

# بخش چهارم

## کسرا نوربخش 810100230

### مقدمه

در این بخش با برنامه نویسی Kernel آشنا شدیم.

### نوشتن مژول کرنل

دقیقاً مطابق به صورت پروژه فایل ایجاد شد و با استفاده از مکرو های گفته شده Metadat ها را include ها صورت گرفت و با استفاده از مکرو های گفته شده include کار اصلی را انجام می دهنند که پرینت می کنند در لاگ کرنل. هم اضافه نمودیم. دو تابع hello\_init و hello\_exit هم اضافه نمودیم.

### ایجاد Makefile

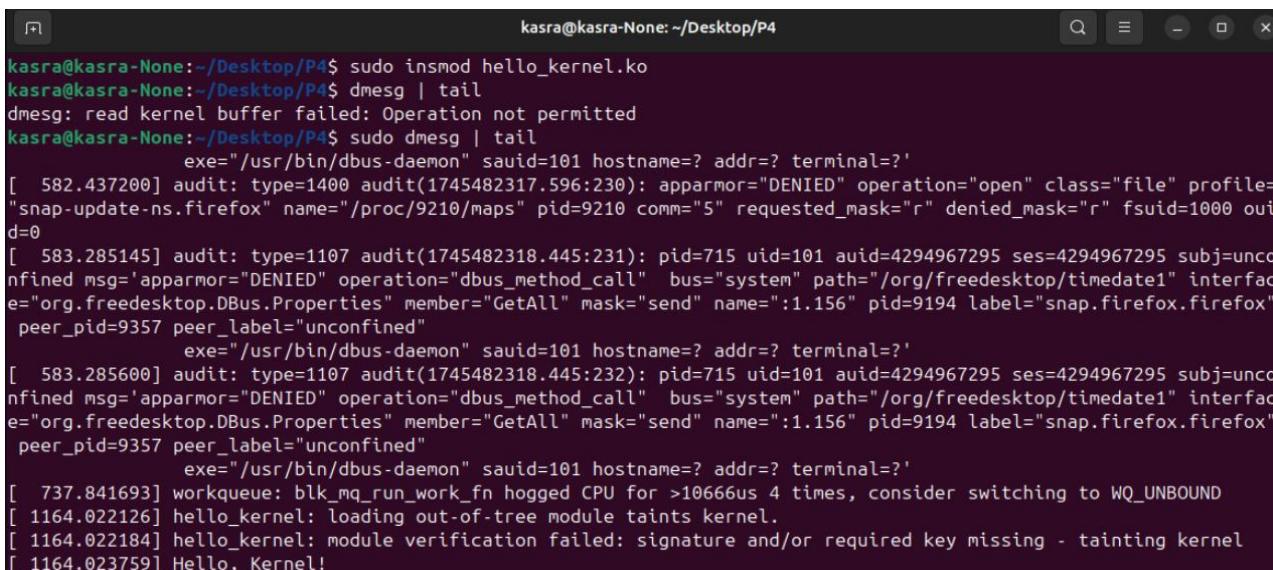
میک فایلی که در صورت پروژه آمده بود کپی شد و استفاده شد.

### فرایند کامپایل

در این بخش make را زدیم ☺

### و Remove Insert

با استفاده از sudo rmmod مژول را insert کردیم و با استفاده از sudo insmod hello\_kernel.ko مژول را مشاهده کردیم. در هر بار با استفاده از sudo dmesg | tail لاگ را مشاهده کردیم.



```
kasra@kasra-None:~/Desktop/P4$ sudo insmod hello_kernel.ko
kasra@kasra-None:~/Desktop/P4$ dmesg | tail
dmsg: read kernel buffer failed: Operation not permitted
kasra@kasra-None:~/Desktop/P4$ sudo dmesg | tail
    exe="/usr/bin/dbus-daemon" sauid=101 hostname=? addr=? terminal=?
[ 582.437200] audit: type=1400 audit(1745482317.596:230): apparmor="DENIED" operation="open" class="file" profile="snap-update-ns.firefox" name="/proc/9210/maps" pid=9210 comm="5" requested_mask="r" denied_mask="r" fsuid=1000 ouid=0
[ 583.285145] audit: type=1107 audit(1745482318.445:231): pid=715 uid=101 auid=4294967295 ses=4294967295 subj=unconfined msg='apparmor="DENIED" operations="dbus_method_call" bus="system" path="/org/freedesktop/timedate1" interface="org.freedesktop.DBus.Properties" member="GetAll" mask="send" name=:1.156' pid=9194 label="snap.firefox.firefox" peer_pid=9357 peer_label="unconfined"
    exe="/usr/bin/dbus-daemon" sauid=101 hostname=? addr=? terminal=?
[ 583.285600] audit: type=1107 audit(1745482318.445:232): pid=715 uid=101 auid=4294967295 ses=4294967295 subj=unconfined msg='apparmor="DENIED" operations="dbus_method_call" bus="system" path="/org/freedesktop/timedate1" interface="org.freedesktop.DBus.Properties" member="GetAll" mask="send" name=:1.156' pid=9194 label="snap.firefox.firefox" peer_pid=9357 peer_label="unconfined"
    exe="/usr/bin/dbus-daemon" sauid=101 hostname=? addr=? terminal=?
[ 737.841693] workqueue: blk_mq_run_work_fn hogged CPU for >10666us 4 times, consider switching to WQ_UNBOUND
[ 1164.022126] hello_kernel: loading out-of-tree module taints kernel.
[ 1164.022184] hello_kernel: module verification failed: signature and/or required key missing - tainting kernel
[ 1164.023759] Hello, Kernel!
```

