دستور کار آزمایشگاه سیستم های عامل

آزمایشگاه سیستم های عامل

آرین محسنی

آبان 1403

دستور کار آزمایشگاه سیستم های عامل

فهرست

دستور کار آزمایشگاه سیستم های عامل

تمرین 1

الف) یک ماژول بنویسید که بتوانید به آن پارامترهایی از نوع short, int, long, string, array به عنوان ورودی بدهید و از آن در کد استفاده کنید (راهنمایی: می‌توانید درباره module_param و MODULE_PARM_DES جستجو کنید).

برای این تمرین یک کد در زبان c مینویسیم که هنگام ورود به هسته عملیات چاپ کردن پارامتر های داده شده را انجام دهد.

در بخش اول کد, تعریف پارامتر هایی است که قرار است هنگام لود کردن ماژول به آن پاس بدهیم.

module_hw.c 5 | Makefile

module_hw.c > MODULE_PARM_DESC

```c
#include <linux/init.h>       // For module_init and module_exit macros
#include <linux/module.h>     // For module-related macros
#include <linux/kernel.h>     // For printk
#include <linux/moduleparam.h> // For module parameters

// Define arrays and variables to hold the parameters
static char *stringArray[3];  // An array of strings with a maximum of 3 elements
static long longArray[3];     // An array of longs
static int intArray[3];       // An array of integers
static short shortArray[3];   // An array of shorts

// Define the number of elements in the arrays
static int stringCount = 0;
static int longCount = 0;
static int intCount = 0;
static int shortCount = 0;

// Declare module parameters and descriptions
module_param_array(stringArray, charp, &stringCount, 0660);
MODULE_PARM_DESC(stringArray, "An array of strings");

module_param_array(longArray, long, &longCount, 0660);
MODULE_PARM_DESC(longArray, "An array of longs");

module_param_array(intArray, int, &intCount, 0660);
MODULE_PARM_DESC(intArray, "An array of integers");

module_param_array(shortArray, short, &shortCount, 0660);
MODULE_PARM_DESC(shortArray, "An array of shorts");

```

تابع module_param_array یکی از توابع مهم در برنامه‌نویسی ماژول‌های کرنل لینوکس است که برای تعریف پارامترهایی که می‌توان آن‌ها را در زمان بارگذاری ماژول

مثلاً هنگام استفاده از insmod تعیین کرد، استفاده می‌شود. این تابع به طور خاص برای استفاده از آرایه‌ها به عنوان پارامتر ورودی کاربرد دارد.

StringArray
این اولین پارامتر نام آرایه‌ای است که به عنوان پارامتر ورودی به ماژول داده می‌شود. در این مورد، آرایه‌ای از رشته‌ها (از نوع charpکه نشان‌دهنده‌ی اشاره‌گر به رشته کاراکتری است) استفاده می‌شود.

Charp
این نوع داده‌ای است که آرایه ذخیره می‌کند. در این مثال، charp یعنی آرایه‌ای از رشته‌های کاراکتری.

&StringCount
این یک اشاره‌گر به متغیری است که تعداد عناصر آرایه ورودی را ذخیره می‌کند. کرنل لینوکس این متغیر را به‌روزرسانی می‌کند و تعداد عناصر موجود در آرایه را به ما می‌دهد.

0660
این مقادیر تعیین‌کننده سطوح دسترسی به پارامتر هستند. در اینجا، سطح دسترسی به‌گونه‌ای است که صاحب فایل و گروه آن می‌توانند بخوانند و بنویسند، اما دیگران نمی‌توانند به آن دسترسی داشته باشند.

در بخش بعدی کد تابعی که هنگام لود شدن ماژول صدا زده میشود را پیاده سازی کرده ایم که در آن اطلاعات ارایه هایی که در ابتدا پاس داده بودیم چاپ میشود.

در بخش بعدی کد تابعی که هنگام لود شدن ماژول صدا زده میشود را پیاده سازی کرده ایم که در آن اطلاعات ارایه هایی که در ابتدا پاس داده بودیم چاپ میشود.

در بخش بعدی کد تابعی که هنگام لود شدن ماژول صدا زده میشود را پیاده سازی کرده ایم که در آن اطلاعات ارایه هایی که در ابتدا پاس داده بودیم چاپ میشود.

