**实验2.3内核模块**

1、实验目的

模块是Linux系统的一种特有机制，可用以动态扩展操作系统内核功能。编写实现某些特定功能的模块，将其作为内核的一部分在管态下运行。本实验通过内核模块编程在/porc文件系统中实现系统时钟的读操作接口。

2、实验内容

设计并构建一个在/proc文件系统中的内核模块clock，支持read()操作，read()返回值为一字符串，其中包块一个空各分开的两个子串，为别代表xtime.tv\_sec和xtime.tv\_usec。

3、实验原理

Linux模块是一些可以作为独立程序来编译的函数和数据类型的集合。在装载这些模块式，将它的代码链接到内核中。Linux模块可以在内核启动时装载，也可以在内核运行的过程中装载。如果在模块装载之前就调用了动态模块的一个函数，那么这次调用将会失败。如果这个模块已被加载，那么内核就可以使用系统调用，并将其传递到模块中的相应函数。

4、实验步骤

（1）**编写内核模块**

文件中主要包含init\_module()，cleanup\_module()，proc\_read\_clock()三个函数。其中init\_module()，cleanup\_module()负责将模块从系统中加载或卸载，以及增加或删除模块在/proc中的入口。proc\_read\_clock()负责产生/proc/clock/my\_clock被读时的动作。

**内核模块源代码proc\_clock.c**

#include <linux/init.h>

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/proc\_fs.h>

#include <linux/fs.h>

#include <linux/vmalloc.h>

#include <asm/uaccess.h>

#include <linux/seq\_file.h>

//Prototypes

static int init\_clock(void);

static void cleanup\_clock(void);

static ssize\_t proc\_read\_clock(struct file \*file,char \_\_user \*pszPage,size\_t size,loff\_t \*off);

//#define MODULE

#define MODULE\_VERSION "1.0"

#define USER\_ROOT\_DIR "clock"

#define MODULE\_NAME "my\_clock"

//user defined directory

struct proc\_dir\_entry \*my\_clock\_dir;

struct proc\_dir\_entry \*my\_clock\_file;

static const struct file\_operations my\_clock\_fops =

{

.owner = THIS\_MODULE,

.read = proc\_read\_clock,

};

//Functions

static int init\_clock(void)

{

//Create user root dir under /proc

printk("clock: init\_module()\n");

my\_clock\_dir= proc\_mkdir(USER\_ROOT\_DIR,NULL);

my\_clock\_file = proc\_create(MODULE\_NAME,0,my\_clock\_dir, &my\_clock\_fops);

printk(KERN\_INFO"%s %s has initialized.\n",MODULE\_NAME,MODULE\_VERSION);

return 0;

}

static void cleanup\_clock(void)

{

printk("clock: cleanup\_module()\n");

remove\_proc\_entry(MODULE\_NAME,my\_clock\_dir);

remove\_proc\_entry(USER\_ROOT\_DIR,NULL);

printk(KERN\_INFO"%s %s has removed.\n",MODULE\_NAME,MODULE\_VERSION);

}

static ssize\_t proc\_read\_clock(struct file \*file,char \_\_user \*pszPage,size\_t size,loff\_t \*off)

{

int len = 0;

struct timeval xtime;

do\_gettimeofday(&xtime);

len = sprintf(pszPage,"%ld %ld\n",xtime.tv\_sec,xtime.tv\_usec);

printk("clock: read\_func()\n");

if (len <= \*off)

{

return 0;

}

len -= \*off;

\*off += len;

return len;

}

module\_init(init\_clock);

module\_exit(cleanup\_clock);

MODULE\_DESCRIPTION("clock module for gettimeofday of proc.");

MODULE\_LICENSE("GPL");

//EXPORT\_NO\_SYMBOLS;

（2）**编译内核模块Makefile文件**

Makefile文件

ifneq ($(KERNELRELEASE),)

#kbuild syntax.dependencyrelationshsip of files and target modules are listed here.

obj-m := proc\_clock.o

else

PWD := $(shell pwd)

KVER := $(shell uname -r)

KDIR := /lib/modules/$(KVER)/build

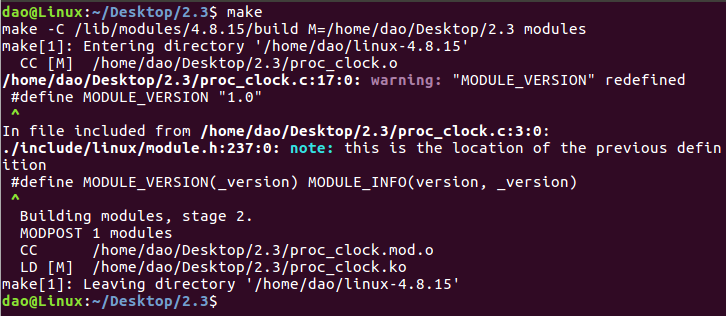
all:

$(MAKE) -C $(KDIR) M=$(PWD) modules

clean:

rm -rf \*.ko \*.o \*.mod.o \*.mod.c \*.symvers

endif

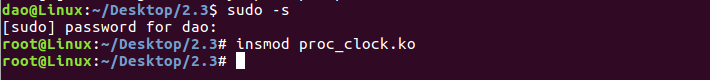


编译完成之后生成proc\_clock.ko模块文件。

（4）**加载内核模块**

输入命令sudo -s转为root用户，在系统root用户下运行用户态模块命令装载内核模块

#insmod proc\_clock.ko



（5）**测试**

测试源代码gettime.c

#include <stdio.h>

#include <sys/time.h>

#include <fcntl.h>

int main(void)

{

struct timeval getSystemTime;

char procClockTime[256];

int infile,len;

gettimeofday(&getSystemTime,NULL);

infile = open("/proc/clock/my\_clock",O\_RDONLY);

if (infile < 0)

{

printf("error!");

return 0;

}

len = read(infile,procClockTime,256);

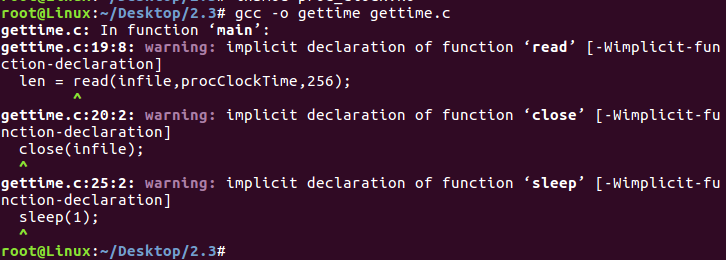
close(infile);

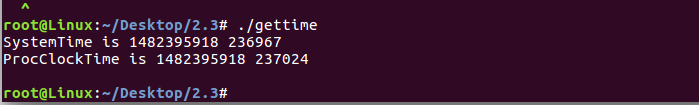
procClockTime[len] = '\0';

printf("SystemTime is %ld %ld\nProcClockTime is %s\n",getSystemTime.tv\_sec ,getSystemTime.tv\_usec,procClockTime);

sleep(1);

}





使用系统调用输出的时间与直接用gettimeofday()函数输出的时间大致相同

（6）**卸载内核模块**

在系统root用户下运行用户态模块命令卸载内核模块

#rmmod proc\_clock.ko

E:\学习\作业\操作系统\课程实验\2.3\screencut\rmmod.png