به نام خدا

محمد جواد زندیه

9831032

آزمایش شماره 2

آزمایشگاه سیستم عامل

تمرین ۱:

مراحل بالا را دنبال کنید تا یک ماژول هسته را ایجاد، بارگذاری و بردارید.ضمن بررسی محتوی بافر سابقه هسته مطمئن شوید مراحل کار را به‌درستی انجام داده‌اید.

ابتدا header های مورد نیاز را نصب می کنیم :

```
javad@javad-virtual-machine:~$ sudo apt-get install linux-headers-generic
[sudo] password for javad:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  linux-headers-5.4.0-88 linux-headers-5.4.0-88-generic
The following NEW packages will be installed:
  linux-headers-5.4.0-88 linux-headers-5.4.0-88-generic linux-headers-generic
0 upgraded, 3 newly installed, 0 to remove and 47 not upgraded.
```

سپس کد برنامه را به زبان c می نویسیم همانطور که در دستور کار گفته شده است:

home > javad > Desktop > project2_OSLab > C KernelModule.c

```c
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>

/* this function is called when the module is loaded */
int simple_init(void){
    printk(KERN_INFO "Loading Module\n");
    return 0;
}

/* this function is called when the module is removed */
void simple_exit(void){
    printk(KERN_INFO "Removing Module\n");
}

/* Macros for registering module entry and exit points. */
module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple module");
MODULE_AUTHOR("SGG");

```

سپس نیاز است برای کامپایل کردن کد خود، یک Makefile بنویسیم:

```
home > javad > Desktop > project2_OSLab > M Makefile
1  obj-m += KernelModule.o
2
3  all:
4      make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
5
6  clean:
7      make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

سپس در ترمینال با دستور make می توان کد خود را کامپایل کرده و فایل مورد نظر یعنی فایل دارای پسوند ko. را تولید کرد:

```
javad@javad-virtual-machine:~/Desktop/project2_OSLab$ ls
KernelModule.c  Makefile
javad@javad-virtual-machine:~/Desktop/project2_OSLab$ make
make -C /lib/modules/5.11.0-37-generic/build M=/home/javad/Desktop/project2_OSLab modules
make[1]: Entering directory '/usr/src/linux-headers-5.11.0-37-generic'
  CC [M]  /home/javad/Desktop/project2_OSLab/KernelModule.o
  MODPOST /home/javad/Desktop/project2_OSLab/Module.symvers
  CC [M]  /home/javad/Desktop/project2_OSLab/KernelModule.mod.o
  LD [M]  /home/javad/Desktop/project2_OSLab/KernelModule.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.11.0-37-generic'
```

پس از تولید این فایل، شروع به load کردن آن روی kernel می کنیم :

```
javad@javad-virtual-machine:~/Desktop/project2_OSLab$ sudo insmod KernelModule.ko
javad@javad-virtual-machine:~/Desktop/project2_OSLab$ lsmod
Module                  Size  Used by
KernelModule           16384  0
nls_utf8               16384  1
isofs                  49152  1
xt_CHECKSUM            16384  1
xt_MASQUERADE          20480  3
xt_conntrack           16384  1
ipt_REJECT             16384  2
```

پس از insert کردن ماژول ای که نوشتیم با دستور lsmod می توان از ماژول هایی که در هسته وجود دارند مطلع شد و می توان دید که ماژول ای که نوشتیم در این لیست قرار دارد.

با دستور dmesg می توان پیغامی که هنگام load شدن این ماژول روی بافر هسته نوشته شده است را دید:

```
javad@javad-virtual-machine:~/Desktop/project2_OSLab$ dmesg
[ 5193.293194] Loading Module
```

حال میخواهیم ماژول را از هسته برداری مسا اصطلاحا unload کنیم:

```
javad@javad-virtual-machine:~/Desktop/project2_OSLab$ sudo rmmod KernelModule
javad@javad-virtual-machine:~/Desktop/project2_OSLab$ dmesg
[ 5193.293194] Loading Module
[ 5235.752821] Removing Module
```

پس از unload کردن هم میتوان دید که پیغام Removing Module را در بافر هسته قرار داده است.

*نکته: برای پاک کردن بافر هسته میتوان از دستور sudo dmesg -c استفاده کرد و همانطور که معلوم است پیش از لود کردن این ماژول من این کار را کرده ام چون پیغام های قبلی در بافر نمایش داده نشده.

تمرین ۲:

در نقطه ورود ماژول، یک لیست پیوندی شامل پنج عنصر struct birthday ایجاد کنید. لیست پیوندی را پیمایش کنید و محتوای آن را به بافر سابقه هسته انتقال دهید. فرمان dmesg را احضار کنید تا مطمئن شوید که به محض بار شدن ماژول هسته، لیست به درستی ایجاد می‌شود.

در نقطه خروج ماژول، عناصر لیست را از لیست پیوندی حذف کرده و دوباره حافظه آزاد شده را به هسته برگردانید. باز هم فرمان dmesg را احضار کنید تا بررسی کنید به محض برداشتن ماژول هسته، لیست حذف می‌شود.

دقیقا تمام مراحل را برای تمرین دوم هم انجام میدهیم :

```
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$ dmesg
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$ ls
LinkedList_KernelModule.c  Makefile
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$ make
make -C /lib/modules/5.11.0-37-generic/build M=/home/javad/Desktop/project2_OSLab/LinkedList_KernelModule modules
make[1]: Entering directory '/usr/src/linux-headers-5.11.0-37-generic'
  CC [M]  /home/javad/Desktop/project2_OSLab/LinkedList_KernelModule/LinkedList_KernelModule.o
  MODPOST /home/javad/Desktop/project2_OSLab/LinkedList_KernelModule/Module.symvers
  CC [M]  /home/javad/Desktop/project2_OSLab/LinkedList_KernelModule/LinkedList_KernelModule.mod.o
  LD [M]  /home/javad/Desktop/project2_OSLab/LinkedList_KernelModule/LinkedList_KernelModule.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.11.0-37-generic'
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$ sudo insmod LinkedList_KernelModule.ko
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$ lsmod
Module                  Size  Used by
LinkedList_KernelModule    16384  0
nls_utf8               16384  1
isofs                  49152  1
xt_CHECKSUM            16384  1
```

