

# گزارش دستورکار دوم آزمایشگاه سیستم‌های عامل

نگار موقتیان، ۹۸۳۱۰۶۲

## ۱. بارگذاری ماژول هسته سخت افزاری:

در این قسمت از آزمایش ماژول هسته nvmem-rave-sp-eeprom که یک EEPROM درایور برای Rave SP می‌باشد را بر روی هسته سیستم عامل بارگذاری کرده و سپس آن را حذف می‌کنیم.

برای این کار ابتدا فایل با پسوند ko. مربوط به این ماژول هسته را از سایت زیر دانلود کردم:

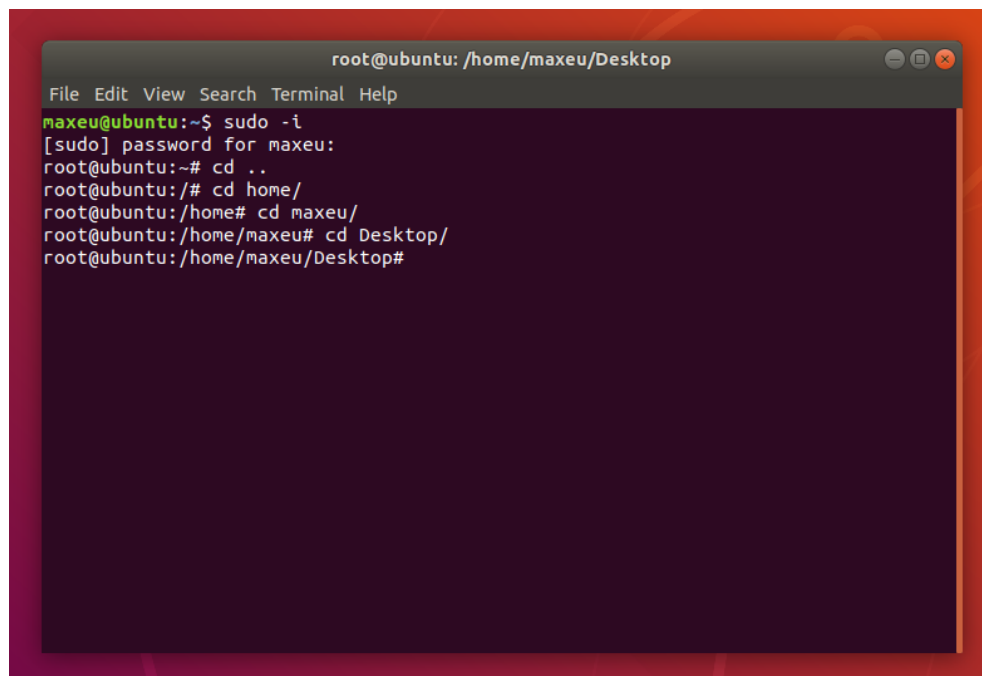
<https://www.candelatech.com/downloads/j/lib/modules/5.15.17+/kernel/drivers/nvmem/>

اما متأسفانه با وجود استفاده از نسخه 18.04 سیستم عامل Ubuntu با خطای Invalid module format مواجه شدم. لذا فایل C. این ماژول را از ریپازیتوری زیر دانلود کرده و آن را make کردم:

<https://github.com/torvalds/linux/blob/master/drivers/nvmem/rave-sp-eeprom.c>

و این بار در بارگذاری فایل خروجی ko. مرحله قبل بر روی هسته مشکلی وجود نداشت (طریقه make کردن فایل‌های زبان C در قسمت‌های بعدی آزمایش به طور کامل توضیح داده خواهد شد).

پس از بدست آوردن فایل ko. منطبق با هسته سیستم عامل مراحل زیر انجام شد.



```
root@ubuntu: /home/maxeu/Desktop
File Edit View Search Terminal Help
maxeu@ubuntu:~$ sudo -i
[sudo] password for maxeu:
root@ubuntu:~# cd ..
root@ubuntu:/# cd home/
root@ubuntu:/home# cd maxeu/
root@ubuntu:/home/maxeu# cd Desktop/
root@ubuntu:/home/maxeu/Desktop#
```

به دلیل این که می‌خواهیم ماژولی را بر روی هسته سیستم عامل بارگذاری کنیم نیاز به دسترسی root داریم. بنابراین در این مرحله با استفاده از دستور `sudo -i` سطح دسترسی را به root تغییر داده و به فولدر Desktop (که فایل `nvmem-rave-sp-EEPROM.ko` در آن قرار دارد) می‌رویم.

```
root@ubuntu: /home/maxeu/Desktop
File Edit View Search Terminal Help
root@ubuntu: /home/maxeu/Desktop# lsmod
Module                               Size  Used by
vmw_vsock_vmci_transport             32768  2
vsock                                36864  3 vmw_vsock_vmci_transport
intel_rapl_msr                       20480  0
intel_rapl_common                    24576  1 intel_rapl_msr
crct10dif_pclmul                     16384  1
crc32_pclmul                         16384  0
ghash_clmulni_intel                  16384  0
aesni_intel                          372736 0
crypto_simd                          16384  1 aesni_intel
cryptd                               24576  2 crypto_simd,ghash_clmulni_intel
glue_helper                          16384  1 aesni_intel
rapl                                 20480  0
vmw_balloon                          24576  0
input_leds                           16384  0
joydev                               28672  0
serio_raw                            20480  0
snd_ens1371                           28672  4
snd_ac97_codec                       135168 1 snd_ens1371
gameport                             16384  1 snd_ens1371
ac97_bus                             16384  1 snd_ac97_codec
snd_pcm                              102400 2 snd_ac97_codec,snd_ens1371
```

حال ابتدا با استفاده از دستور `lsmod` لیست ماژول‌هایی که در حال حاضر بر روی هسته سیستم هستند را مشاهده می‌کنیم.

```
root@ubuntu: /home/maxeu/Desktop
File Edit View Search Terminal Help
sch_fq_codel                         20480  2
parport_pc                           40960  0
ppdev                                24576  0
lp                                    20480  0
parport                              53248  3 parport_pc,lp,ppdev
ip_tables                            32768  0
x_tables                             45056  1 ip_tables
autofs4                              45056  2
hid_generic                          16384  0
usbhid                               53248  0
hid                                  126976 2 usbhid,hid_generic
psmouse                             151552 0
mptspi                               24576  1
mptscsih                             40960  1 mptspi
mptbase                              94208  2 mptspi,mptscsih
ahci                                  40960  2
libahci                              32768  1 ahci
e1000                                143360 0
scsi_transport_spi                   32768  1 mptspi
i2c_piix4                            28672  0
pata_acpi                            16384  0
floppy                               81920  0
root@ubuntu: /home/maxeu/Desktop# insmod nvmem-rave-sp-EEPROM.ko
root@ubuntu: /home/maxeu/Desktop#
```

