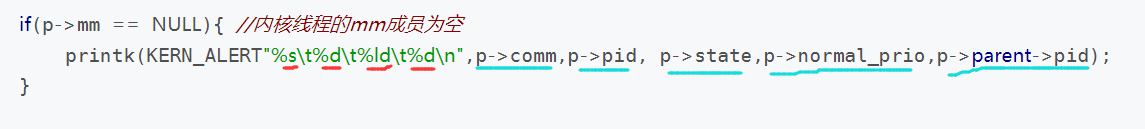
图2、第二个实验

# 五 个人实验改进与总结

## 5.1 个人实验改进

**（大部分同学的代码都是参考网上资料，或者借鉴已完成同学的，此处重点突出自己如何改进，为什么这样改进，说出自己改进部分的优缺点，有效文字不少于200字）**

1、在第一个实验（列出所有内核进程信息）中，网上搜到的代码大多都是同一个版本，而这个版本的作者其实写错一些东西，惊人的是那么多写博客的人居然没有把它改过来。估计都是直接把代码复制粘贴了。原作者在写输出进程信息（printk）的时候，变量数搞错了，最后一个p->parent->pid是不会输出的。只要在前面再额外加一个%d就行了



2、在阅读源码的过程中，我发现了第一个实验中遍历前的proc=&init\_task这一步其实是多余的。来看看for\_each\_process的源码，可见上述步骤是多余的重复工作。

#define for\_each\_process(p)

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

3、重新起了变量名吧，提高可读性。

## 5.2 个人实验总结

**（此处类似于写观后感，同学们辛辛苦苦完成实验，单独编译内核可能就折腾很久，相信会有很多的感触和收获，让你的笔在第一时间记录这美好的瞬间，有效文字不得少于200字）**

实验二总的来说难度不大吧，时间主要都花在了找有关处理进程的内核函数，像遍历进程for\_each\_process这些，以及阅读源码上。即便是2.6x的内核源码，读起来也是有点吃力的。

实验一(指第一份实验报告的修改nice值实验)与实验二有一个共同的地方就是都需要获取进程的PCB(task\_struct)，这里实验一和实验二在获取pid\_task(struct pid \* pid, enum pid\_type)的第一个参数时，使用了不同的获取方法。实验一中使用了find\_get\_pid()而这里使用了find\_vpid()，二者的返回值是一样的，都是struct pid，看看源码

struct pid \*find\_get\_pid(pid\_t nr)

{ struct pid \*pid;

rcu\_read\_lock();

pid = get\_pid(find\_vpid(nr));

rcu\_read\_unlock();

return pid;}

可见，find\_get\_pid包含了find\_vpid。find\_get\_pid多了读锁和释放读锁操作。

代码编写完后，模块编译安装卸载方面倒是没有太多坑，规规矩矩完成。写报告花的时间也是相当的长啊！我感觉都快赶上做实验的时间了。

# 六 参考文献

**（此处罗列同学们在做实验过程中 所参考的文献、书籍、网站， 或者向某个高手同学请教都可以写， 大多数同学应该是参考 网站、书籍和问人比较多，以条目化的形式呈现， 如1. 2. 3. ）**

内核2.6源码

<https://elixir.bootlin.com/linux/v2.6.28.6/source/include>

task\_struct信息（因为这里源码有点难理清，不是很好理解，于是找了别人整理的）

<https://blog.csdn.net/npy_lp/article/details/7292563>