Ahmed Mohamed AbdElmoneam Mohamed

احمد محمد عبد المنعم محمد

3rd Year CS Department

الفرقة الثالثة حاسبات

سكشن ١

Steps

I have designed two kernel modules as required for the project. The first one which reports *jiffies* after viewing the proc file using *cat*. And the second one which reports the seconds elapsed since the kernel module was loaded.

In the first module design I have included the required libraries at first to my code. Then, I created *proc_init* and *proc_exit* to be my module entry and exit points. Finally, in *proc_read* I have printed *jiffies* to the user space buffer as required.

In the second kernel design I have included the required libraries to use *jiffies* and *HZ* and I have used 3 unsigned long variables to calculate the seconds elapsed since the kernel module was loaded. As I have subtracted the *jiffies* when the module was loaded from the *jiffies* when I used *cat* command and divided the total over *HZ* to get the number of seconds.

I have written some lines in the *Makefile* to compile the module and to generate the .ko file which will be inserted to the kernel.

Code

Task 1

<u>.c file</u>

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

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/proc_fs.h>

#include <linux/uaccess.h>

#include <linux/jiffies.h>


#define BUFFER_SIZE 128

#define PROC_NAME "jiffies"


ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count, loff_t *pos);


static struct file_operations proc_ops = {

        .owner = THIS_MODULE,

        .read = proc_read,

};
```

```
int proc_init(void){

	proc_create(PROC_NAME, 0666, NULL, &proc_ops);

	return 0;

}


void proc_exit(void){

	remove_proc_entry(PROC_NAME, NULL);

}


ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count, loff_t *pos){

	int rv = 0;

	char buffer[BUFFER_SIZE];

	static int completed = 0;

	if (completed){

		completed = 0;

		return 0;

	}

	completed = 1;

	rv = sprintf (buffer, "%lu \n", jiffies);

	copy_to_user(usr_buf, buffer, rv);

	return rv;

}


module_init(proc_init);
```

```
module_exit(proc_exit);


MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Hello Module");
MODULE_AUTHOR("SGG");
```

Makefile

```
obj-m += Task1.o


KDIR = /lib/modules/$(shell uname -r)/build


PWD = $(shell pwd)


all:
	$(MAKE) -C $(KDIR) M=$(PWD) modules
install:
	$(MAKE) -C $(KDIR) M=$(PWD) modules_install
%:
	$(MAKE) -C $(KDIR) M=$(PWD) $@
```

Task 2

.c file

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

#include <linux/kernel.h>

#include <linux/module.h>

#include <linux/proc_fs.h>

#include <linux/uaccess.h>

#include <linux/jiffies.h>

#include <asm/param.h>


#define BUFFER_SIZE 128

#define PROC_NAME "seconds"


unsigned long start = 0;

unsigned long end = 0;

unsigned long sec = 0;


ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count, loff_t *pos);


static struct file_operations proc_ops = {

	.owner = THIS_MODULE,

	.read = proc_read,

};
```

