Mini-project: Adding A Kernel Module

Aim:

To add a kernel module to an existing base kernel.

Theory:

What is a Kernel?

The kernel is the central module of an operating system (OS). It is the part of the operating system that loads first, and it remains in main memory. Because it stays in memory, it is important for the kernel to be as small as possible while still providing all the essential services required by other parts of the operating system and applications. The kernel code is usually loaded into a protected area of memory to prevent it from being overwritten by programs or other parts of the operating system.
Typically, the kernel is responsible for memory management, process and task management, and disk management. The kernel connects the system hardware to the application software. Every operating system has a kernel. For example, the Linux kernel is used by numerous operating systems including Linux, FreeBSD, Android, and others.

What is a Kernel Module?

Kernel modules are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without the need to reboot the system.
To create a kernel module, you can read The Linux Kernel Module Programming Guide. A module can be configured as built-in or loadable. To dynamically load or remove a module, it has to be configured as a loadable module in the kernel configuration (the line related to the module will, therefore, display the letter M).
The kernel modules will have a .ko extension. On a normal Linux system, the kernel modules will reside inside /lib/modules/<kernel_version>/kernel/ directory.

Some Important Commands

Modules are stored in /usr/lib/modules/*kernel_release*. You can use the command uname -r to get your current kernel release version.

To show what kernel modules are currently loaded:

```
$ lsmod
```

To show information about a module:

```
$ modinfo module_name
```

To list the options that are set for a loaded module:

```
$ systool -v -m module_name
```

To display the comprehensive configuration of all the modules:

```
$ modprobe -c | less
```

To display the configuration of a particular module:

```
$ modprobe -c | grep module_name
```

List the dependencies of a module (or alias), including the module itself:

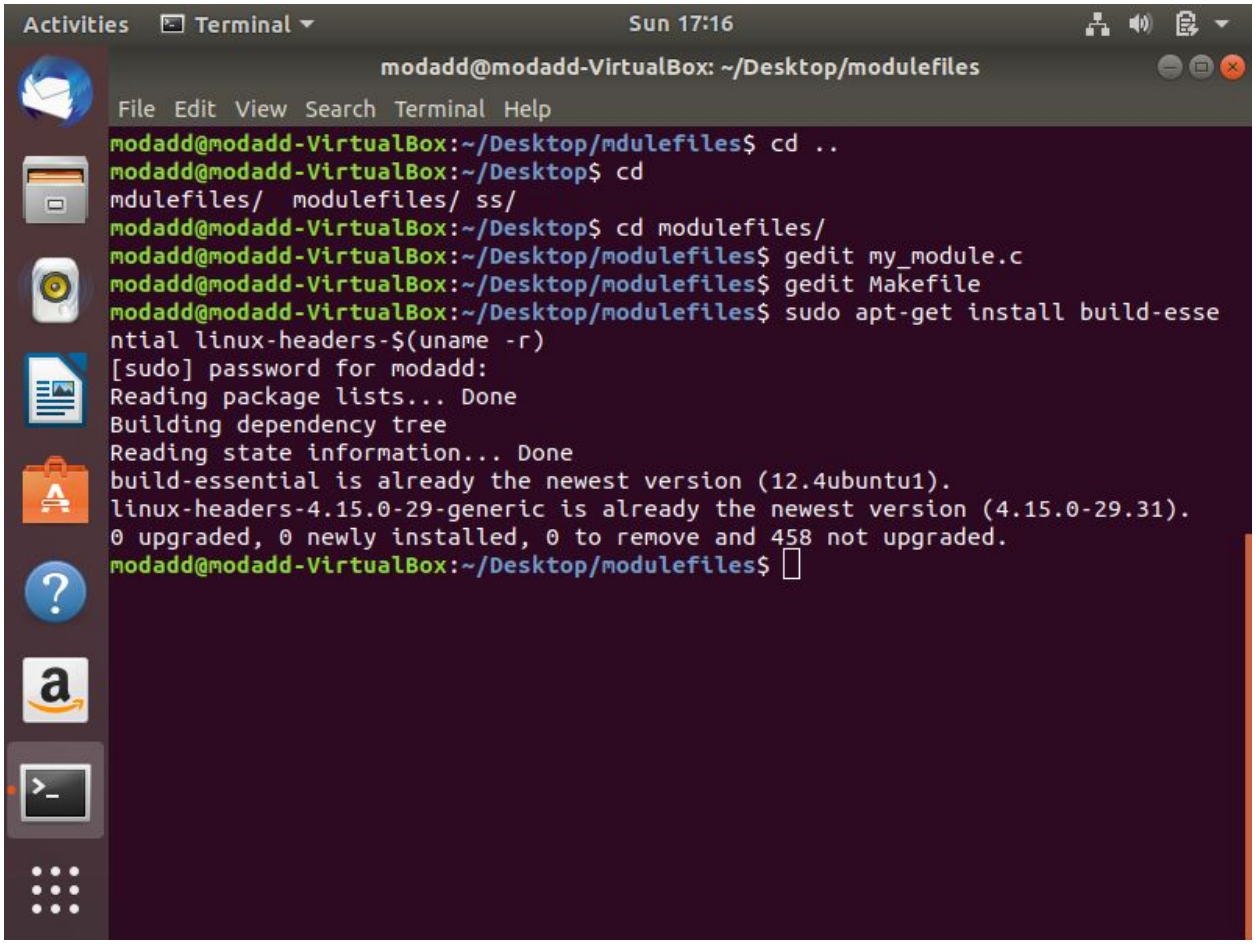
```
$ modprobe --show-depends module_name
```

