实验报告

学号：15051415　姓名：冯冠玺

实验题目：Linux 内核模块编译

1、实验要求

（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

（3）请根据自身情况，进一步阅读分析程序中用到的相关内核函数的源码实现。

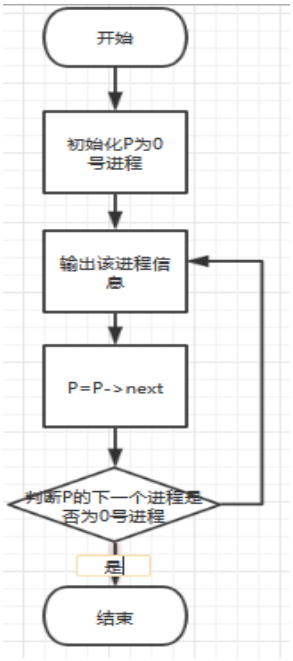
2、核心思路及代码

第一部分的代码思路比较简单，主要是通过 Linux 系统中的 for_each_process 宏，从 0 号进程开始（initial_task 开始），遍历所有进程，并把每一个进程的信息打印出来。这里需要注意的是：

```
#define for_each_process(p)
  for (p = &init_task ; (p = next_task(p)) != &init_task;)
```

init_task 是 0 号进程并且，Linux 把所有进程的任务结构体相互链接成一个环形双向链表，即最后一个进程的任务结构体的下一个是 0 号进程。因此遍历完所有进程的标志是——p = next_task(p)) != &init_task。

具体算法流程图为：



第二个部分的关键是理解 list_for_each 与 list_entry 两者之间是如何协调，找到一个进程的 task_struct 结构体的。因为，在 Linux 中一个简单的链表结构示意是这样的——

```
struct list_head{
  struct list_head *next,*prev;
};
struct numlist{
  int num;
  struct list_head list;
};
```

通过 list_for_each 得到的只是下一个结构体的 struct 中的 list_head，因此需要 list_entry 函数来逆向减去 struct 的数据域大小，从而真正找到节点的起始位置，以便引用节点中的其他域。

算法流程图为：

3、问题与解决方案

问题 1：在 make 时，找不到目标文件

解决方案：这个问题基本上是一个失误。因为我将 Makefile 文件命名为了 MakeFile，从而导致出错。

问题 2:报错：for_each_process 函数没有声明

解决方案：这个问题的原因在于，不同的 Linux 内核版本。在 Linux4.14 以后，for_each_process 函数在#include <linux/sched/signal.h>中。

源码：

模块一

```
#include<linux/init.h>
#include<linux/module.h>
#include<linux/kernel.h>
#include<linux/sched.h>
#include <linux/sched/signal.h>
static int __init ltmodule_1_init(void){
  struct task_struct *p;
  printk("Name\tPID\tState\tPrio\t\n");
  for_each_process(p)
     printk("%s\t%d\t%ld\t%d\n",p->comm,p->pid,p->state,p->prio); //输出进程的名称,pid 号,状态,优先级
  return 0;
}

static void __exit ltmodule_1_exit(void){
   printk("lt's module01 exit!\n");
}

module_init(ltmodule_1_init);
module_exit(ltmodule_1_exit);
MODULE_LICENSE("GPL");
```

模块一 Makefile 文件

```
obj-m := xfmodule_1.o
    KDIR := /lib/modules/$(shell uname -r)/build
    PWD := $(shell pwd)
default:
    make -C $(KDIR) M=$(PWD) modules
clean:
    make -C $(KDIR) M=$(PWD) clean
```

模块二

```
#include<linux/init.h>
#include<linux/module.h>
#include<linux/kernel.h>
#include<linux/sched.h>          //有进程描述符 task_struct
#include<linux/moduleparam.h>     //模块带参数
```

```c
static int pid;                 //参数申明

module_param(pid,int,0644);       //参数说明

static int __init ltmodule_2_init(void){
  struct task_struct *p;
  struct task_struct *parent;
  struct task_struct *process;
  struct list_head *list;

  printk("lt's Process Begin!\n");
  printk("与之关系\t\t\t 进程名\t\tPID\n");

  //根据 pid 找到进程的地址
  p=pid_task(find_vpid(pid),PIDTYPE_PID);

  printk("自己\t\t\t%s\t\t%d\n",p->comm,p->pid);  //输出自己的信息

  parent=p->parent;             //父进程
  printk("父进程\t\t\t%s\t\t%d\n",parent->comm,parent->pid);
  list= &parent->children;
  list_for_each(list,&parent->children)  //遍历 parent 的 children，即是
他的兄弟
  {
       process=list_entry(list,struct task_struct,sibling);
       printk("兄弟进程\t\t\t%s\t\t%d\n",process->comm,process->pid);
  }

   list=&p->children;     //子进程
   list_for_each(list,&p->children)       //遍历子进程
  {
         process=list_entry(list,struct task_struct,sibling);
         printk("子进程\t\t\t%s\t\t%d\n",process->comm,process->pid);
  }

   return 0;
}


static void __exit ltmodule_2_exit(void){
   printk("lt's module_2_2 exit!\n");
}
```

```
module_init(ltmodule_2_init);
module_exit(ltmodule_2_exit);
MODULE_LICENSE("GPL");
```

模块二 Makefile 文件

```
obj-m := xfmodule_2.o
  KDIR := /lib/modules/$(shell uname -r)/build
  PWD := $(shell pwd)
default:
  make -C $(KDIR) M=$(PWD) modules
clean:
  make -C $(KDIR) M=$(PWD) clean
```