عکس 1: لاگ بعد از زدن insmod

```

kasra@kasra-None:~/Desktop/P4$ sudo rmmod hello_kernel
kasra@kasra-None:~/Desktop/P4$ sudo dmesg | tail
[ 583.285145] audit: type=1107 audit(1745482318.445:231): pid=715 uid=101 auid=4294967295 ses=4294967295 subj=unco
nfigured msg='apparmor="DENIED" operation="dbus_method_call" bus="system" path="/org/freedesktop/timedate1" interfac
e="org.freedesktop.DBus.Properties" member="GetAll" mask="send" name=:1.156 pid=9194 label="snap.firefox.firefox"
 peer_pid=9357 peer_label="unconfined"
        exe="/usr/bin/dbus-daemon" sauid=101 hostname=? addr=? terminal=?
[ 583.285600] audit: type=1107 audit(1745482318.445:232): pid=715 uid=101 auid=4294967295 ses=4294967295 subj=unco
nfigured msg='apparmor="DENIED" operation="dbus_method_call" bus="system" path="/org/freedesktop/timedate1" interfac
e="org.freedesktop.DBus.Properties" member="GetAll" mask="send" name=:1.156 pid=9194 label="snap.firefox.firefox"
 peer_pid=9357 peer_label="unconfined"
        exe="/usr/bin/dbus-daemon" sauid=101 hostname=? addr=? terminal=?
[ 737.841693] workqueue: blk_mq_run_work_fn hogged CPU for >10666us 4 times, consider switching to WQ_UNBOUND
[ 1164.022126] hello_kernel: loading out-of-tree module taints kernel.
[ 1164.022184] hello_kernel: module verification failed: signature and/or required key missing - tainting kernel
[ 1164.023759] Hello, Kernel!
[ 1215.458406] workqueue: blk_mq_run_work_fn hogged CPU for >10666us 8 times, consider switching to WQ_UNBOUND
[ 1240.549086] Goodbye, Kernel!

```

## عکس 2: لاغ بعد از زدن rmmod

دو خط مشکوک در لاغ وجود داشت که در زیر آمد:

hello\_kernel: loading out-of-tree module taints kernel.

hello\_kernel: module verification failed: signature and/or required key missing - tainting kernel

اما به دلیل آنکه ما مازولمان را امضا دیجیتال نکرده ایم، به ما می گویند که این مازول تمرینی می باشد و استاندارد نیست.

## سوالات تشریحی

1. کرنل های Monolithic و Microkernel دو معماری متفاوت در طراحی کرنل او اس هستند. در کرنل Monolithic تمام سرویس های سیستم عامل مثل مدیریت حافظه، درایورها، فایل سیستم در فضای کرنل اجرا می شوند و این باعث می شود عملکرد سریع تری داشته باشد، اما در عوض اگر یک بخش دچار خطا شود، ممکن است کل سیستم از کار بیفتد (Single Point of Failure). اما، Microkernel فقط عملکردهای اصلی مانند مدیریت حافظه و اسکجوولینگ را در کرنل نگه می دارد و بقیه سرویس ها در فضای کاربر اجرا می شوند، که این موجب پایداری بیشتر و افزایش امنیت سیستم می شود، هرچند ممکن است عملکرد کند تر شود.