در این مرحله به سادگی با استفاده از دستور `insmod nvmem-rave-sp-eeprom.ko` ماژول مورد نظر را بر روی هسته سیستم عامل باگذاری می کنیم.

```
root@ubuntu: /home/maxeu/Desktop
File Edit View Search Terminal Help
floppy 81920 0
root@ubuntu:/home/maxeu/Desktop# insmod nvmem-rave-sp-eeprom.ko
root@ubuntu:/home/maxeu/Desktop# lsmod
Module Size Used by
nvmem_rave_sp_eeprom 16384 0
vmw_vsock_vmci_transport 32768 2
vsock 36864 3 vmw_vsock_vmci_transport
intel_rapl_msr 20480 0
intel_rapl_common 24576 1 intel_rapl_msr
crc10dif_pclmul 16384 1
crc32_pclmul 16384 0
ghash_clmulni_intel 16384 0
aesni_intel 372736 0
crypto_simd 16384 1 aesni_intel
cryptd 24576 2 crypto_simd,ghash_clmulni_intel
glue_helper 16384 1 aesni_intel
rapl 20480 0
vmw_balloon 24576 0
input_leds 16384 0
joydev 28672 0
serio_raw 20480 0
snd_ens1371 28672 4
snd_ac97_codec 135168 1 snd_ens1371
```

بار دیگر دستور `lsmod` را اجرا می کنیم. مشاهده می شود که ماژول فوق به لیست ماژول های هسته اضافه شده است.

```
root@ubuntu: /home/maxeu/Desktop
File Edit View Search Terminal Help
sch_fq_codel 20480 2
parport_pc 40960 0
ppdev 24576 0
lp 20480 0
parport 53248 3 parport_pc,lp,ppdev
ip_tables 32768 0
x_tables 45056 1 ip_tables
autofs4 45056 2
hid_generic 16384 0
usbhid 53248 0
hid 126976 2 usbhid,hid_generic
psmouse 151552 0
mptspi 24576 1
mptscsih 40960 1 mptspi
mptbase 94208 2 mptspi,mptscsih
ahci 40960 2
libahci 32768 1 ahci
e1000 143360 0
scsi_transport_spi 32768 1 mptspi
i2c_piix4 28672 0
pata_acpi 16384 0
floppy 81920 0
root@ubuntu:/home/maxeu/Desktop# rmmod nvmem_rave_sp_eeprom
root@ubuntu:/home/maxeu/Desktop#
```

برای حذف این ماژول نیز از دستور `rmmod` (بر خلاف `insmod`) استفاده می کنیم.

```
root@ubuntu: /home/maxeu/Desktop
File Edit View Search Terminal Help
pata_acpi 16384 0
floppy 81920 0
root@ubuntu: /home/maxeu/Desktop# rmmod nvme_rave_sp_eeprom
root@ubuntu: /home/maxeu/Desktop# lsmod
Module Size Used by
vmw_vsock_vmci_transport 32768 2
vsock 36864 3 vmw_vsock_vmci_transport
intel_rapl_msrm 20480 0
intel_rapl_common 24576 1 intel_rapl_msrm
crct10dif_pclmul 16384 1
crc32_pclmul 16384 0
ghash_clmulni_intel 16384 0
aesni_intel 372736 0
crypto_simd 16384 1 aesni_intel
cryptd 24576 2 crypto_simd,ghash_clmulni_intel
glue_helper 16384 1 aesni_intel
rapl 20480 0
vmw_balloon 24576 0
input_leds 16384 0
joydev 28672 0
serio_raw 20480 0
snd_ens1371 28672 6
snd_ac97_codec 135168 1 snd_ens1371
```

با اجرای دوباره دستور lsmod مطمئن می‌شویم که ماژول فوق با موفقیت حذف شده و دیگر میان لیست ماژول‌های هسته وجود ندارد.

## ۲. بخش ۱ – ایجاد ماژول‌های هسته:

در این قسمت از آزمایش ابتدا یک ماژول هسته به زبان C و مطابق با دستور کار می‌نویسیم. سپس این ماژول را make می‌کنیم تا از فایل C نوشته شده یک خروجی قابل بارگذاری بر هسته سیستم عامل با پسوند ko. ایجاد شود. سپس مانند قسمت قبلی آزمایش این فایل ko. را بر روی هسته بارگذاری کرده و در انتهای آزمایش آن را حذف می‌کنیم. همچنین در این مراحل با استفاده از دستور dmesg بافر هسته را بررسی می‌کنیم تا از بارگذاری و حذف صحیح ماژول فوق اطمینان حاصل کنیم.

کد مربوط به این ماژول در ادامه آمده و توضیحات مربوطه داده شده است.

```

#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("A simple module written to learn about Linux kernel modules");
MODULE_AUTHOR("Negar Movaghatian - 9831062");

```

در خطوط ابتدایی کتابخانه‌های لازم برای ایجاد ماژول هسته را اضافه کرده‌ایم.

توابع `simple_init` و `simple_exit` نقاط شروع و پایان برنامه ماژول هستند که با استفاده از ماکروهای `module_init` و `module_exit` که در خطوط پایانی برنامه نوشته شده‌اند این دو تابع را به عنوان تابع شروع و پایان معرفی می‌کنیم. در این ماژول ساده تنها کاری که این دو تابع انجام می‌دهند نوشتن پیغامی هنگام بارگذاری و حذف ماژول بر روی بافر هسته است تا از طریق آن بتوانیم از موفقیت‌آمیز بودن روند کار اطمینان حاصل کنیم. در انتهای برنامه نیز توضیحاتی درباره ماژول فوق را در قالب ماکروهای از پیش تعریف شده نوشته‌ایم.

و اما برای بارگذاری این ماژول بر روی هسته نیاز به یک فایل خروجی با پسوند `.ko` داریم. برای `build` کردن برنامه از یک `makefile` و دستور `make` استفاده برای اجرای آن استفاده می‌کنیم. برای استفاده از این دستور و کامپایل کردن برنامه نوشته شده به زبان C ابتدا با استفاده از دستورات زیر `make` و `gcc` را بر روی سیستم عامل نصب می‌کنیم:

```

apt get update
apt install make
apt install gcc

