Linux 内核实验报告

实验题目：实验四-动态模块设计实验

学号：　200900301236　　（辅修号：　）
日期：2012.5.5　　班级：09 软 1 姓名：王添枝
Email：tzwang2012@163.com

实验目的：能够自己设计动态模块，并能够将其添加到内核中。懂得在用户程序中调用添加的动态模块。

硬件环境：
软件环境：ubuntu10.10
linux 内核:2.6.35.13
gcc:4.4.5

实验步骤：

一、实验原理：

模块一旦被转载进系统，就会在系统态下运行。如果模块知道系统的数据结构地址，就可以读写相应的数据结构。模块作为一种抽象的数据类型，他有一个可以通过静态内核中断的接口。加入模块的每个新函数必须在该模块被装载到内核时进行注册。如果模块式静态装载的，加入模块的所有函数在内核启动时候进行注册。如果是动态装载的，必须在装载这个模块时候动态注册。注册工作在 init_module() 中完成，注销工作是在 cheanup_module()中完成。

对于模块的使用有两种方式。一种是设备驱动程序，一种是/proc 文件。

二、实验步骤

1.问题 A

编译内核模块，用命令：sudo insmod main.ko 安装编译好的内核模块。再写一个测试程序测试安装好的模块：从/proc/mydir/myfile 中读取数据，并用 gettimeofday()获取当前的时间，进行比较。

2.问题 B

编写内核模块，其中读取当前进程号为 current->pid，进程名为 current->comm。从当前进程开始往上遍历，直到进程号为 1。从 init_task 开始用 for_each_process 宏遍历当前的任务队列。

3.问题 C

编写内核模块，带有一个参数，若参数为 0，则为问题 B 的功能，否则为读取 jiffies 的值。安装时的命令为:sudo insmod exp4c.ko param=1。

三、调试记录

1.问题 A

安装模块的命令：sudo insmod main.ko

```
wangtianzhi@wangtianzhi-NV48:~/桌面/linux内核实验/实验四$ ./exp4a
gettimeofday=1336820173
jiffies=1213861
```

2.问题 B

安装模块的命令:sudo insmod exp4b.ko

3.问题 C

安装模块的命令:sudo insmod exp4c.ko

结论分析与体会:

模块是系统中的一个重要组成部分，它使添加内核功能变得更加容易了。本实验的内容相对较简单，但是却为以后的实验，甚至将来的内核编程奠定了基础。

程序完整源码

1.问题 A

(1)模块源码 main.c

```
/
*************************************************************
****************
                          main.c    -    description
     begin                 : 一  4月    4 21:01:11 CST 2003
     copyright             : (C) 2003 by 张鸿烈
     Function              : 编写创建 proc 文件系统的模块,该程
序创建在/proc 目录下
                           : 创建 mydir 目录,在 mydir 目录下创
建保存当前系统时间
             : jiffies 值的文件 myfile,
*************************************************************
****************/
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/proc_fs.h>
#include <linux/jiffies.h>
#include <asm/uaccess.h>
#include <linux/moduleparam.h>
#define MODULE_NAME "Myproc"
#define MYDATA_LEN   16

//放用户空间传入的数据
struct my_proc_data{
   char value[MYDATA_LEN];
};
```

```
struct my_proc_data mydata;
//proc结构变量
static struct proc_dir_entry *example_dir;
static struct proc_dir_entry *date_file;
static int   param;
module_param(param, int , 0644);

//读文件 myfile的读驱动函数
static int proc_read(char *page, char **start,off_t off,int
count,int *eof,void *data ){
    int len=0 ;
     struct my_proc_data *mydatap = (struct my_proc_data *)
data;
    len+=sprintf(page,"%s%ld\n",mydatap->value,jiffies);
    return len;
}
//写文件 myfile 的写驱动函数
static   int   proc_write(struct   file   *file,const   char
*buffer,unsigned long count,void *data){
    int len ;
     struct my_proc_data *mydatap = (struct my_proc_data *)
data;
    if(count>MYDATA_LEN)
       len = MYDATA_LEN;
    else
       len = count;
    if(copy_from_user(mydatap->value,buffer,len)){
       return -EFAULT;
    }
    mydatap->value[len-1] = '\0';
    return len;
}
//装入模块
int init_module(void)
{
   //创建 proc/myfile目录
         example_dir   =   (struct   proc_dir_entry   *   )
proc_mkdir("mydir",0);
   if(example_dir == 0){
        printk("mkdir fail\n");
      return -1;
   }
   //创建/proc/mydir/myfile 文件
         date_file   =   (struct   proc_dir_entry   *   )
```

```
create_proc_entry("myfile",0666,example_dir);
   if(date_file == 0){
      remove_proc_entry("myfile",0);
      printk("mkfile fail\n");
      return -ENOMEM;
   }
   strcpy(mydata.value,"Ticks=");
   date_file->data=&mydata;
   date_file->read_proc=&proc_read;
   date_file->write_proc=&proc_write;
   //date_file->owner=THIS_MODULE;
   return 0;
}
//卸载模块
void cleanup_module(void)
{
   remove_proc_entry("myfile",example_dir);
   remove_proc_entry("mydir",NULL);
   printk("Goodbye.\n");
}
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Test");
MODULE_AUTHOR("xxx");
```