*The above Commands will be used to step-by-step while the module addition.

Procedure:

1. Installing the linux headers

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



2. Kernel Module Source Code

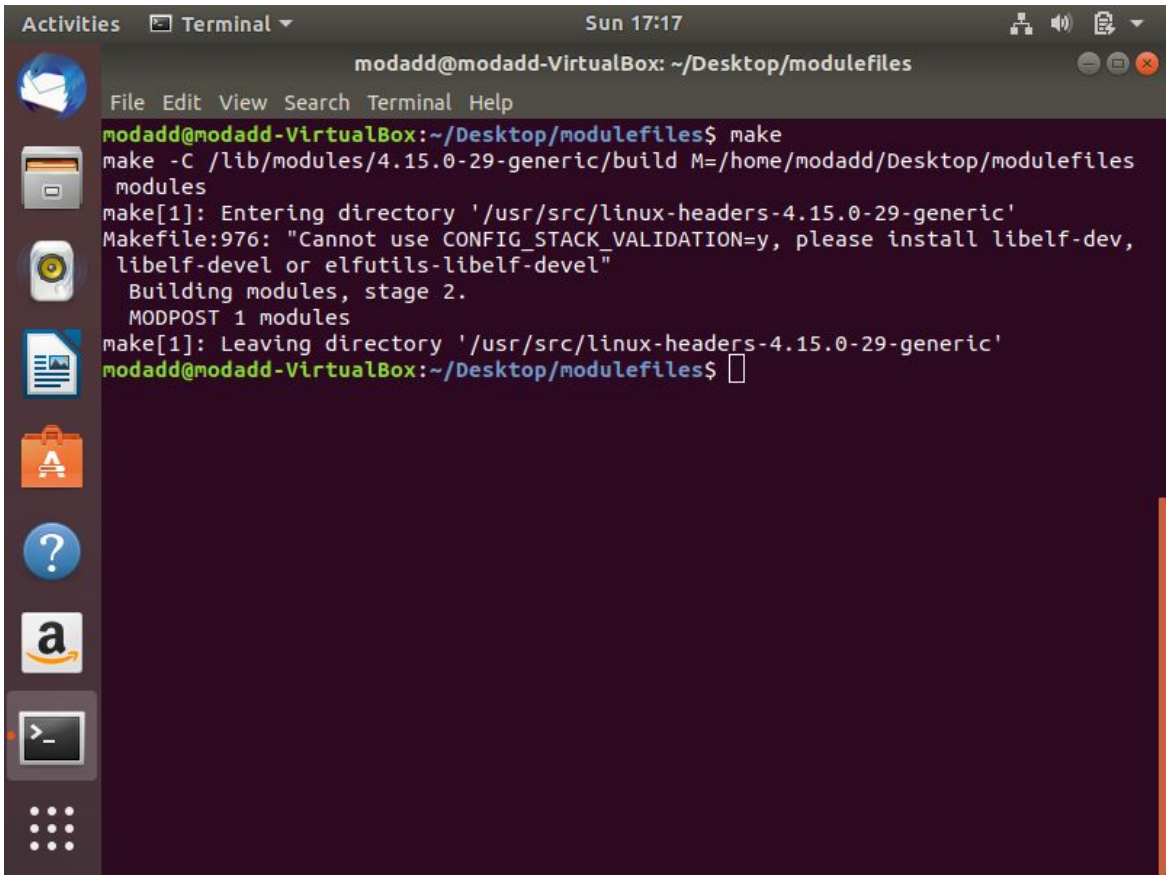
```c
#include <linux/module.h>    // included for all kernel modules
#include <linux/kernel.h>    // included for KERN_INFO
#include <linux/init.h>      // included for __init and __exit macros

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Author");
MODULE_DESCRIPTION("Test module for addition");

static int __init newmod_init(void) {
    printk(KERN_INFO "NEW MODULE IS INSERTED\n");
    return 0;    // Non-zero return means that the module couldn't be
loaded.
}
```

```
static void __exit newmod_cleanup(void) {
    printk(KERN_INFO "MODULE REMOVED\n");
}
module_init(newmod_init);
module_exit(newmod_cleanup);
```