```

به علاوه از یک makefile با محتوای زیر استفاده می‌کنیم. این فایل را در فولدر برنامه نوشته شده قرار می‌دهیم تا make بتواند دستورالعمل‌های آن را اجرا کند.

```
KERNELDIR=/lib/modules/`uname -r`/build
#ARCH=i386
#KERNELDIR=/usr/src/kernels/`uname -r`-i686

MODULES = simple-module.ko
obj-m += simple-module.o

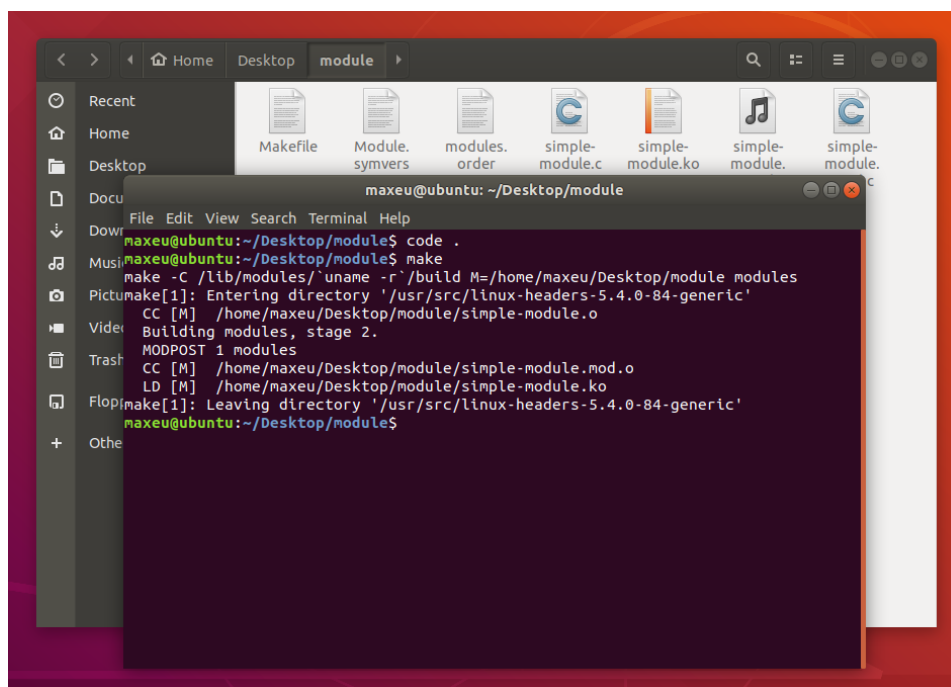
all:
    make -C $(KERNELDIR) M=$(PWD) modules

clean:
    make -C $(KERNELDIR) M=$(PWD) clean
```

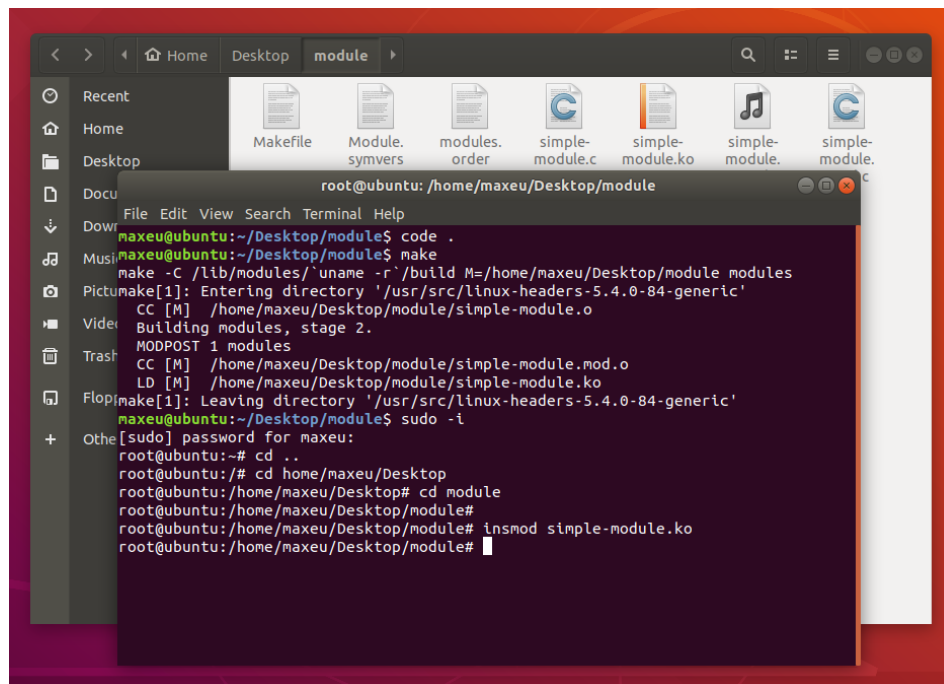
برای مشخص کردن محتوای این فایل از سایت زیر استفاده شده‌است:

[https://www.cs.bham.ac.uk/~exr/lectures/systems/07\\_08/kernelProgramming.php](https://www.cs.bham.ac.uk/~exr/lectures/systems/07_08/kernelProgramming.php)

برای جلوگیری از خطای Invalid module format و تطابق ماژول build شده با نسخه kernel مسیر هسته سیستم عامل (تحت عنوان KERNELDIR) به عنوان آرگومان به make داده شده‌است. در خط ۴ و ۵ نیز نام فایل‌هایی خروجی ماژول (با پسوند .ko) و object ها (با پسوند .o) مشخص شده‌اند. با قرار دادن این فایل در فولدر برنامه‌ای که نوشته‌ایم و اجرای دستور make می‌توان فایل‌های خروجی را بدست آورد.

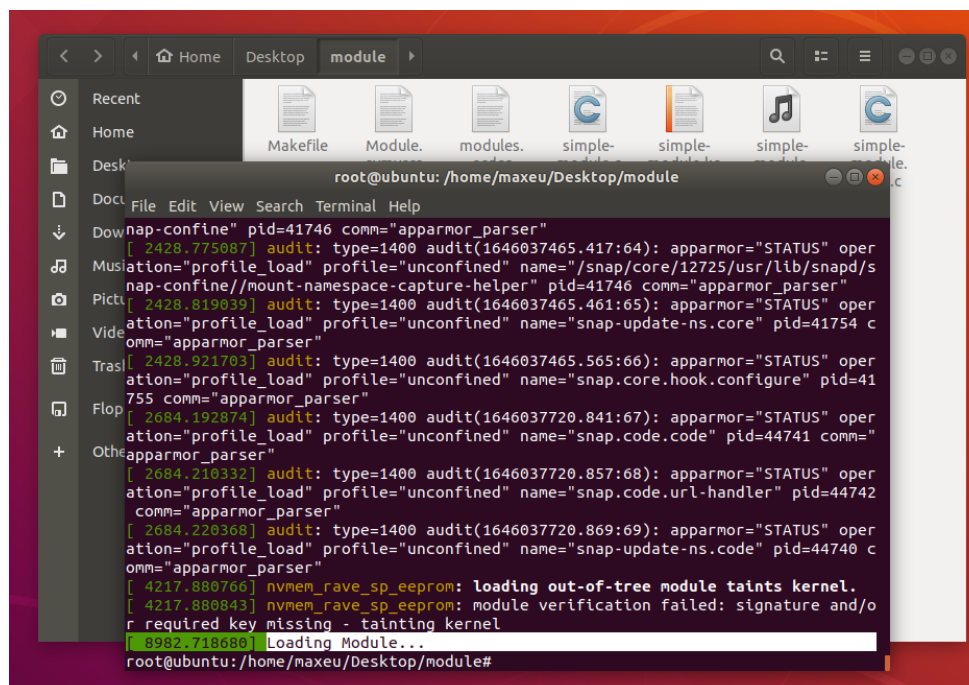


پس از طی این مراحل و بدست آوردن فایل خروجی کافیست مانند بخش اول آزمایش سطح دسترسی را به root تغییر داده و دستور insmod را اجرا کنیم.



