

دستور کار شماره ۳ os

1) ابتدا از نصب بودن make و gcc روی سیستم مطمئن شوید

`sudo apt install make`

`sudo apt install gcc`

از دستور ات بالا میتوانید برای نصب انها استفاده کنید

2) تکه کد `simpe_module.c` را آماده کنید (هر اسمی که میخواهید باشد اما پسوند ان باید `.c` باشد)

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/printk.h>

int simple_init(void){
    pr_info("Loading module...\n");
    return 0;
}

void simple_exit(void){
    pr_info("Removing module.\n");
}

module_init(simple_init);
module_exit(simple_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("simple module");
MODULE_AUTHOR("OS-LAB-Group");
```

پس از ان نیاز داریم تا تکه کد را کامپایل کنیم به این منظور از Makefile استفاده میکنیم

فایلی با نام Makefile ساخته (بدون پسوند) و کد زیر را در ان قرار میدهیم(مطمئن شوید که ایندنت ها درست رعایت شده در غیر این صورت هنگام `make` ارور میگیرید)

```

simple_module.o += obj-m
CURDIR))$ =:PWD
:all
build M=$(PWD) modules/(r- shell uname)$ /lib/modules/ C- make
:clean
build M=$(PWD) clean/(r- shell uname)$ /lib/modules/ C- make

```

اگر فایل c. را با نامی به غیر از simple\_module استفاده کرده اید از آن به در خط اول استفاده کنید

اگر بدون مشکل اجرا شود باید خروجی مشابه با این داشته باشید و فایلی با اسم فایل شما و پسوند .ko ساخته شود

```

mem@mem:~/0s/simple_module$ make
make -C /lib/modules/4.15.0-213-generic/build M=/home/mem/0s/simple_module modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-213-generic'
  CC [M] /home/mem/0s/simple_module/simple_module.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC /home/mem/0s/simple_module/simple_module.mod.o
  LD [M] /home/mem/0s/simple_module/simple_module.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-213-generic'
mem@mem:~/0s/simple_module$ ls
Makefile      Module.symvers  simple_module.ko  simple_module.mod.o
modules.order  simple_module.c  simple_module.mod.c  simple_module.o
mem@mem:~/0s/simple_module$

```

حال شما میتوانید تکه کدتان را در kernel اجرا کنید

اجرا کردن: module:

```
sudo insmod simple_module.ko
```

توقف و خارج شدن از: module:

```
sudo rmmod simple_module
```

برای مشاهده log های تولید

ی میتوانید از دستور `dmesg | tail` استفاده کنید

```

[3000] password for mem:
mem@mem:~/0s/simple_module$ dmesg | tail
[  5.773262] IPv6: ADDRCONF(NETDEV_UP): enp0s3: link is not ready
[  5.780469] e1000: enp0s3 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[  5.780764] IPv6: ADDRCONF(NETDEV_CHANGE): enp0s3: link becomes ready
[  8.370988] new mount options do not match the existing superblock, will be ignored
[ 13.955797] systemd-journald[439]: File /var/log/journal/75999007b78d4d82a0658be3c4684229/user-1000.journal corrupt
ed or uncleanly shut down, renaming and replacing.
[ 1190.540256] simple_module: loading out-of-tree module taints kernel.
[ 1190.540312] simple_module: module verification failed: signature and/or required key missing - tainting kernel
[ 1190.540572] Loading module...
[ 1356.218145] Removing module.
[ 3102.576761] Loading module...
mem@mem:~/0s/simple_module$

```

برای قسمت دوم آزمایش هم همین کار را باید انجام داد فقط تکه کد C. تغییر کرده است

تکه کد C.

```

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

struct birthday {
    int day;
    int month;
    int year;
    struct list_head list;
};

LIST_HEAD(my_list);

int my_module_init(void) {
    struct birthday *b1;

    b1 = kmalloc(sizeof(struct birthday), GFP_KERNEL);
    if (!b1) {
        return -ENOMEM;
    }

    b1->day = 2;
    b1->month = 8;
    b1->year = 1995;

    INIT_LIST_HEAD(&b1->list);
    list_add_tail(&b1->list, &my_list);
    pr_info("Birthday added: %d/%d/%d\n", b1->day, b1->month, b1->year);
    return 0;
}

```

```

void my_module_exit(void) {
    struct birthday *b1;
    struct list_head *pos, *q;