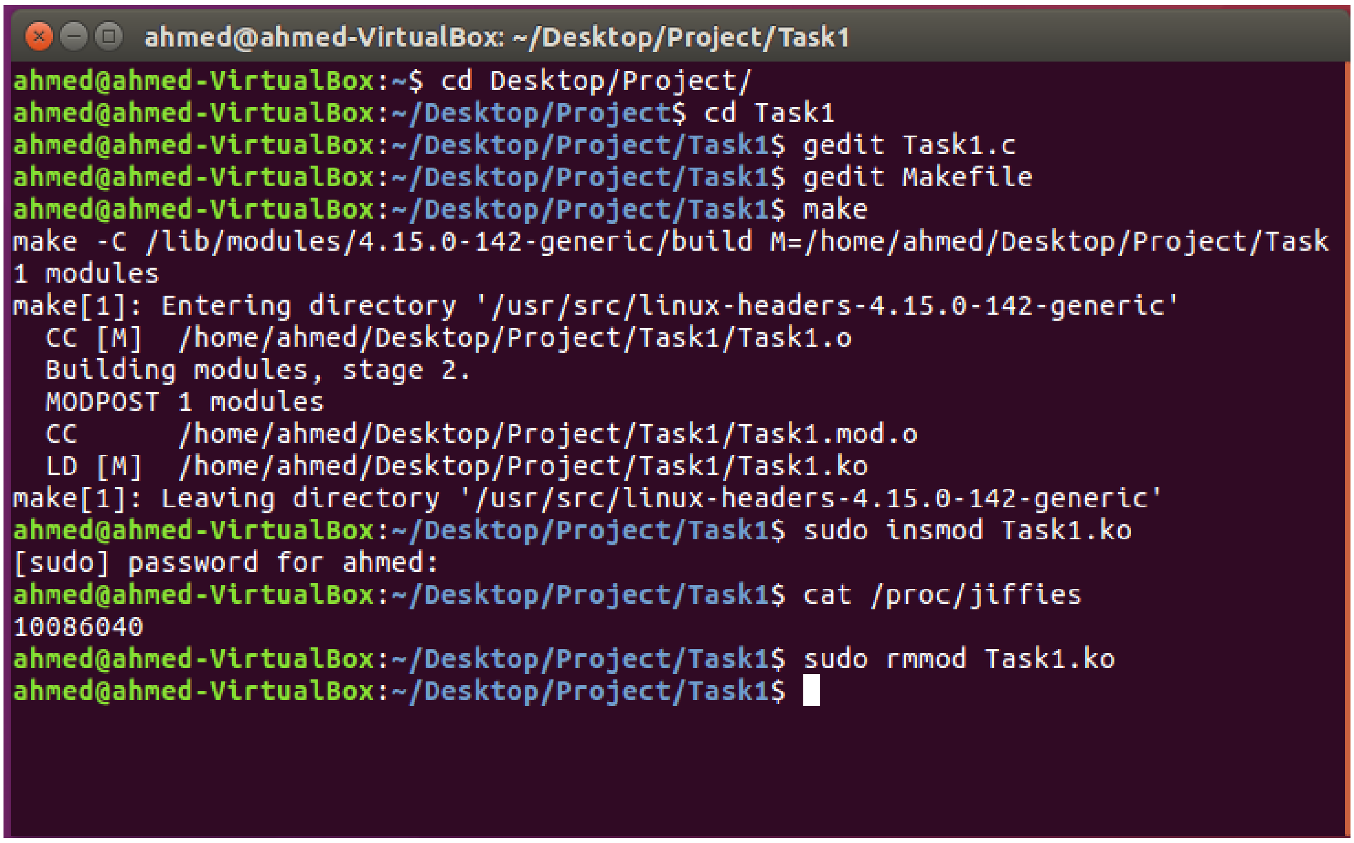
```
int proc_init(void){
        proc_create(PROC_NAME, 0666, NULL, &proc_ops);
        start = jiffies;
        return 0;
}

void proc_exit(void){
        remove_proc_entry(PROC_NAME, NULL);
}

ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count, loff_t *pos){
        int rv = 0;
        char buffer[BUFFER_SIZE];
        static int completed = 0;
        end = jiffies;
        sec = (end - start) / HZ;
        if (completed){
                completed = 0;
                return 0;
        }
        completed = 1;
        rv = sprintf (buffer, "%lu \n", sec);
        copy_to_user(usr_buf, buffer, rv);
        return rv;
}
```

```
module_init(proc_init);
module_exit(proc_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Hello Module");
MODULE_AUTHOR("SGG");
```