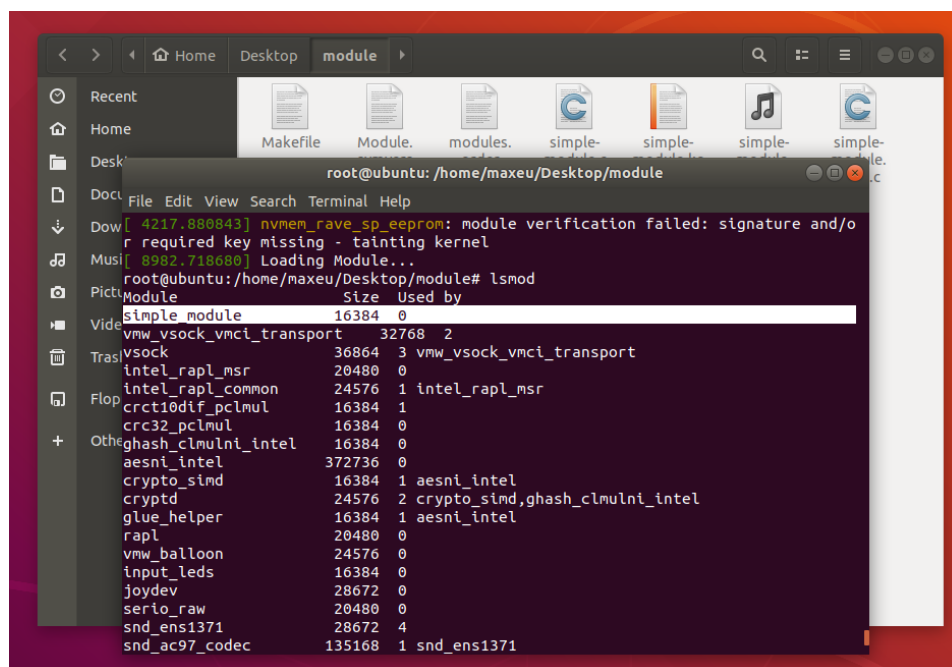
```
root@ubuntu: /home/maxeu/Desktop/module
maxeu@ubuntu:~/Desktop/module$ code .
maxeu@ubuntu:~/Desktop/module$ make
make -C /lib/modules/`uname -r`/build M=/home/maxeu/Desktop/module modules
make[1]: Entering directory '/usr/src/linux-headers-5.4.0-84-generic'
CC [M] /home/maxeu/Desktop/module/simple-module.o
Building modules, stage 2.
MODPOST 1 modules
CC [M] /home/maxeu/Desktop/module/simple-module.mod.o
LD [M] /home/maxeu/Desktop/module/simple-module.ko
make[1]: Leaving directory '/usr/src/linux-headers-5.4.0-84-generic'
maxeu@ubuntu:~/Desktop/module$ sudo -i
[sudo] password for maxeu:
root@ubuntu:~# cd ..
root@ubuntu:~/Desktop# cd module
root@ubuntu:~/Desktop/module#
root@ubuntu:~/Desktop/module# insmod simple-module.ko
root@ubuntu:~/Desktop/module#
```

طبق دستور کار با استفاده از دستور dmesg بافر هسته را چک می کنیم تا بررسی کنیم پیغامی که انتظار داشتیم زمان بارگذاری ماژول هسته چاپ شود در بافر نوشته شده یا خیر.



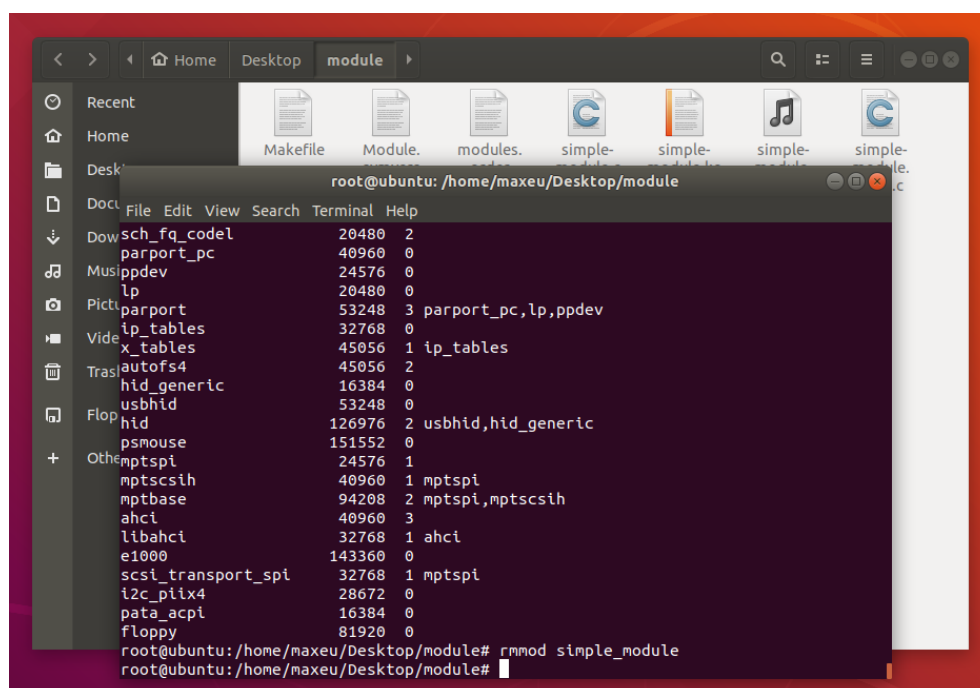
```
root@ubuntu: /home/maxeu/Desktop/module
[ 2428.775087] audit: type=1400 audit(1646037465.417:64): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/snap/core/12725/usr/lib/snapd/snap-confine/mount-namespace-capture-helper" pid=41746 comm="apparmor_parser"
[ 2428.819039] audit: type=1400 audit(1646037465.461:65): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap-update-ns.core" pid=41754 comm="apparmor_parser"
[ 2428.921703] audit: type=1400 audit(1646037465.565:66): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap.core.hook.configure" pid=41755 comm="apparmor_parser"
[ 2684.192874] audit: type=1400 audit(1646037720.841:67): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap.code.code" pid=44741 comm="apparmor_parser"
[ 2684.210332] audit: type=1400 audit(1646037720.857:68): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap.code.url-handler" pid=44742 comm="apparmor_parser"
[ 2684.220368] audit: type=1400 audit(1646037720.869:69): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap-update-ns.code" pid=44740 comm="apparmor_parser"
[ 4217.880760] nvmem_rave_sp_eeprom: loading out-of-tree module taints kernel.
[ 4217.880843] nvmem_rave_sp_eeprom: module verification failed: signature and/or required key missing - tainting kernel
[ 8982.718680] Loading Module...
root@ubuntu:~/Desktop/module#
```