2. Kernel panic وضعیتی است که در آن کرنل با خطای روبه رو می شود که قادر به ادامه اجرای امن نیست، مانند باگی در دسترسی حافظه یا باگ در درایورها. در این حالت کرنل سیستم را متوقف کرده و عمدهاً یک پیام خطای نمایش داده می شود تا به برنامه نویس برای تشخیص علت کمک کند. برای مدیریت این وضعیت، برخی سیستمها اطلاعاتی مانند dump حافظه تولید می کنند که بعداً می توان از آن برای تحلیل خطای استفاده کرد.

3. برای دیباگ کردن مازول کرنل یا درایورها ابزارهای مختلفی وجود دارد مانند ftrace، printk، dmesg. یکی از ابزارهای رایج printk است که مشابه printf در برنامه نویسی است و برای چاپ پیام های دیباگ در لاغ کرنل استفاده می شود. این پیامها با استفاده از دستور dmesg قابل دیدن اند و در تشخیص مراحل اجرای مازول یا علت باگ ها بدرد می خورند.

## لینک های استفاده شده

برای پیش برد این بخش از ChatGPT کمک گرفته شد که لینک آن:

<https://chatgpt.com/share/6809f58e-a83c-800d-a5e5-3b0f23dbd44c>

چت کامل:

Write me a C code that has a Initialization function that using \_\_init macro, that uses printk, prints "Hello, Kernel!". It also must have a exit function that using \_\_exit macro, that uses printk, prints "Goodbye, Kernel!". we must add these headers: <linux/init.h>, <linux/kernel.h>, <linux/module.h>. We also must use these macros for adding Metadata: MODULE\_AUTHOR, MODULE\_LICENSE, MODULE\_DESCRIPTION.

**ChatGPT said:**

Here's a simple Linux Kernel Module in C that matches all your requirements:

C

CopyEdit

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

// Metadata about the module
MODULE_AUTHOR("Your Name");
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("A simple kernel module that prints Hello and Goodbye");

// Initialization function
static int __init hello_init(void)
{
    printk(KERN_INFO "Hello, Kernel!\n");
    return 0;
}

// Exit/cleanup function
static void __exit hello_exit(void)
{
    printk(KERN_INFO "Goodbye, Kernel!\n");
}
```

// Register init and exit functions

    module\_init(hello\_init);

    module\_exit(hello\_exit);

**To build and load this module:**

Save it as hello.c. .1

Create a Makefile. .2

makefile

CopyEdit

obj-m += hello.o

all:

make -C /lib/modules/\$(shell uname -r)/build M=\$(PWD) modules

clean:

make -C /lib/modules/\$(shell uname -r)/build M=\$(PWD) clean

Run make to compile. .3

Use sudo insmod hello.ko to insert the module. .4

Use dmesg | tail to check the kernel messages. .5

Use sudo rmmod hello to remove the module. .6

Let me know if you want to expand it with parameters or logging levels!

**You said:**

Write the make file like this and tell me step by step what to do:  
ifneq (\$(KERNELRELEASE),) obj-m := hello\_kernel.o  
else  
KERNEL\_DIR := /lib/modules/\$(shell uname -r)/build  
PWD := \$(shell pwd)  
all: \$(MAKE) -C \$(KERNEL\_DIR) M=\$(PWD) modules  
clean \$(MAKE) -C \$(KERNEL\_DIR) M=\$(PWD) clean  
endif

**ChatGPT said:**

Here's your requested **Makefile** formatted with the conditional structure and indentation (make sure to use **tabs**, not spaces, for commands under targets like all and clean):

 **Makefile**

make

CopyEdit

ifneq (\$(KERNELRELEASE),)

    obj-m := hello\_kernel.o

else



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

all:

```
$(MAKE) -C $(KERNEL_DIR) M=$(PWD) modules
```

clean:

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

endif

---

## Step-by-Step Guide to Compile and Load the Kernel Module

### 1. Create your kernel module source file

Save this as hello\_kernel.c in a new folder:

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

```
MODULE_AUTHOR("Your Name");
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("A simple kernel module that prints Hello and Goodbye");
```

```
static int __init hello_init(void)
{
    printk(KERN_INFO "Hello, Kernel!\n");
    return 0;
}
```

```
static void __exit hello_exit(void)
{
    printk(KERN_INFO "Goodbye, Kernel!\n");
}
```