    // Iterate through the list and free allocated memory
    list_for_each_safe(pos, q, &my_list) {
        b1 = list_entry(pos, struct birthday, list);
        pr_info("Removing birthday: %d/%d/%d\n", b1->day, b1->month, b1->year);
        list_del(pos);
        kfree(b1);
    }
}

module_init(my_module_init);
module_exit(my_module_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Birthday List Module");
MODULE_AUTHOR("OS-LAB-Group");

```

تمرین ها

سوال یک)

هر دو تا شو انجام دادم

۱- دادن پارامتر های ورودی

```

#include <linux/init.h>
#include <linux/module.h>
#include <linux/printk.h>
#include <linux/slab.h>
#include <linux/moduleparam.h>

// Parameters that will be passed during module insertion
static char *my_string = "default_string";

```

```

static long my_long = 100;
static int my_int = 42;
static short my_short = 12;

module_param(my_string, charp, 0000);
module_param(my_long, long, 0000);
module_param(my_int, int, 0000);
module_param(my_short, short, 0000);

// Descriptions for the parameters
MODULE_PARM_DESC(my_string, "A string parameter");
MODULE_PARM_DESC(my_long, "A long integer parameter");
MODULE_PARM_DESC(my_int, "An integer parameter");
MODULE_PARM_DESC(my_short, "A short integer parameter");

int my_module_init(void) {
    pr_info("printing params....");
    pr_info("Module parameters: string=%s, long=%ld, int=%d, short=%d\n", my_string, my_long, my_int, my_short);

    return 0;
}

void my_module_exit(void) {
    pr_info("exiting from your module...");
}

module_init(my_module_init);
module_exit(my_module_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Birthday List Module with Parameters");
MODULE_AUTHOR("OS-LAB-Group");

```

خروجی

```

make[1]: Leaving directory '/usr/src/linux-headers-4.19.0-219-generic
mem@mem:~/OperatingSystemsAssignments/3_kernel_modules/param_module$ sudo insmod module_with_param.ko my_string="AmirH
ossein" my_long=20 my_int=1000 my_short=1

mem@mem:~/OperatingSystemsAssignments/3_kernel_modules/param_module$ sudo rmmod module_with_param
mem@mem:~/OperatingSystemsAssignments/3_kernel_modules/param_module$ dmesg | tail

[ 564.367568] Deleted successfully
[ 564.367569] Deleted successfully
[ 1705.741920] Person ID: 1, First Name: AmirHossein Motaghian
[ 1718.502171] Module unloaded and memory freed
[ 3258.110725] printing params....
[ 3258.110728] Module parameters: string=AmirHossein, long=20, int=1000, short=1
[ 3266.206020] exiting from your module...
[ 3323.902853] printing params....
[ 3323.902855] Module parameters: string=AmirHossein, long=20, int=1000, short=1
[ 3327.406457] exiting from your module...
mem@mem:~/OperatingSystemsAssignments/3_kernel_modules/param_module$

```

اطلاعات مربوط به process

```

#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/sched/signal.h>
#include <linux/mm.h>

int my_module_init(void) {
    struct task_struct *task;
    struct mm_struct *mm;

    pr_info("Loading process info...");

    for_each_process(task) {
        mm = task->mm;
        pr_info("Process: PID=%d | Name=%s\n", task->pid, task->comm);
        if (mm) {
            pr_info("Memory Usage (RSS): %lu KB\n", get_mm_rss(mm) * PAGE_SIZE / 1024);
        }

        pr_info(" CPU Time: User Mode=%llu | Kernel Mode=%llu\n", task->utime, task->stime);
    }