```
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$ dmesg
[  459.705493] LinkedList_KernelModule: loading out-of-tree module taints kernel.
[  459.705570] LinkedList_KernelModule: module verification failed: signature and/or required key missing - tainting kernel
[  459.705707] day 1 ,month 2 ,year 2000
[  459.705709] day 2 ,month 3 ,year 2001
[  459.705710] day 3 ,month 4 ,year 2002
[  459.705711] day 4 ,month 5 ,year 2003
[  459.705711] day 5 ,month 6 ,year 2004
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$ sudo rmmod LinkedList_KernelModule
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$ dmesg
[  459.705493] LinkedList_KernelModule: loading out-of-tree module taints kernel.
[  459.705570] LinkedList_KernelModule: module verification failed: signature and/or required key missing - tainting kernel
[  459.705707] day 1 ,month 2 ,year 2000
[  459.705709] day 2 ,month 3 ,year 2001
[  459.705710] day 3 ,month 4 ,year 2002
[  459.705711] day 4 ,month 5 ,year 2003
[  459.705711] day 5 ,month 6 ,year 2004
[  493.439688] Finished removing linked list
javad@javad-virtual-machine:~/Desktop/project2_OSLab/LinkedList_KernelModule$
```

برای اطمینان از unload شدن ماژول در انتهای پیغام Finished removing linked list را قرار دادم که در اینجا هم میتوان دید که این پیغام چاپ شده است و پاک کردن داده ها به درستی انجام شده است و حافظه به باقر هسته برگردانده شده است.

کد برنامه :

توضیحات هر بخش از کد به صورت کامنت قرار داده شده است.

```c
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/list.h>
#include <linux/types.h>
#include <linux/slab.h>

/* implementation of list_head struct in types.h file :
 struct list_head {
     struct list_head *next, *prev;
 }; */

struct birthday{
    int day;
    int month;
    int year;

    /*list_head struct, put linked list among nodes of linked list
    list has prev and next pointer
    linux kernel list implementation*/
    struct list_head list;
};

/*this Macro makes birthday_list object as a pointer to first node of linked list
birthday_list type is struct list_head*/
static LIST_HEAD(birthday_list_head);

/* this function is called when the module is loaded */
int my_init(void){
    struct birthday *person1, *person2, *person3, *person4, *person5, *ptr1;
    /*kmalloc for kernel memory allocation
    GFP_KERNEL flag is for general purpose allocation of kernel memory*/
    person1 = kmalloc(sizeof(struct birthday), GFP_KERNEL);
    person1->day = 1;
    person1->month = 2;
    person1->year = 2000;
    /*this Macro initialize list struct of birthday struct*/
    INIT_LIST_HEAD(&person1->list);
    /*this Macro add person1 into linked list tail*/
    list_add_tail(&person1->list, &birthday_list_head);

    person2 = kmalloc(sizeof(struct birthday), GFP_KERNEL);
    person2->day = 2;
    person2->month = 3;
    person2->year = 2001;
    INIT_LIST_HEAD(&person2->list);
    list_add_tail(&person2->list, &birthday_list_head);

    person3 = kmalloc(sizeof(struct birthday), GFP_KERNEL);
    person3->day = 3;
    person3->month = 4;
    person3->year = 2002;
    INIT_LIST_HEAD(&person3->list);
    list_add_tail(&person3->list, &birthday_list_head);
```

```c
    person4 = kmalloc(sizeof(struct birthday), GFP_KERNEL);
    person4->day = 4;
    person4->month = 5;
    person4->year = 2003;
    INIT_LIST_HEAD(&person4->list);
    list_add_tail(&person4->list, &birthday_list_head);

    person5 = kmalloc(sizeof(struct birthday), GFP_KERNEL);
    person5->day = 5;
    person5->month = 6;
    person5->year = 2004;
    INIT_LIST_HEAD(&person5->list);
    list_add_tail(&person5->list, &birthday_list_head);

    /*this Macro is uses for linked list iteration
    ptr : on each iteration ptr points to the next birthday struct
    &birthday_list_head : pointer to list head of our linked list
    list : name of struct list_head variable in birthday struct*/
    list_for_each_entry(ptrl, &birthday_list_head, list){
        printk(KERN_INFO "day %d ,month %d ,year %d\n", ptrl->day, ptrl->month, ptrl->year);
    }
    return 0;
}
```

```c
/* this function is called when the module is removed */
void my_exit(void){
    /*Go through the list and free the memory.*/
    struct birthday *ptr2, *temp;
    list_for_each_entry_safe(ptr2, temp, &birthday_list_head, list) {
        /*delete nodes of linked list*/
        list_del(&ptr2->list);
        /*free the memory which is allocated using kmalloc*/
        kfree(ptr2);
    }
    printk(KERN_INFO "Finished removing linked list\n");
}

/* Macros for registering module entry and exit points. */
module_init(my_init);
module_exit(my_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("LinkedList module");
MODULE_AUTHOR("MJZ");
```

:Makefile

```
obj-m += LinkedList_KernelModule.o

KDIR = /lib/modules/$(shell uname -r)/build

all:
	make -C $(KDIR) M=$(shell pwd) modules

clean:
	make -C $(KDIR) M=$(shell pwd) clean
```