در بخش بعدی کد تابعی که هنگام لود شدن ماژول صدا زده میشود را پیاده سازی کرده ایم که در آن اطلاعات ارایه هایی که در ابتدا پاس داده بودیم چاپ میشود.

دستور کار آزمایشگاه سیستم های عامل

در بخش بعدی کد تابعی که هنگام لود شدن ماژول صدا زده میشود را پیاده سازی کرده ایم که در آن اطلاعات ارایه هایی که در ابتدا پاس داده بودیم چاپ میشود.

```c
static int __init simple_module_init(void)
{
    int i;

    // Print the passed string array
    pr_info("Module loaded with the following string array parameters:\n");
    for (i = 0; i < stringCount; i++) {
        pr_info("stringArray[%d] = %s\n", i, stringArray[i]);
    }

    // Print the passed long array
    pr_info("Module loaded with the following long array parameters:\n");
    for (i = 0; i < longCount; i++) {
        pr_info("longArray[%d] = %ld\n", i, longArray[i]);
    }

    // Print the passed int array
    pr_info("Module loaded with the following int array parameters:\n");
    for (i = 0; i < intCount; i++) {
        pr_info("intArray[%d] = %d\n", i, intArray[i]);
    }

    // Print the passed short array
    pr_info("Module loaded with the following short array parameters:\n");
    for (i = 0; i < shortCount; i++) {
        pr_info("shortArray[%d] = %hd\n", i, shortArray[i]);
    }

    return 0;
}
```

و در آخر هم لایسنس های مورد نیاز و تابع هنگام خارج شدن از هسته

```c
// Function to execute when the module is removed
static void __exit simple_module_exit(void)
{
    pr_info("Removing module\n");
}

// Register the init and exit functions
module_init(simple_module_init);
module_exit(simple_module_exit);

// Module metadata
MODULE_LICENSE("GPL"); //GPL = GNU General Public License
MODULE_DESCRIPTION("simple module");
MODULE_AUTHOR("OS-Lab-Group");
```

دستور کار آزمایشگاه سیستم های عامل

برای اجرا این کد باید یک فایل Makefile بسازیم که به این شکل است.

```
Makefile
obj-m += module_hw.o

PWD := $(CURDIR)

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

all:
    $(MAKE) -C $(KERNEL_DIR) M=$(PWD) modules

clean:
    $(MAKE) -C $(KERNEL_DIR) M=$(PWD) clean

```

سپس در ترمینال make میکنیم و بعد دستور زیر را اجرا میکنیم:

```
sudo insmod module_hw.ko stringArray="str1,str2,str3" longArray="123,456,789" intArray="1,2,3" shortArray="100,200,300"
```

در این دستور علاوه بر لود کردن ماژول بر روی هسته, به ارایه ها ورودی تستی نیز می دهد تا در ادامه آن را چاپ کنیم.

با دستور sudo dmesg | tail اتفاقات رخ داده بعد از لود شدن را مشاهده میکنیم.

```
arian@arian-VirtualBox:~/Desktop/University/OS/OS_Lab/Lab3/HW$ sudo dmesg | tail
[  884.375610] longArray[1] = 456
[  884.375611] longArray[2] = 789
[  884.375611] Module loaded with the following int array parameters:
[  884.375612] intArray[0] = 1
[  884.375612] intArray[1] = 2
[  884.375613] intArray[2] = 3
[  884.375614] Module loaded with the following short array parameters:
[  884.375614] shortArray[0] = 100
[  884.375615] shortArray[1] = 200
[  884.375616] shortArray[2] = 300
```

همانطور که مشخص است ورودی های ما را ذخیره و سپس نمایش می دهد.

در آخر از هسته با دستورsudo rmmod module_hw خارج میشویم.

```
[ 2721.580135] Removing module
arian@arian-VirtualBox:~/Desktop/University/OS/OS_Lab/Lab3/HW$
```

دستور کار آزمایشگاه سیستم های عامل

تمرین 2

در این بخش باید یک struct با محتویات زیر تعریف کنیم و سپس آن را مقدار دهی , پیمایش و پاک کنیم.

part2 > C birthday_module.c > birthday > month

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

/* Define the structure for birthday */
struct birthday {
    int day;
    int month;
    int year;
    struct list_head list; // Kernel's list structure
};

/* Initialize the head of the linked list */
static LIST_HEAD(birthday_list);
```

بخشی که موقع لود شدن اجرا میشود و مقدار دهی انجام میشود.