    pr_info("All Data of process shared.\n");
    return 0;
}

void my_module_exit(void) {
    pr_info("Process Info Module Exiting\n");
}

module_init(my_module_init);

```

```
module_exit(my_module_exit);
```

```
MODULE_LICENSE("GPL");
```

```
MODULE_DESCRIPTION("A Module to Display Process Information");
```

```
MODULE_AUTHOR("OS-LAB-Group");
```

```
[ 3550.940724] Loading process info...
[ 3550.940727] Process: PID=1 | Name=systemd
[ 3550.940730] Memory Usage (RSS): 8716 KB
[ 3550.940731] CPU Time: User Mode=108000000 | Kernel Mode=1432000000
[ 3550.940733] Process: PID=2 | Name=kthreadd
[ 3550.940735] CPU Time: User Mode=0 | Kernel Mode=80000000
[ 3550.940736] Process: PID=4 | Name=kworker/0:0H
[ 3550.940738] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.940739] Process: PID=5 | Name=kworker/u4:0
[ 3550.940741] CPU Time: User Mode=0 | Kernel Mode=1360000000
[ 3550.940743] Process: PID=6 | Name=mm_percpu_wq
[ 3550.940744] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.940745] Process: PID=7 | Name=ksoftirqd/0
[ 3550.940747] CPU Time: User Mode=0 | Kernel Mode=480000000
[ 3550.940748] Process: PID=8 | Name=rcu_sched
[ 3550.940750] CPU Time: User Mode=0 | Kernel Mode=1960000000
[ 3550.940751] Process: PID=9 | Name=rcu_bh
[ 3550.940753] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.940754] Process: PID=10 | Name=migration/0
[ 3550.940756] CPU Time: User Mode=0 | Kernel Mode=120000000
[ 3550.940758] Process: PID=11 | Name=watchdog/0
[ 3550.940775] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.940777] Process: PID=12 | Name=cpuhp/0
[ 3550.940778] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.940779] Process: PID=13 | Name=cpuhp/1
[ 3550.940781] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.940782] Process: PID=14 | Name=watchdog/1
[ 3550.940783] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.940785] Process: PID=15 | Name=migration/1
[ 3550.940786] CPU Time: User Mode=0 | Kernel Mode=120000000
[ 3550.940788] Process: PID=16 | Name=ksoftirqd/1
[ 3550.940790] CPU Time: User Mode=0 | Kernel Mode=80000000
[ 3550.940791] Process: PID=18 | Name=kworker/1:0H
[ 3550.940792] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.940794] Process: PID=19 | Name=kdevtmpfs
```



```

[ 3550.941035] Process: PID=1447 | Name=systemd
[ 3550.941036] Memory Usage (RSS): 7668 KB
[ 3550.941037] CPU Time: User Mode=160000000 | Kernel Mode=320000000
[ 3550.941039] Process: PID=1449 | Name=(sd-pam)
[ 3550.941040] Memory Usage (RSS): 2472 KB
[ 3550.941042] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.941043] Process: PID=1581 | Name=sshd
[ 3550.941045] Memory Usage (RSS): 4492 KB
[ 3550.941046] CPU Time: User Mode=1440000000 | Kernel Mode=5680000000
[ 3550.941047] Process: PID=1582 | Name=bash
[ 3550.941049] Memory Usage (RSS): 5576 KB
[ 3550.941050] CPU Time: User Mode=2960000000 | Kernel Mode=2720000000
[ 3550.941052] Process: PID=1607 | Name=nano
[ 3550.941053] Memory Usage (RSS): 6016 KB
[ 3550.941054] CPU Time: User Mode=880000000 | Kernel Mode=280000000
[ 3550.941055] Process: PID=2738 | Name=kworker/0:0
[ 3550.941057] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.941059] Process: PID=4750 | Name=kworker/u4:2
[ 3550.941060] CPU Time: User Mode=0 | Kernel Mode=1080000000
[ 3550.941061] Process: PID=7381 | Name=kworker/1:0
[ 3550.941063] CPU Time: User Mode=0 | Kernel Mode=0
[ 3550.941064] Process: PID=8691 | Name=sudo
[ 3550.941065] Memory Usage (RSS): 4092 KB
[ 3550.941066] CPU Time: User Mode=0 | Kernel Mode=80000000
[ 3550.941068] Process: PID=8692 | Name=insmod
[ 3550.941069] Memory Usage (RSS): 848 KB
[ 3550.941070] CPU Time: User Mode=0 | Kernel Mode=40000000
[ 3550.941071] All Data of process shared.
[ 3557.310979] Process Info Module Exiting

```

تمرین دوم

```

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

// Define the birthday structure
struct birthday {
    int day;
    int month;
    int year;
    struct list_head list;
};

// Initialize the list head
LIST_HEAD(my_list);

```



```

int my_module_init(void) {
    struct list_head *pos;
    struct birthday *b1;
    int i;

    // Create and add 5 birthday elements to the list
    for (i = 0; i < 5; i++) {
        struct birthday *new_birthday;
        new_birthday = kmalloc(sizeof(struct birthday), GFP_KERNEL);
        if (!new_birthday) {
            pr_err("Memory allocation failed\n");
            continue;
        }