همانطور که انتظار داشتیم عبارت “Loading Module...” چاپ شده‌است. به علاوه با استفاده از دستور lsmod لیست ماژول‌های هسته را بررسی می‌کنیم.



```
root@ubuntu: /home/maxeu/Desktop/module# lsmod
Module                  Size  Used by
simple_module            16384  0
vmw_vsock_vmci_transport 32768  2
vsock                   36864  3 vmw_vsock_vmci_transport
intel_rapl_msr          20480  0
intel_rapl_common       24576  1 intel_rapl_msr
crc10dif_pclmul         16384  1
crc32_pclmul            16384  0
ghash_clmulni_intel     16384  0
aesni_intel            372736 0
crypto_simd             16384  1 aesni_intel
cryptd                  24576  2 crypto_simd,ghash_clmulni_intel
glue_helper             16384  1 aesni_intel
rapl                    20480  0
vmw_balloon            24576  0
input_leds             16384  0
joydev                  28672  0
serio_raw               20480  0
snd_ens1371             28672  4
snd_ac97_codec          135168 1 snd_ens1371
```

ماژول simple\_module در این لیست قرار دارد، بنابراین ماژول هسته‌ای که نوشته بودیم با موفقیت بارگذاری شده‌است. حال مانند قسمت اول آزمایش ماژول نوشته شده را با استفاده از دستور rmmmod حذف می‌کنیم.



```
root@ubuntu: /home/maxeu/Desktop/module# rmmmod simple_module
root@ubuntu: /home/maxeu/Desktop/module#
```



بار دیگر از دستور lsmod استفاده می‌کنیم. این بار ماژول مورد نظر در لیست ماژول‌های هسته وجود ندارد.

```

root@ubuntu: /home/maxeu/Desktop/module# rmmod simple_module
root@ubuntu: /home/maxeu/Desktop/module# lsmod
Module                  Size  Used by
vmw_vsock_vmci_transport 32768  2
vsock                   36864  3 vmw_vsock_vmci_transport
intel_rapl_msr           20480  0
intel_rapl_common        24576  1 intel_rapl_msr
crc32_pclmul             16384  1
ghash_clmulni_intel      16384  0
aesni_intel              372736 0
crypto_simd              16384  1 aesni_intel
cryptd                   24576  2 crypto_simd,ghash_clmulni_intel
glue_helper              16384  1 aesni_intel
rapl                     20480  0
vmw_balloon              24576  0
input_leds               16384  0
joydev                   28672  0
serio_raw                20480  0
snd_ens1371              28672  4
snd_ac97_codec           135168 1 snd_ens1371

```

و با استفاده از دستور dmesg بافر هسته را چک می‌کنیم تا بررسی کنیم پیغامی که انتظار داشتیم زمان حذف ماژول هسته چاپ شود در بافر نوشته شده یا خیر. عبارت “Removing Module...” در بافر چاپ شده‌است، بنابراین حذف ماژول نیز با موفقیت انجام شده.

```

root@ubuntu: /home/maxeu/Desktop/module# dmesg
[ 2428.775087] audit: type=1400 audit(1646037465.417:64): apparmor="STATUS" operation="profile_load" profile="unconfined" name="/snap/core/12725/usr/lib/snapd/snap-confine/mount-namespace-capture-helper" pid=41746 comm="apparmor_parser"
[ 2428.819039] audit: type=1400 audit(1646037465.461:65): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap-update-ns.core" pid=41754 comm="apparmor_parser"
[ 2428.921703] audit: type=1400 audit(1646037465.565:66): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap.core.hook.configure" pid=41755 comm="apparmor_parser"
[ 2684.192874] audit: type=1400 audit(1646037720.841:67): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap.code.code" pid=44741 comm="apparmor_parser"
[ 2684.210332] audit: type=1400 audit(1646037720.857:68): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap.code.url-handler" pid=44742 comm="apparmor_parser"
[ 2684.220368] audit: type=1400 audit(1646037720.869:69): apparmor="STATUS" operation="profile_load" profile="unconfined" name="snap-update-ns.code" pid=44740 comm="apparmor_parser"
[ 4217.880766] nvmem_rave_sp_eeeprom: loading out-of-tree module taints kernel.
[ 4217.880843] nvmem_rave_sp_eeeprom: module verification failed: signature and/or required key missing - tainting kernel
[ 8982.718680] Loading Module...
[ 9140.421642] Removing Module...
root@ubuntu: /home/maxeu/Desktop/module#

```

### ۳. بخش ۲ – ساختمان داده‌های هسته:

در این قسمت نیز مانند قسمت قبل آزمایش با استفاده از زبان C یک ماژول هسته نوشته، آن را بر روی هسته بارگذاری کرده و سپس حذف می‌کنیم. کد مربوط به هر بخش از این ماژول در ادامه آمده و توضیحات مربوطه داده شده است.

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

در ابتدا کتابخانه‌های مورد نیاز را به ماژول مورد نظر اضافه می‌کنیم. سه کتابخانه اول برای نوشتن ماژول‌های هسته، کتابخانه بعدی برای استفاده از ساختمان داده Linked List موجود در هسته لینوکس و کتابخانه آخر برای استفاده از دستور kcalloc برای تخصیص حافظه هسته استفاده می‌شود.

```
static LIST_HEAD(birthday_list);

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

در خطوط بعدی برنامه ابتدا با استفاده از ماکروی LIST\_HEAD متغیر birthday\_list را به عنوان ابتدای linked list ای که در ادامه از آن استفاده می‌کنیم تعریف و مقداری می‌کنیم. سپس یک struct به نام birthday می‌سازیم که یک تاریخ را در خود نگهداری کرده و به دلیل داشتن یک عضو از جنس list\_head قابلیت استفاده به عنوان یک node در linked list هسته را دارد.

در تابع simple\_init که در صفحه بعد آمده (و در انتهای برنامه آن را به عنوان نقطه شروع ماژول معرفی می‌کنیم)، ابتدا یک پیغام بر روی بافر هسته قرار می‌دهیم. زمان بارگذاری ماژول هسته این پیغام به ما کمک می‌کند تا مطمئن باشیم ماژول به درستی بر روی هسته سیستم عامل بارگذاری شده است. در ادامه ۵ نمونه از struct birthday را ساخته، به آن حافظه تخصیص داده، فیلدهای آن را مقداری کرده و در آخر آن را به انتهای linked list ای که ساخته بودیم اضافه می‌کنیم. پس از اضافه شدن تمام افراد با استفاده از دستور list\_for\_each\_entry بر روی تمام اعضاء linked list ساخته شده پیمایش کرده و اطلاعات مربوط به آن را چاپ

می‌کنیم. در انتها نیز پیغامی بر روی بافر هسته چاپ می‌کنیم تا مطمئن شویم تمام node ها با موفقیت اضافه شده‌اند.

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

    struct birthday *person1;
    person1 = kmalloc(sizeof(person1), GFP_KERNEL);
    person1->day = 12;
    person1->month = 10;
    person1->year = 2001;
    INIT_LIST_HEAD(&person1->list);
    list_add_tail(&person1->list, &birthday_list);

    .
    .
    .

    struct birthday *ptr;
    int i = 1;
    list_for_each_entry(ptr, &birthday_list, list)
        printk(KERN_INFO "%d. %d/%d/%d\n", i++, ptr->year, ptr->month, ptr->day);

    printk(KERN_INFO "End of init function. Added all nodes successfully!\n");

    return 0;
}
```

در تابع simple\_exit نیز (که در انتهای برنامه آن را به عنوان نقطهٔ پایان ماژول معرفی می‌کنیم)، ابتدا یک پیغام بر روی بافر هسته قرار می‌دهیم. زمان حذف ماژول هسته این پیغام به ما کمک می‌کند تا مطمئن باشیم ماژول به درستی از روی هستهٔ سیستم عامل حذف شده‌است.