```c
int simple_init(void) {
    struct birthday *person;
    int i;

    printk(KERN_INFO "Loading Module\n");

    /* Create 5 birthday instances and add them to the linked list */
    for (i = 1; i <= 5; i++) {
        person = kmalloc(sizeof(*person), GFP_KERNEL); // Allocate memory
        if (!person) {
            printk(KERN_ALERT "Memory allocation failed\n");
            return -ENOMEM;
        }
        person->day = i;
        person->month = i + 1;
        person->year = 1990 + i;
        INIT_LIST_HEAD(&person->list);  // Initialize list structure
        list_add_tail(&person->list, &birthday_list);  // Add to list

        printk(KERN_INFO "Added person: %d/%d/%d\n", person->day, person->month, person->year);
    }

    /* Traverse and print the list */
    printk(KERN_INFO "Traversing the birthday list\n");
    list_for_each_entry(person, &birthday_list, list) {
        printk(KERN_INFO "Birthday: %d/%d/%d\n", person->day, person->month, person->year);
    }

    return 0;
}
```

و در نهایت این تابع هنگام خروج از هسته اجرا شده و بخش هایی مثل برعکس کردن لیست تولد ها و چاپ کردن آن ها و آزاد کردن فضای اختصاص داده شده را در بر دارد.

```c
/* This function is called when the module is removed */
void simple_exit(void) {
    struct birthday *person, *next;

    printk(KERN_INFO "Removing Module\n");

    /* Reverse traverse the list and print */
    printk(KERN_INFO "Reversing the birthday list\n");
    list_for_each_entry_reverse(person, &birthday_list, list) {
        printk(KERN_INFO "Birthday: %d/%d/%d\n", person->day, person->month, person->year);
    }

    /* Delete the list and free memory */
    list_for_each_entry_safe(person, next, &birthday_list, list) {
        printk(KERN_INFO "Deleting person: %d/%d/%d\n", person->day, person->month, person->year);
        list_del(&person->list);  // Remove from list
        kfree(person);  // Free memory
    }
}

/* Registering the init and exit functions */
module_init(simple_init);
module_exit(simple_exit);

/* License and description information */
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("A simple birthday linked list module");
MODULE_AUTHOR("OS-Lab-Group");
```

یک Makefile میسازیم و اجرا میکنیم

ماژول را لود میکنیم

```
arian@arian-VirtualBox:~/Desktop/University/OS/OS_Lab/Lab3/HW/part2$ sudo insmod birthday_module.ko
arian@arian-VirtualBox:~/Desktop/University/OS/OS_Lab/Lab3/HW/part2$ sudo dmesg | tail
[ 4148.836005] Added person: 2/3/1992
[ 4148.836006] Added person: 3/4/1993
[ 4148.836007] Added person: 4/5/1994
[ 4148.836007] Added person: 5/6/1995
[ 4148.836008] Traversing the birthday list
[ 4148.836009] Birthday: 1/2/1991
[ 4148.836009] Birthday: 2/3/1992
[ 4148.836010] Birthday: 3/4/1993
[ 4148.836011] Birthday: 4/5/1994
[ 4148.836011] Birthday: 5/6/1995
```

دستور کار آزمایشگاه سیستم های عامل

و در نهایت از هسته خارج میشویم.

```
arian@arian-VirtualBox:~/Desktop/University/OS/OS_Lab/Lab3/HW/part2$ sudo rmmod birthday_module
[sudo] password for arian:
arian@arian-VirtualBox:~/Desktop/University/OS/OS_Lab/Lab3/HW/part2$ sudo dmesg | tail
[ 5662.542361] Birthday: 5/6/1995
[ 5662.542362] Birthday: 4/5/1994
[ 5662.542363] Birthday: 3/4/1993
[ 5662.542364] Birthday: 2/3/1992
[ 5662.542365] Birthday: 1/2/1991
[ 5662.542366] Deleting person: 1/2/1991
[ 5662.542367] Deleting person: 2/3/1992
[ 5662.542368] Deleting person: 3/4/1993
[ 5662.542369] Deleting person: 4/5/1994
[ 5662.542369] Deleting person: 5/6/1995
```