(2)测试程序 exp4a.c

```
/*************************************************************
File :exp4a.c
Copyright :(C)2012 by wangtianzhi
Function  : 分析实验以上模块，编写一个测试该模块的用户程序。比较该模块读取的时间和用 gettimeofday()读取的时间的精度。

*************************************************************/
#include <stdio.h>
#include <sys/time.h>

#define HZ 250
FILE *thisProcFile;
struct timeval now;
char str[20],c1,c2,c3,c4,c5,c6;
unsigned long jiffies;
int main(){
   gettimeofday(&now, NULL);
   thisProcFile = fopen("/proc/mydir/myfile","r");
   fgets(str,20,thisProcFile);
   sscanf(str,"%c%c%c%c%c%c
```

```
%ld",&c1,&c2,&c3,&c4,&c5,&c6,&jiffies);
   printf("gettimeofday=%ld\n",now.tv_sec);
// float stime = ((float)jiffies)/HZ;
// printf("%f\n",stime);
   printf("jiffies=%ld\n",jiffies);
   close(thisProcFile);
   return 0;
}
```

2.问题B exp4b.c

```
/
*************************************************************
***************
                          main.c   -   description
     Function              : 编写创建proc文件系统的模块,该程序创建在/proc目录下
                           : 实现一个模块用它遍历当前进程的父进程和任务队列，并将遍历结果输出到一个proc文件中（遍历可以从current当前进程开始，父进程遍历到初始化进程，遍历任务队列可以利用for_each_process宏）
*************************************************************
***************/
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/proc_fs.h>
#include <linux/jiffies.h>
#include <asm/uaccess.h>
#include <linux/moduleparam.h>
#include <linux/list.h>
#include <linux/sched.h>
#define MODULE_NAME "Myproc"
#define MYDATA_LEN   8200

//放用户空间传入的数据
struct my_proc_data{
   char value[MYDATA_LEN];
};
struct my_proc_data mydata;
struct task_struct *task;
//proc结构变量
static struct proc_dir_entry *example_dir;
static struct proc_dir_entry *date_file;
static int   param;
module_param(param, int , 0644);
```

```
//读文件 myfile的读驱动函数
static int proc_read(char *page, char **start,off_t off,int
count,int *eof,void *data ){
    int len=0 ;
     struct my_proc_data *mydatap = (struct my_proc_data *)
data;
    strcpy(mydata.value,"The parents process：\n");
    sprintf(mydata.value,"%s %d %s\n",mydata.value,current-
>pid,current->comm);//进程ID，进程名称
   task = NULL;
   task = current;
   //从初始进程开始向上遍历
   while(task->pid!=1){
      task=task->parent;
      sprintf(mydata.value,"%s %d %s \n",mydata.value,task-
>pid, task->comm);
   }
   task = &init_task;//arch/x86/kernel.c
  // printk("当前的进程是：%d %s \n",current->pid,current-
>comm);
    sprintf(mydata.value,"%s %s \n",mydata.value, "Current
task_struct:");
   for_each_process(task){
      sprintf(mydata.value,"%s %d %s \n",mydata.value,task-
>pid,task->comm);
      printk("%d %s\n",task->pid, task->comm);
   }
    len+=sprintf(page,"%s\n",mydata.value);
    //len+=sprintf(page,"%s\n",mydatap->value);
    return len;
}
//写文件 myfile 的写驱动函数
static int proc_write(struct file *file,const char
*buffer,unsigned long count,void *data){
    int len ;
     struct my_proc_data *mydatap = (struct my_proc_data *)
data;
    if(count>MYDATA_LEN)
       len = MYDATA_LEN;
    else
       len = count;
    if(copy_from_user(mydatap->value,buffer,len)){
       return -EFAULT;
```

```
    }
    mydatap->value[len-1] = '\0';
    return len;
}
//装入模块
int init_module(void)
{
   //创建 proc/myfile目录
         example_dir   =   (struct   proc_dir_entry   *   )
proc_mkdir("exp4dir",0);
   if(example_dir == 0){
        printk("mkdir fail\n");
      return -1;
   }
   //创建/proc/mydir/myfile 文件
          date_file   =   (struct   proc_dir_entry   *   )
create_proc_entry("exp4b",0666,example_dir);
   if(date_file == 0){
      remove_proc_entry("exp4b",0);
      printk("mkfile fail\n");
      return -ENOMEM;
   }


   date_file->data=&mydata;
   date_file->read_proc=&proc_read;
   date_file->write_proc=&proc_write;
   //date_file->owner=THIS_MODULE;
   return 0;
}
//卸载模块
void cleanup_module(void)
{
   remove_proc_entry("exp4b",example_dir);
   remove_proc_entry("exp4dir",NULL);
   printk("Goodbye.\n");
}
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("exp4 partB");
MODULE_AUTHOR("wtz");
```

