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.

Hello world program: hello.c

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

/* Meta information */
MODULE_LICENSE(“GPL”);
MODULE_AUTHOR(“Linux ubuntu”);
MODULE_DESCRIPTION(“A hello world LKM”);

/* Load the module into the kernel */

static int __init ModuleInit(void)
{
    printk(“Hello world!\n”);
    return 0;
}

/* Remove the module from the kernel */

static void __exit ModuleExit(void)
{
    printk(“Good Bye, Kernel !\n”);
}
```

Save the file and create a makefile, Makefile

```
obj-m+= hello.o
str=`uname -r`
all:
	make -C /lib/modules/$(str)/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(str)/build M=$(PWD) clean
```

Save the Makefile.

Steps to execute the kernel file.

Now in the terminal,

1. Check the root. Use $ uname -r
6.2.0-36-generic

2. Change the root. Use $ su prg
Password:
prg@prg:/home/student$

3. Change the directory. Use $ cd ~
prg@prg:~$

4. Copy the file from student to super user directory. Use cp /home/student/hello.c . and cp /home/student/Makefile .

5. Now execute the program using make command. Use $ make

6. 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
[sudo] password for prg:

prg@prg:~$

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.

```
prg@prg:~$ lsmod
prg@prg:~$ 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".

prg@prg:~$ 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.

```
prg@prg:~$ sudo rmmod hello
prg@prg:~$ dmesg | tail
```

Goodbye, Kernel