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

    int i = 1;
    struct list_head *ptr, *tmp;
    list_for_each_safe(ptr, tmp, &birthday_list){
        printk(KERN_INFO "Freeing node %d\n", i++);
        list_del(ptr);
        kfree(ptr);
    }

    printk(KERN_INFO "End of exit function. removed all nodes successfully!\n");
}
```

به دلیل این که زمانی که این تابع در حال اجرا است می‌خواهیم ماژول را از روی هسته برداریم باید حافظه هسته که در تابع `simple_init` به متغیرهای این ماژول اختصاص داده بودیم را به هسته برگردانیم، بنابراین نیاز داریم که بر روی تمام عناصر `linked list` پیمایش شده حرکت کنیم، آن‌ها را از `linked list` حذف کرده و با استفاده از دستور `kfree` حافظه تخصیص داده شده به آن را آزاد کنیم. در این زمان به ازای هر `node` پیغامی نیز بر روی بافر هسته چاپ می‌کنیم تا مطمئن شویم تمام `node` ها حذف شده‌اند.

اما برای پیمایش بر روی تمام اعضاء این بار از دستور `list_for_each_safe` استفاده می‌کنیم، زیرا می‌خواهیم همزمان با پیمایش لیست `node` هایی را از آن حذف کنیم و استفاده از `for_each` ساده می‌تواند باعث مشکلاتی شود. برای فراگیری طریقه استفاده از این دستور از سایت‌های زیر استفاده شده‌است:

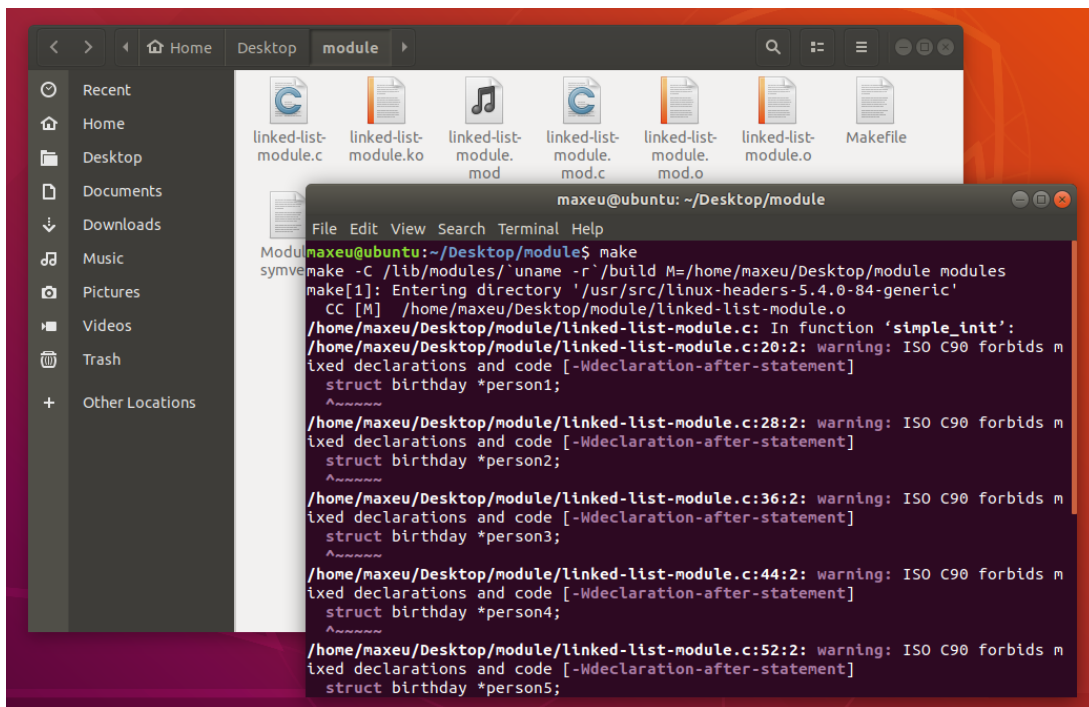
<https://stackoverflow.com/questions/9207850/why-do-we-need-list-for-each-safe-in-for-deleting-nodes-in-kernel-linked-list>

<https://www.kernel.org/doc/htmldocs/kernel-api/API-list-for-each-safe.html>

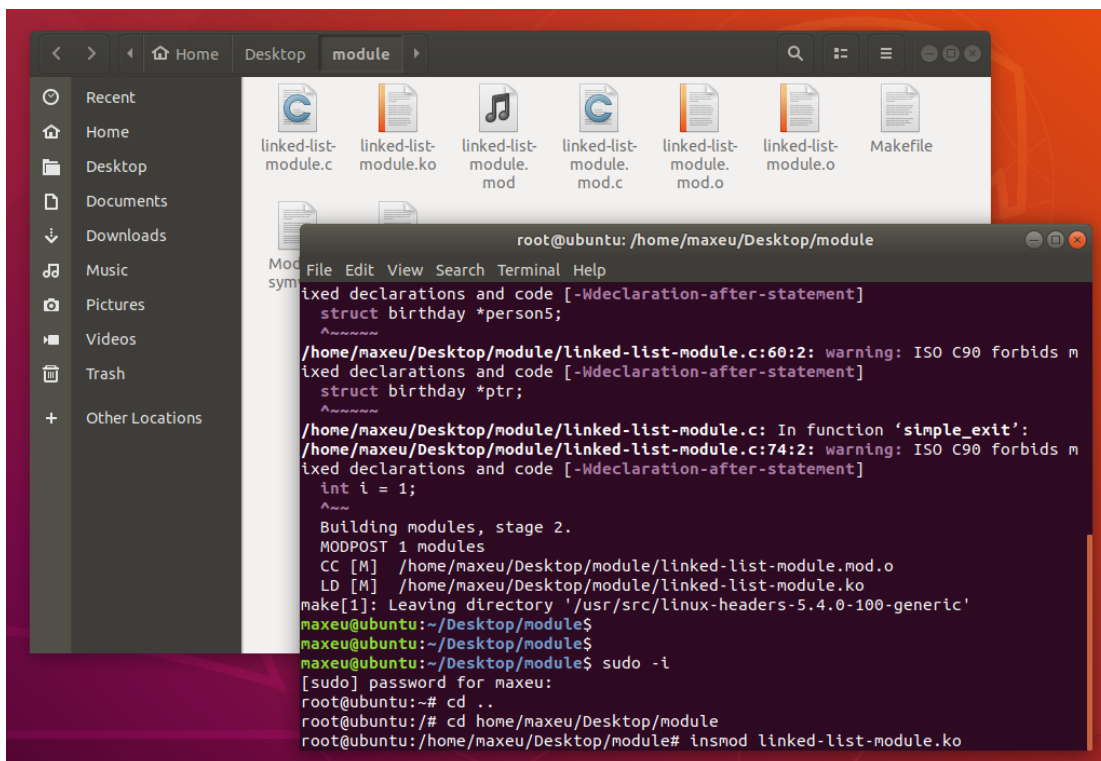
```
/* Macros for registering module entry and exit points */
module_init(simple_init);
module_exit(simple_exit);
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("A module written to work with Linux kernel data structures");
MODULE_AUTHOR("Negar Movaghatian - 9831062");
```

در انتها نیز با استفاده از ماکروهای `module_init` و `module_exit` دو تابع `simple_init` و `simple_exit` را به عنوان توابع شروع و پایان ماژول معرفی می‌کنیم. به علاوه توضیحاتی درباره ماژول فوق را در قالب ماکروهای از پیش تعریف شده اضافه می‌کنیم.

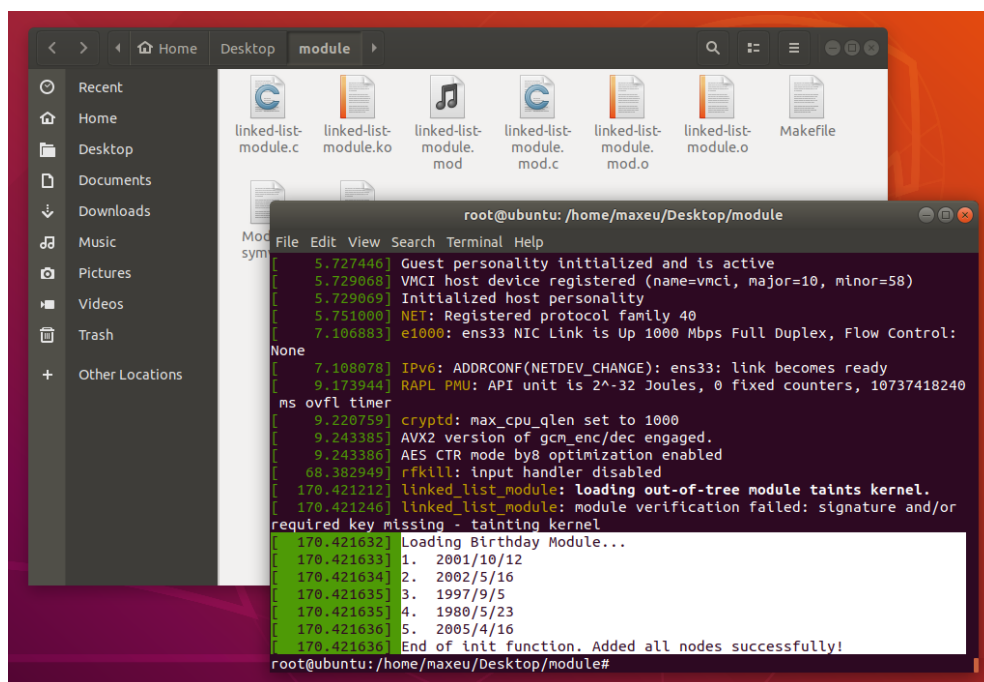