3.问题C　　exp4c.c

```
/
**********************************************************
***************
```

```
                        main.c   -   description
     Function                  : 编写创建proc文件系统的模块,该程序创建在/proc目录下
                               : 实现一个模块用它遍历当前进程的父进程和任务队列，并将遍历结果输出到一个proc文件中（遍历可以从current当前进程开始，父进程遍历到初始化进程，遍历任务队列可以利用for_each_process宏）
*************************************************************************/
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/init.h>
#include <linux/proc_fs.h>
#include <linux/jiffies.h>
#include <asm/uaccess.h>
#include <linux/moduleparam.h>
#include <linux/list.h>
#include <linux/sched.h>
#define MODULE_NAME "Myproc"
#define MYDATA_LEN   8200

//放用户空间传入的数据
struct my_proc_data{
   char value[MYDATA_LEN];
};
char str[20];
struct my_proc_data mydata;
struct task_struct *task;
//proc结构变量
static struct proc_dir_entry *example_dir;
static struct proc_dir_entry *date_file;
static int   param;
module_param(param, int , 0644);

//读文件 myfile的读驱动函数
static int proc_read(char *page, char **start,off_t off,int count,int *eof,void *data ){
    int len=0 ;
     struct my_proc_data *mydatap = (struct my_proc_data *)data;
    if(param ==0){
      strcpy(mydata.value,"The parents process: \n");
                              sprintf(mydata.value,"%s     %d %s\n",mydata.value,current->pid,current->comm);//进程ID，进
```

程名称

```
      task = NULL;
      task = current;
      //从初始进程开始向上遍历
      while(task->pid!=1){
         task=task->parent;
                     sprintf(mydata.value,"%s     %d     %s
\n",mydata.value,task->pid, task->comm);
      }
      task = &init_task;//arch/x86/kernel.c
          // printk("当前的进程是：%d %s \n",current-
>pid,current->comm);
            sprintf(mydata.value,"%s  %s  \n",mydata.value,
"Current task_struct:");
      for_each_process(task){
                     sprintf(mydata.value,"%s     %d     %s
\n",mydata.value,task->pid,task->comm);
         printk("%d %s\n",task->pid, task->comm);
      }
      len+=sprintf(page,"%s\n",mydata.value);
   }
   else{
      sprintf(str,"%s","jiffies=:");
      len+=sprintf(page,"%s%ld\n",str,jiffies);
   }
    //len+=sprintf(page,"%s\n",mydatap->value);
    return len;
}
//写文件 myfile 的写驱动函数
static  int  proc_write(struct  file  *file,const  char
*buffer,unsigned long count,void *data){
    int len ;
     struct my_proc_data *mydatap = (struct my_proc_data *)
data;
    if(count>MYDATA_LEN)
       len = MYDATA_LEN;
    else
       len = count;
    if(copy_from_user(mydatap->value,buffer,len)){
       return -EFAULT;
    }
    mydatap->value[len-1] = '\0';
    return len;
}
```

```
//装入模块
int init_module(void)
{
   //创建 proc/myfile目录
         example_dir   =   (struct   proc_dir_entry   *   )
proc_mkdir("part4dir",0);
   if(example_dir == 0){
        printk("mkdir fail\n");
      return -1;
   }
   //创建/proc/mydir/myfile 文件
          date_file   =   (struct   proc_dir_entry   *   )
create_proc_entry("part4",0666,example_dir);
   if(date_file == 0){
      remove_proc_entry("part4",0);
      printk("mkfile fail\n");
      return -ENOMEM;
   }


   date_file->data=&mydata;
   date_file->read_proc=&proc_read;
   date_file->write_proc=&proc_write;
   //date_file->owner=THIS_MODULE;
   return 0;
}
//卸载模块
void cleanup_module(void)
{
   remove_proc_entry("part4",example_dir);
   remove_proc_entry("part4dir",NULL);
   printk("Goodbye.\n");
}
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("exp4 partC");
MODULE_AUTHOR("wtz");
```

参考材料

linux 操作系统内核实习
linux 内核设计与实现