        // Assign arbitrary date values (you can change them as needed)
        new_birthday->day = 2 + i;
        new_birthday->month = 8;
        new_birthday->year = 1995 + i;

        // Add each element to the list
        list_add_tail(&new_birthday->list, &my_list);
    }

    // Print the list (forward iteration)
    pr_info("Forward Iteration: \n");
    list_for_each(pos, &my_list) {
        b1 = list_entry(pos, struct birthday, list);
        pr_info("Birthday: %d/%d/%d\n", b1->day, b1->month, b1->year);
    }

    return 0;
}

void my_module_exit(void) {
    struct birthday *entry, *tmp;

    list_for_each_entry_safe(entry, tmp, &my_list, list) {
        list_del(&entry->list);
        kfree(entry); // Free the memory for the entry
        printk(KERN_INFO "Deleted successfully\n");
    }
}

module_init(my_module_init);
module_exit(my_module_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Birthday List Module");

```

```
MODULE_AUTHOR("OS-LAB-Group");
```

```
nem@mem:~/OperatingSystemsAssignments/3_kernel_modules/five_struct$ ^C
nem@mem:~/OperatingSystemsAssignments/3_kernel_modules/five_struct$ sudo insmod five_struct.ko
nem@mem:~/OperatingSystemsAssignments/3_kernel_modules/five_struct$ sudo rmmod five_struct
nem@mem:~/OperatingSystemsAssignments/3_kernel_modules/five_struct$ dmesg | tail
[ 3704.054022] Birthday: 2/8/1995
[ 3704.054023] Birthday: 3/8/1996
[ 3704.054024] Birthday: 4/8/1997
[ 3704.054025] Birthday: 5/8/1998
[ 3704.054027] Birthday: 6/8/1999
[ 3714.108032] Deleted successfully
[ 3714.108034] Deleted successfully
[ 3714.108035] Deleted successfully
[ 3714.108036] Deleted successfully
[ 3714.108037] Deleted successfully
```

امتیازی

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/slab.h>
#include <linux/hashtable.h>
#include <linux/string.h>

struct person {
    int id;
    char firstname[30];
    struct hlist_node node;
};

static DEFINE_HASHTABLE(person_table, 5);

static int __init my_module_init(void) {
    struct person *new_person;

    new_person = kmalloc(sizeof(struct person), GFP_KERNEL);
    if (!new_person) {
        printk(KERN_ERR "Memory allocation failed for person\n");
        return -ENOMEM;
    }

    new_person->id = 1;
    strncpy(new_person->firstname, "AmirHossein Motaghian", sizeof(new_person->firstname) - 1);
    new_person->firstname[sizeof(new_person->firstname) - 1] = '\0';

    hash_add(person_table, &new_person->node, new_person->id);

    struct person *entry;
```

```

hash_for_each_possible(person_table, entry, node, new_person->id) {
    if (entry->id == new_person->id) {
        printk(KERN_INFO "Person ID: %d, First Name: %s\n", entry->id, entry->firstname);
    }
}

return 0;
}

static void __exit my_module_exit(void) {
    struct person *entry;
    struct hlist_node *tmp;
    int bkt;

    hash_for_each_safe(person_table, bkt, tmp, entry, node) {
        hash_del(&entry->node);
        kfree(entry);
    }

    printk(KERN_INFO "Module unloaded and memory freed\n");
}

module_init(my_module_init);
module_exit(my_module_exit);
MODULE_LICENSE("GPL");

```

```

em@mem:~/OperatingSystemsAssignments/3_kernel_modules/hashtable$ sudo insmod hashtable.ko
em@mem:~/OperatingSystemsAssignments/3_kernel_modules/hashtable$ sudo rmmod hashtable
em@mem:~/OperatingSystemsAssignments/3_kernel_modules/hashtable$ dmesg | tail
3704.054024] Birthday: 4/8/1997
3704.054025] Birthday: 5/8/1998
3704.054027] Birthday: 6/8/1999
3714.108032] Deleted successfully
3714.108034] Deleted successfully
3714.108035] Deleted successfully
3714.108036] Deleted successfully
3714.108037] Deleted successfully
4411.233620] Person ID: 1, First Name: AmirHossein Motaghian
4416.282470] Module unloaded and memory freed
em@mem:~/OperatingSystemsAssignments/3_kernel_modules/hashtable$

```

دو خط آخر خروجی این ماژول است