حال برای بارگذاری این ماژول بر روی هسته نیاز به یک فایل خروجی با پسوند `.ko` داریم. برای `build` کردن برنامه مانند آزمایش قبل از یک `makefile` و دستور `make` استفاده برای اجرای آن استفاده می‌کنیم. توضیحات مربوط به این فایل در آزمایش قبل آمده‌است. تنها کفایت نام فایل‌های خروجی را به نام این ماژول تغییر دهیم. با قرار دادن این فایل در فولدر برنامه‌ای که نوشته‌ایم و اجرای دستور `make` می‌توان فایل‌های خروجی را بدست آورد.



پس از طی این مراحل و بدست آوردن فایل خروجی کافیت مانند بخش‌های قبلی آزمایش سطح دسترسی را به root تغییر داده و دستور insmod را اجرا کنیم.

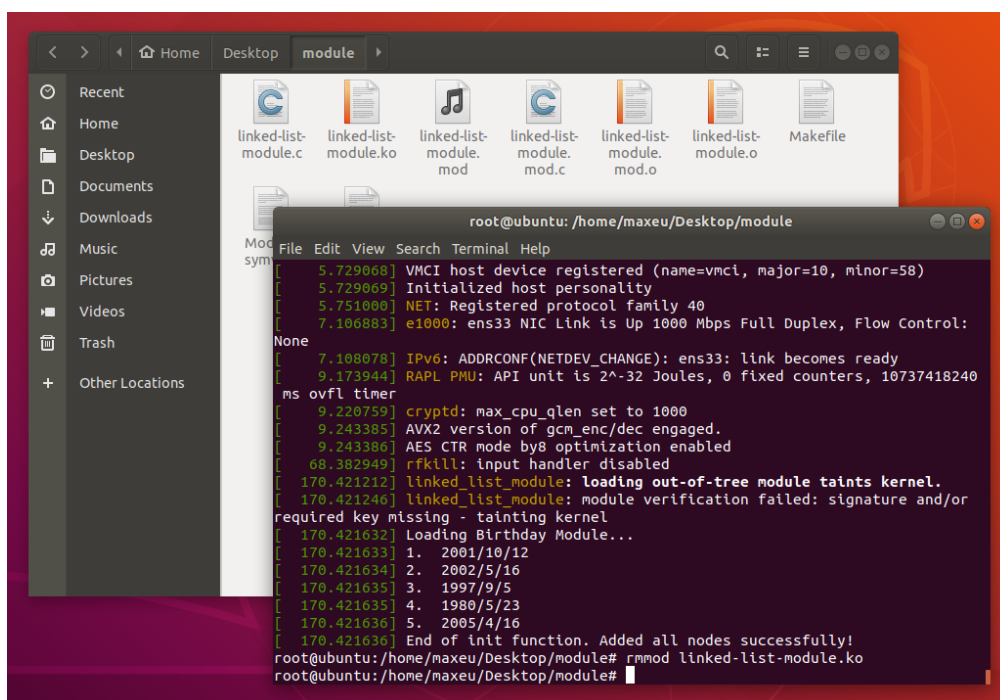


با استفاده از دستور dmesg بافر هسته را چک می‌کنیم تا بررسی کنیم پیغامی که انتظار داشتیم زمان بارگذاری ماژول هسته چاپ شود در بافر نوشته شده یا خیر.



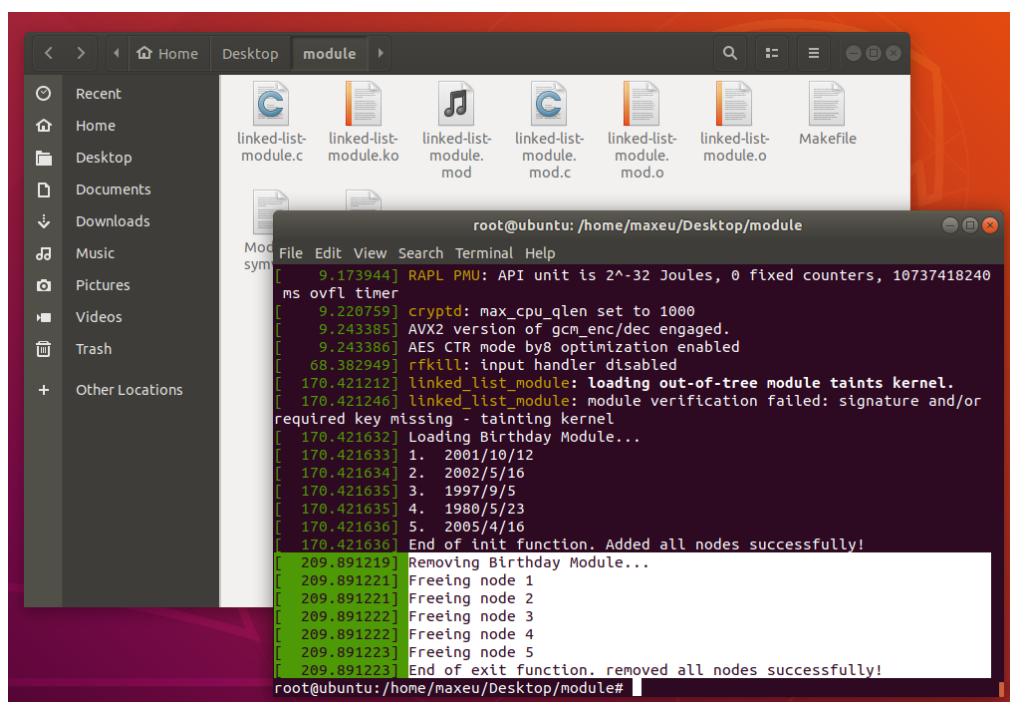
```
root@ubuntu: /home/maxeu/Desktop/module
[ 5.727446] Guest personality initialized and is active
[ 5.729068] VMCI host device registered (name=vmci, major=10, minor=58)
[ 5.729069] Initialized host personality
[ 5.751000] NET: Registered protocol family 40
[ 7.106883] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
None
[ 7.108078] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[ 9.173944] RAPL PMU: API unit is 2^-32 Joules, 0 fixed counters, 10737418240
ms ovfl timer
[ 9.220759] cryptd: max_cpu_qlen set to 1000
[ 9.243385] AVX2 version of gcm_enc/dec engaged.
[ 9.243386] AES CTR mode by8 optimization enabled
[ 68.382949] rfkill: input handler disabled
[ 170.421212] linked_list_module: loading out-of-tree module taints kernel.
[ 170.421246] linked_list_module: module verification failed: signature and/or
required key missing - tainting kernel
[ 170.421632] Loading Birthday Module...
[ 170.421633] 1. 2001/10/12
[ 170.421634] 2. 2002/5/16
[ 170.421635] 3. 1997/9/5
[ 170.421635] 4. 1980/5/23
[ 170.421636] 5. 2005/4/16
[ 170.421636] End of init function. Added all nodes successfully!
root@ubuntu: /home/maxeu/Desktop/module#
```

همانطور که انتظار داشتیم عبارت “Loading Birthday Module...” و لیست تمامی node ها چاپ شده‌است. حال مانند قسمت‌های قبلی آزمایش ماژول نوشته شده را با استفاده از دستور rmmmod حذف می‌کنیم.



