}

Writing Linux kernel module is a daunting task. This easy to follow, step by step guide will show us all the steps of writing Linux kernel module. Read this writing to learn about what is a kernel module.

This tutorial will be our starting point for writing Linux kernel module. For this one we won't be using any real hardware. But gradually we'll learn how to write a device driver for an actual hardware. Our approach will start from learning most basic module to advanced device driver.

Preparation:

Before writing Linux kernel module, at first we've to prepare our system for compiling the module code. We've to install kernel headers. Kernel headers are header file for kernel module. Without the header files, our module won't recognize the kernel functions. We can install the kernel headers using following command.

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

Writing the code:

Now lets start with this simple code. Save this code in a file and name it *test_module.c*

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

static int test_module_init(void)
{
    printk("Test Module Installedn")

    return 0;
}

static void test_module_exit(void)
```

```
{
    printk("Test Module Removedn")
}

module_init(test_module_init);
module_exit(test_module_exit);

MODULE_AUTHOR("IsonProjects");
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Simple Kernel Module");
```

Now lets see a step by step illustration of what's going on in this code. The macros at the end of the code **MODULE_AUTHOR, MODULE_LICENSE, MODULE_DESCRIPTION** just gives information about the author of the module, the license of the module and some description.

Every kernel module must have an initialization function and an exit function. The initialization function is the entry point of the kernel module where we should initialize all data structure that are required for kernel function. The exit function is the exit point of the kernel where we can release the allocated resources. In modern kernel we can name the initialization and exit function anything we want, provided it doesn't violate normal function naming guideline. In the above code, our initialization function is ***test_module_init*** and exit function is ***test_module_exit***. But the kernel doesn't know which one is for initialization and which one is for exit. We've to let the kernel know about each function. This is done with **module_init** and **module_exit**. **module_init** will declare which function is initialization function(line 15) and **module_exit** will declare the exit function (line 16).

Write Makefile:

Next step is to create a **Makefile** for building our kernel module. Write the following code in a file and save it with the name **Makefile**. We'll use this same **Makefile** or a little modification of it in our other tutorials. Note that the first line of the **Makefile** contains the name of the source file with **.o** extension.

```
obj-m := test_module.o
```

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

all:
	$(MAKE) -C $(KERNELDIR) M=$(PWD)

clean:
	rm -f *.o *~ core .depend .*.cmd *.ko *.mod.c
	rm -rf .tmp_versions
	rm Module.symvers
	rm modules.order
```

Compile & Build:

Next step is to actually build the kernel module from the source. Keep both **test_module.c** and **Makefile** in the same directory. Open a terminal and go to that directory and run the ***make*** command from the terminal. This will compile the kernel module source code and create the binary. Multiple files will be created inside that directory. Look for the one with a **.ko** extension. This is the desired binary that we've to install. In our case the created binary name will be **test_module.ko**.

Install the binary:

Now we've to install the created binary. We'll use **insmod** command for installing the module. This command requires *root* permission. We execute the following command.

```
sudo insmod test_module.ko
```

This command will install the kernel module in the system.

Testing:

So far we've written the kernel module, compiled it, created the executable binary and installed it in the system. Now we need to test that it actually worked. Notice in the module initialization function we have a **printk** function call. This function will print the string in the kernel log. We can use the ***dmesg*** command to check the kernel log. In the terminal execute ***dmesg*** command and you'll see a bunch of text is displayed on the terminal. Look at the bottom of the text. If our kernel module installation was successful, we should see the text “**Test Module Installed**” in the log.

Uninstall the module:

To uninstall the kernel module we use the ***rmmod*** command as below

```
sudo rmmod test_module.ko
```

This will remove the module from the system. Run the ***dmesg*** command and you’ll see the text “**Test Module Removed**” in the kernel log.