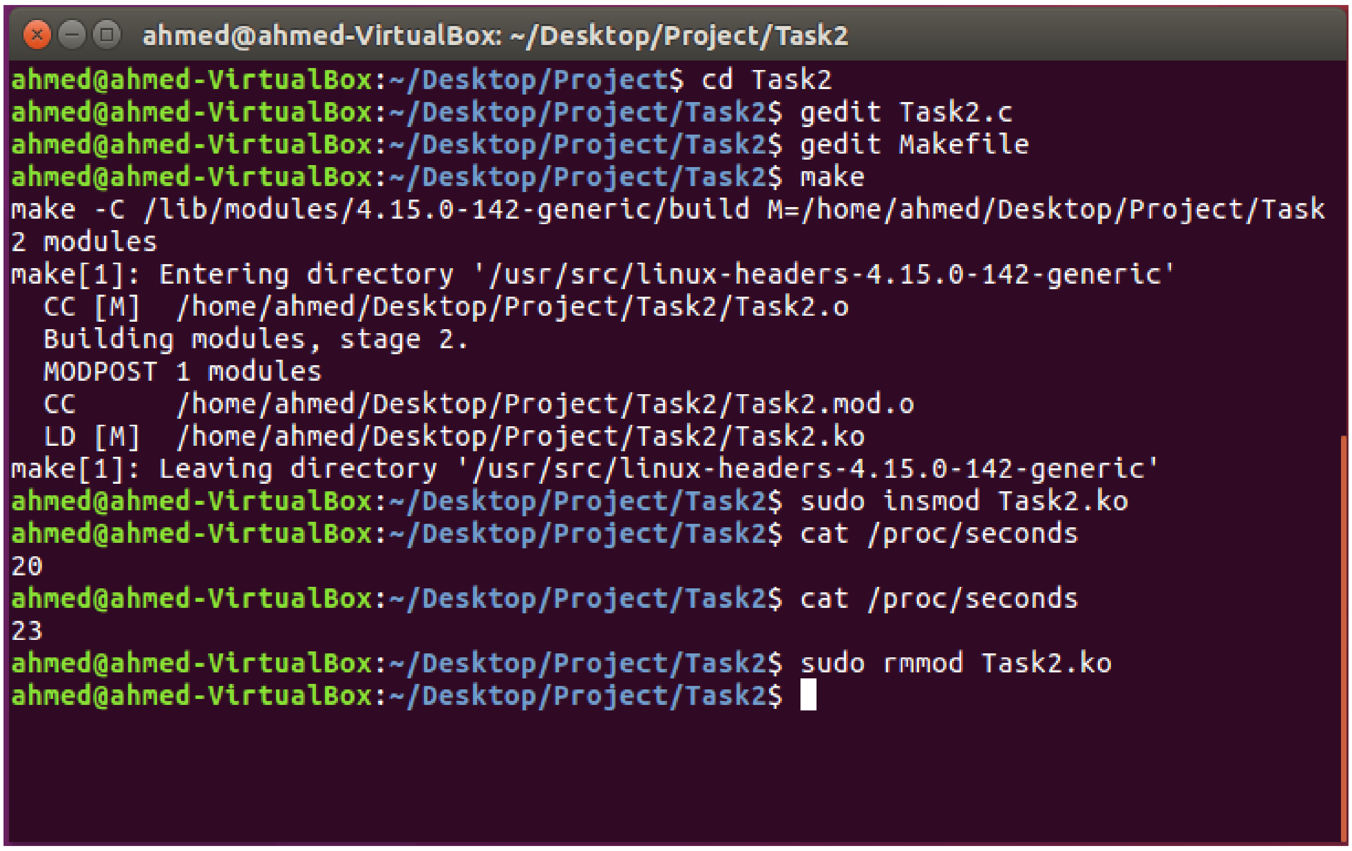
Makefile

```
obj-m += Task2.o

KDIR = /lib/modules/$(shell uname -r)/build

PWD = $(shell pwd)

all:
	$(MAKE) -C $(KDIR) M=$(PWD) modules
install:
	$(MAKE) -C $(KDIR) M=$(PWD) modules_install
%:
	$(MAKE) -C $(KDIR) M=$(PWD) $@
```

Screen Shots



```
ahmed@ahmed-VirtualBox:~$ cd Desktop/Project/
ahmed@ahmed-VirtualBox:~/Desktop/Project$ cd Task1
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task1$ gedit Task1.c
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task1$ gedit Makefile
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task1$ make
make -C /lib/modules/4.15.0-142-generic/build M=/home/ahmed/Desktop/Project/Task
1 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-142-generic'
  CC [M]  /home/ahmed/Desktop/Project/Task1/Task1.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/ahmed/Desktop/Project/Task1/Task1.mod.o
  LD [M]  /home/ahmed/Desktop/Project/Task1/Task1.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-142-generic'
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task1$ sudo insmod Task1.ko
[sudo] password for ahmed:
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task1$ cat /proc/jiffies
10086040
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task1$ sudo rmmod Task1.ko
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task1$ 
```



```
ahmed@ahmed-VirtualBox:~/Desktop/Project$ cd Task2
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task2$ gedit Task2.c
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task2$ gedit Makefile
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task2$ make
make -C /lib/modules/4.15.0-142-generic/build M=/home/ahmed/Desktop/Project/Task
2 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-142-generic'
  CC [M]  /home/ahmed/Desktop/Project/Task2/Task2.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/ahmed/Desktop/Project/Task2/Task2.mod.o
  LD [M]  /home/ahmed/Desktop/Project/Task2/Task2.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-142-generic'
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task2$ sudo insmod Task2.ko
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task2$ cat /proc/seconds
20
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task2$ cat /proc/seconds
23
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task2$ sudo rmmod Task2.ko
ahmed@ahmed-VirtualBox:~/Desktop/Project/Task2$ 
```

Key Points

- I have learnt how to design a kernel module and how to insert it into the kernel.
- I have identified the difference between *printf*, *printk* and *sprintf* while coding in *C*.
- I dealt with */proc* file system and I created */proc* files to solve my assignments.
- I have included many important *Linux* libraries to enable me to deal with kernel.
- I made use of *jiffies* and *HZ* to design a module that calculates the number seconds elapsed since a module is loaded into a kernel.