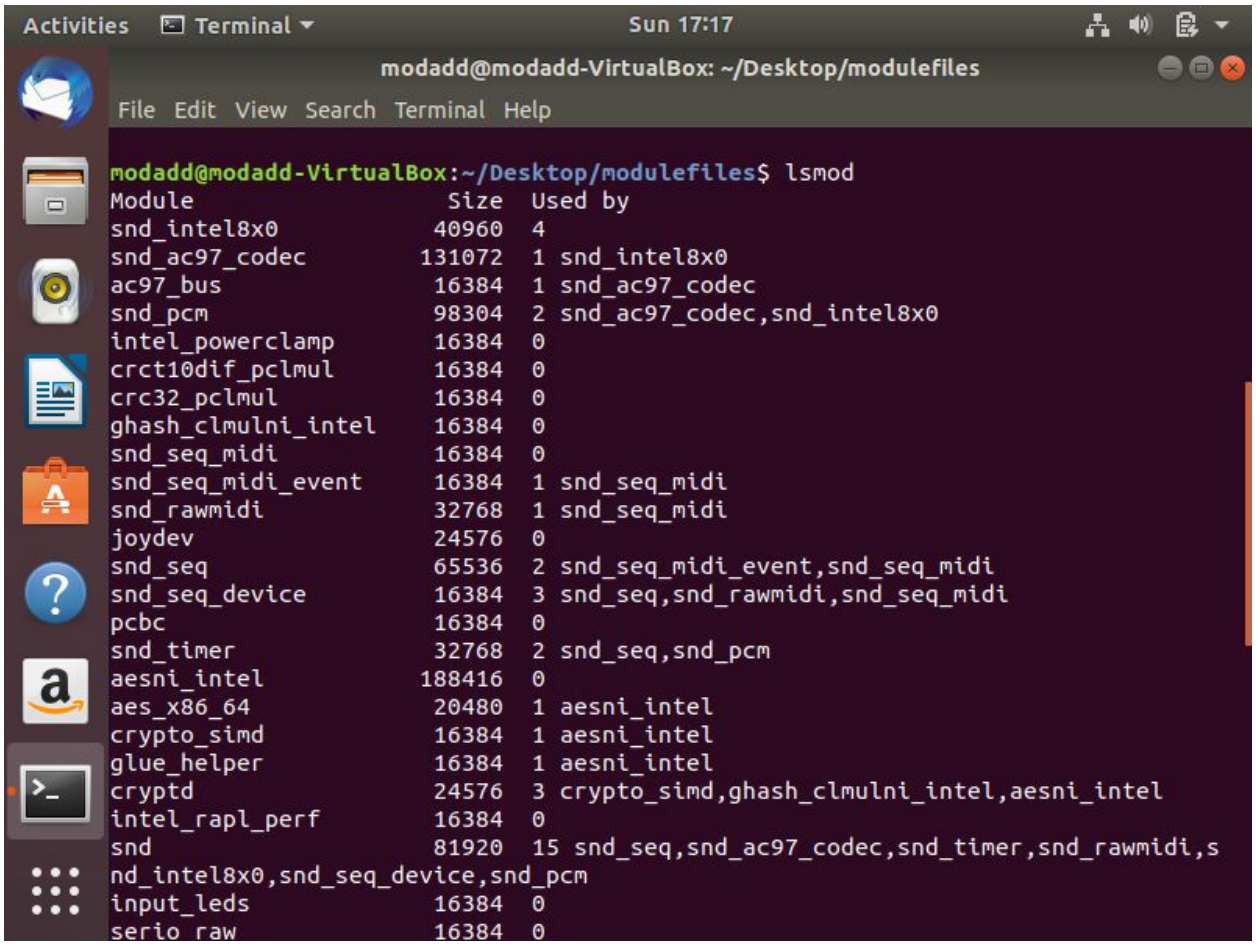
3. Create Makefile to Compile Kernel Module

```
obj-m += my_module.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
```

