

Lab 9: Introduction to Device drivers

The kernel module is the logical separate functionality which can be added at a later stage in the running kernel. This extends the Linux kernel features/functions.

These modules are files with the extension as “.ko” , which means kernel objects. If we come across any file with “.ko” as an extension, it clearly indicates that it is a kernel module.

Example - hello-world.ko, iosample.ko

The process of adding a kernel module to the kernel is also known as the loading of a kernel module. The kernel module can only be loaded if it is compatible with the kernel source tree. This means the kernel module is not platform-independent. These can be loaded only on the kernel with the same source tree and build configuration.

Make sure you have the required development tools:

Run the command –

```
sudo apt update
```

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

Creating the Module Directory – Open a terminal and create a new directory for your module:

```
mkdir ~/hello_world_module
```

```
cd ~/hello_world_module
```

Write the Hello world program module -

Hello world program: hello.c

```
// hello_world.c
```

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

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

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

```
MODULE_AUTHOR("Your Name");
```

```
MODULE_DESCRIPTION("A simple Hello World device driver");
```

```
static int __init hello_world_init(void) {
```

```
    printk(KERN_INFO "Hello, World!\n");
```

```
    return 0; // Non-zero return means that the module couldn't be  
loaded.
```

```
}
```

```
static void __exit hello_world_exit(void) {
```

```
    printk(KERN_INFO "Goodbye, World!\n");
```

```
}
```

```
module_init(hello_world_init);
```

```
module_exit(hello_world_exit);
```

Create the file named Makefile in the same directory -

Save the file and create a makefile, Makefile

```
# Makefile
```

```
obj-m += hello_world.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
```

Save the Makefile.

Steps to execute the kernel file.

Now in the terminal, run:

1. Now execute the program using make command. Use `$ make`
2. List the files created after executing make command. Use `$ ls`
hello.c hello.mod.o hello.ko hello.o hello.mod

Load the Module in the Kernel

Linux provides a utility known as “insmod”. This is the utility which can be used to load the kernel module at the running kernel.

To load the kernel module, just execute the command as insmod followed by the module file name.

Use `$ sudo insmod hello.ko`

With the above command’s successful execution, our module is added to the Linux kernel. To check whether the module is loaded.

Use `lsmod`. This command will list currently loaded kernel modules on your system. If you want to find a specific module. This can be network driver module (e1000) then you can do via `grep` command.

```
$ lsmod
```

```
$ lsmod | grep hello
```

Module	Size	Usedby
hello	16384	0
.		
.		
.		

If you want to see the information regarding the modules, the user has created. Use `$ modinfo hello.ko`

`modinfo` stand for 'module information'. This command will show the information about a kernel module.

To check the message "hello-world", use the `dmesg` command. If the command results in no error message, then it can be considered module addition is successful. As we are extending the kernel functions, so it requires super user privileges to execute the "`insmod`".

```
$ sudo dmesg | tail
```

```
Hello world!!!
```

Removing the Kernel module

To remove or unload the kernel module, we can use the Linux command "`rmmod`". This is the command which is used to remove the loaded kernel module and listed by `lsmod`.

```
$ sudo rmmod hello
```

```
$ dmesg | tail
```

```
Goodbye, Kernel
```

Exercises –

1. Write a Linux character device driver with read and write operations.
2. Write a Linux kernel module to implement a simple pseudo-random number generator as a character device.
3. Write a simple Linux device driver to create a character device.