```
module_init(hello_init);
module_exit(hello_exit);
```

## 2. Create the Makefile

Place the Makefile you wrote in the same folder as hello\_kernel.c.

## 3. Build the module

Open a terminal in the module folder and run:

bash

CopyEdit

make

This compiles hello\_kernel.c into a loadable kernel module named hello\_kernel.ko.

## 4. Insert the module into the kernel

bash

CopyEdit

sudo insmod hello\_kernel.ko

## 5. Check the output

Use dmesg to see the kernel logs:

bash

CopyEdit

dmesg | tail

You should see:

CopyEdit

Hello, Kernel!

## 6. Remove the module

bash

CopyEdit

sudo rmmod hello\_kernel

## 7. Check the goodbye message

bash

CopyEdit

dmesg | tail

You should now see:

CopyEdit

Goodbye, Kernel!



## ✍ 8. Clean up build artifacts

bash

CopyEdit

make clean

Let me know if you want to add parameters or test multiple modules!

**You said:**

I am working with WSL and I have ubuntu on it on my windows 11, when I maked it said this: kasra@Kasra:~/P4\$ make make -C /lib/modules/5.15.153.1-microsoft-standard-WSL2/build M=/home/kasra/P4 modules make[1]: \*\*\* /lib/modules/5.15.153.1-microsoft-standard-WSL2/build: No such file or directory. Stop. make: \*\*\* [Makefile:9: all] Error 2

**ChatGPT said:**

Ah, got it — you're running this on **WSL2 (Windows Subsystem for Linux)**. That error is because the **Linux kernel headers are missing** — and **WSL doesn't support building kernel modules by default**.

### 🔧 Why this happens:

In regular Linux, `/lib/modules/$(uname -r)/build` is a symlink to the kernel source or headers needed to build modules. But:

- WSL2 uses a **Microsoft-customized kernel**.

- It's **not a full Linux kernel environment** — it lacks kernel headers, and you **cannot compile kernel modules** directly in WSL.

### Your Options:

#### Option 1: Use a real Linux VM (Recommended for kernel dev)

- Install **Ubuntu** in **VirtualBox**, **VMware**, or **WSL with a real kernel via systemd-nspawn**.

- Inside that, install:

bash

CopyEdit

`sudo apt update`

`sudo apt install build-essential linux-headers-$(uname -r)`

Then your make will work, since proper kernel headers are available.

#### Option 2: Dual Boot or External Boot into Linux

If you're doing more low-level Linux development (e.g., kernel modules, device drivers), it's easier to work directly in an actual Linux boot (or on a live USB).

#### Option 3: Use WSL2 with a Custom Kernel (Advanced)

**⚠ This requires:**

Building a custom WSL kernel with module support. •

Pointing WSL to use it (.wslconfig). More info: <https://learn.microsoft.com/en-us/windows/wsl/build-custom-kernel>

### TL;DR:

You **cannot compile kernel modules inside WSL2** by default because it doesn't expose the kernel for building modules.

### Next Steps:

If you want help setting up Ubuntu in a VM (like VMware or VirtualBox), or customizing WSL for kernel module support, I can guide you through it.

Want to go that route?

### **You said:**

I will switch to my VM and tell you what will happen there.

### **ChatGPT said:**

Perfect — that's a good move for kernel development. Once you're in your Ubuntu VM:

#### Do this before running make:

bash

CopyEdit

sudo apt update

sudo apt install build-essential linux-headers-\$(uname -r)

Then head to your module directory and run:

bash

CopyEdit

make

Once you've tried it out, let me know what happens — if you hit any errors or weird messages, just paste them here and I'll help you troubleshoot.

### **You said:**

I've entered the sudo appts you said before running make, when I entered make it said: Makefile:9: \*\*\* missing separator.  
Stop.

### **ChatGPT said:**

Ah yes, that classic missing separator error in Makefiles — it means **you used spaces instead of a TAB** for one of the command lines in your Makefile.

### Fix it:

In your Makefile, **make sure the lines under all: and clean: are indented with an actual tab, not spaces.**

Here's how it should look — copy this carefully and make sure to use a **real TAB character** before the \$(MAKE) commands:

make

CopyEdit



```
ifneq ($(KERNELRELEASE),)
    obj-m := hello_kernel.o
```

```
else
    KERNEL_DIR := /lib/modules/$(shell uname -r)/build
    PWD := $(shell pwd)
```

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

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

```
endif
```

#### Quick test:

You can check if you're using tabs by running:

```
bash
```

```
CopyEdit
```

```
cat -A Makefile
```

Tabs will show as ^I. If you see spaces instead, fix those lines.

#### Tip:

Most editors (like nano, vim, VSCode) can be set to insert tabs instead of spaces — if you're using VSCode:

- Hit Ctrl+, for settings

- Search for "**insert spaces**"

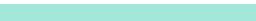
- Uncheck it, and make sure "**Tab Size**" = 4

Let me know once you've fixed it or want me to generate a fresh, clean Makefile for you.

#### **You said:**

Is it ok now? : kasra@kasra-None:~/Desktop/P4\$ make make -C /lib/modules/6.5.0-44-generic/build  
M=/home/kasra/Desktop/P4 modules make[1]: Entering directory '/usr/src/linux-headers-6.5.0-44-generic' warning: the compiler differs from the one used to build the kernel The kernel was built by: x86\_64-linux-gnu-gcc-13 (Ubuntu 13.2.0-

4ubuntu3) 13.2.0 You are using: gcc-13 (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0 CC [M]  
/home/kasra/Desktop/P4/hello\_kernel.o MODPOST /home/kasra/Desktop/P4/Module.symvers CC [M]  
/home/kasra/Desktop/P4/hello\_kernel.mod.o LD [M] /home/kasra/Desktop/P4/hello\_kernel.ko BTF [M]  
/home/kasra/Desktop/P4/hello\_kernel.ko Skipping BTF generation for /home/kasra/Desktop/P4/hello\_kernel.ko due to unavailability of vmlinux make[1]: Leaving directory '/usr/src/linux-headers-6.5.0-44-generic'

ChatGPT said:

Yes! 🎉 That output means your **kernel module compiled successfully!** 🎉

 Let's break it down:

- The warning about the compiler version mismatch is **not critical** — it's just telling you that your kernel was built with GCC 13.2 and you're compiling with GCC 13.3. That's usually fine as long as the major version matches (which it does). •

- The hello\_kernel.ko file was successfully built. •

- The "Skipping BTF generation" is also not fatal — it just means debugging info (BTF) won't be embedded unless you have the full vmlinux file. You can ignore this for now unless you're doing deep tracing or eBPF work. •

 **Next Steps:**

Now you can load and test the module:

bash

CopyEdit

# Insert the module

sudo insmod hello\_kernel.ko

# Check kernel logs

dmesg | tail

You should see:

CopyEdit

Hello, Kernel!

Then, to remove the module:

bash

CopyEdit

sudo rmmod hello\_kernel

dmesg | tail

You should see:

CopyEdit

Goodbye, Kernel!

Let me know if you want to try passing parameters to your module, logging at different levels, or auto-loading at boot!

You said:

I've done the things you said, is it ok? [ 737.841693] workqueue: blk\_mq\_run\_work\_fn hogged CPU for >10666us 4 times, consider switching to WQ\_UNBOUND [ 1164.022126] hello\_kernel: loading out-of-tree module taints kernel. [ 1164.022184] hello\_kernel: module verification failed: signature and/or required key missing - tainting kernel [ 1164.023759] Hello, Kernel! [

1215.458406] workqueue: blk\_mq\_run\_work\_fn hogged CPU for >10666us 8 times, consider switching to WQ\_UNBOUND [1240.549086] Goodbye, Kernel!

**ChatGPT said:**

Yes — that looks **perfect!** 🎉 Your kernel module is working exactly as expected.

 **Let's go through the important lines:**

**These are expected and confirm success:**

csharp

CopyEdit

[1164.023759] Hello, Kernel!

[1240.549086] Goodbye, Kernel!

These are your printk() messages from the module's `__init` and `__exit` functions. Awesome!

 **⚠ These are not errors, just informational:**

vbnnet

CopyEdit

hello\_kernel: loading out-of-tree module taints kernel.

hello\_kernel: module verification failed: signature and/or required key missing - tainting kernel

These simply mean:

Your module wasn't signed (normal for learning/dev). •

It's "tainting" the kernel — which **just flags that non-standard code is running**. Totally fine during development. •

 **☒ These are unrelated warnings:**

makefile

CopyEdit

workqueue: blk\_mq\_run\_work\_fn hogged CPU for >10666us ...

These are from your disk I/O system and have nothing to do with your module. They're often harmless unless you're doing performance tuning on block devices.

**☒ Conclusion:**

You successfully:

Built a kernel module •

Inserted it into the kernel •

Verified it ran •



Cleanly removed it •

🔥 You're on fire! Want to try making a module that takes parameters next?