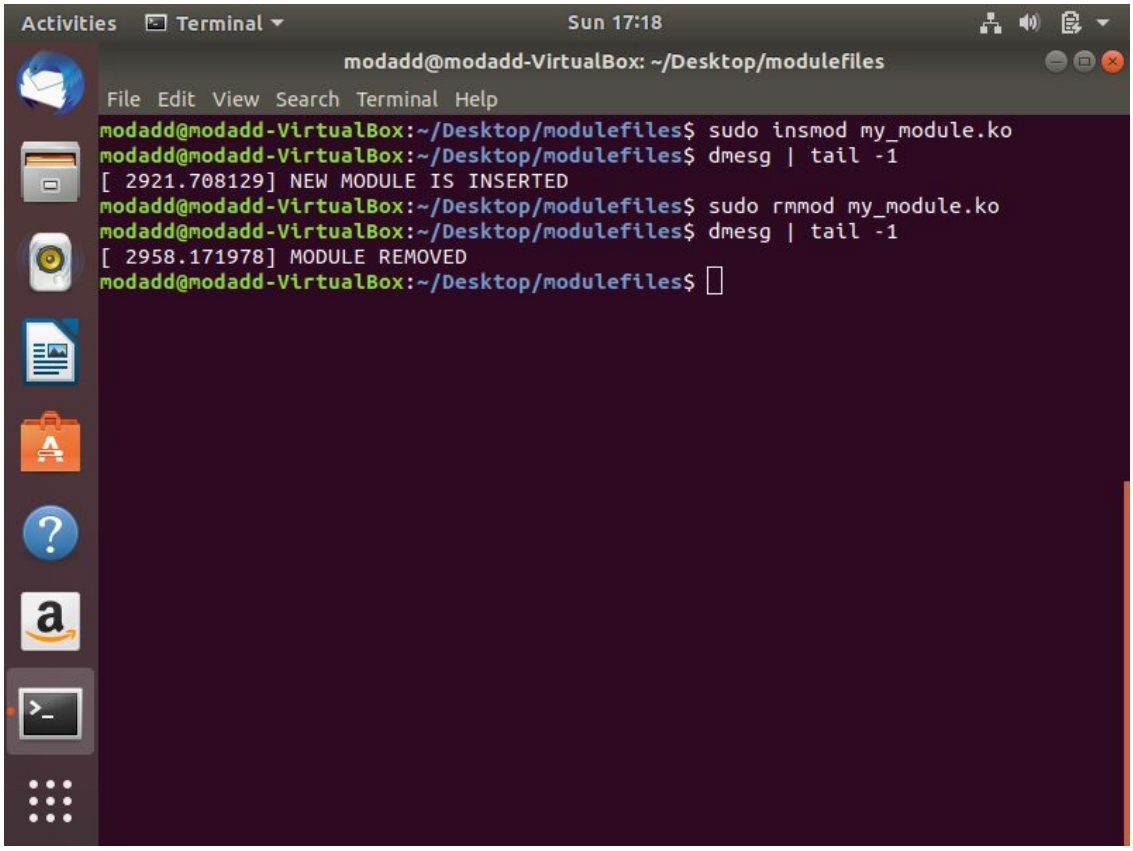
4. make command executed for module compilation and .ko file is created.



5. Lets see the current kernel modules



6.Look at the initialization message stored in the kernel message buffer after insertion of module(insmod) ,also after the removal of the module(rmmod).



Conclusion:

A kernel object file is created when make file is compiled and a kernel object file is generated. This is the module that is inserted using the insmod command and the init macro is invoked and a message is added to the buffer similarly during rmmod; exit macro is executed and the message is added to the kernel buffer.
Thus, following the above concepts, I added a Kernel module to the existing OS module.