```
root@ubuntu: /home/maxeu/Desktop/module
[ 5.729068] VMCI host device registered (name=vmci, major=10, minor=58)
[ 5.729069] Initialized host personality
[ 5.751000] NET: Registered protocol family 40
[ 7.106883] e1000: ens33 NIC Link is Up 1000 Mbps Full Duplex, Flow Control:
None
[ 7.108078] IPv6: ADDRCONF(NETDEV_CHANGE): ens33: link becomes ready
[ 9.173944] RAPL PMU: API unit is 2^-32 Joules, 0 fixed counters, 10737418240
ms ovfl timer
[ 9.220759] cryptd: max_cpu_qlen set to 1000
[ 9.243385] AVX2 version of gcm_enc/dec engaged.
[ 9.243386] AES CTR mode by8 optimization enabled
[ 68.382949] rfkill: input handler disabled
[ 170.421212] linked_list_module: loading out-of-tree module taints kernel.
[ 170.421246] linked_list_module: module verification failed: signature and/or
required key missing - tainting kernel
[ 170.421632] Loading Birthday Module...
[ 170.421633] 1. 2001/10/12
[ 170.421634] 2. 2002/5/16
[ 170.421635] 3. 1997/9/5
[ 170.421635] 4. 1980/5/23
[ 170.421636] 5. 2005/4/16
[ 170.421636] End of init function. Added all nodes successfully!
root@ubuntu: /home/maxeu/Desktop/module# rmmmod linked-list-module.ko
root@ubuntu: /home/maxeu/Desktop/module#
```

بار دیگر با استفاده از دستور dmesg بافر هسته را چک می‌کنیم تا بررسی کنیم پیغامی که انتظار داشتیم زمان حذف ماژول هسته چاپ شود در بافر نوشته شده یا خیر.



```
root@ubuntu: /home/maxeu/Desktop/module
File Edit View Search Terminal Help
[ 9.173944] RAPL PMU: API unit is 2^32 Joules, 0 fixed counters, 10737418240
ms ovfl timer
[ 9.220759] cryptd: max_cpu_qlen set to 1000
[ 9.243385] AVX2 version of gcm_enc/dec engaged.
[ 9.243386] AES CTR mode by8 optimization enabled
[ 68.382949] rfkill: input handler disabled
[ 170.421212] linked_list_module: loading out-of-tree module taints kernel.
[ 170.421246] linked_list_module: module verification failed: signature and/or
required key missing - tainting kernel
[ 170.421632] Loading Birthday Module...
[ 170.421633] 1. 2001/10/12
[ 170.421634] 2. 2002/5/16
[ 170.421635] 3. 1997/9/5
[ 170.421635] 4. 1980/5/23
[ 170.421636] 5. 2005/4/16
[ 170.421636] End of init function. Added all nodes successfully!
[ 209.891219] Removing Birthday Module...
[ 209.891221] Freeing node 1
[ 209.891221] Freeing node 2
[ 209.891222] Freeing node 3
[ 209.891222] Freeing node 4
[ 209.891223] Freeing node 5
[ 209.891223] End of exit function. removed all nodes successfully!
root@ubuntu: /home/maxeu/Desktop/module#
```

عبارت “Removing Birthday Module...” و “Freeing node” به ازای تمام node ها در بافر چاپ شده‌است، بنابراین حذف ماژول نیز با موفقیت انجام شده.