# Operating Systems – I (CS3510)

## *Assignment – II (Loading and Unloading Kernel modules)*

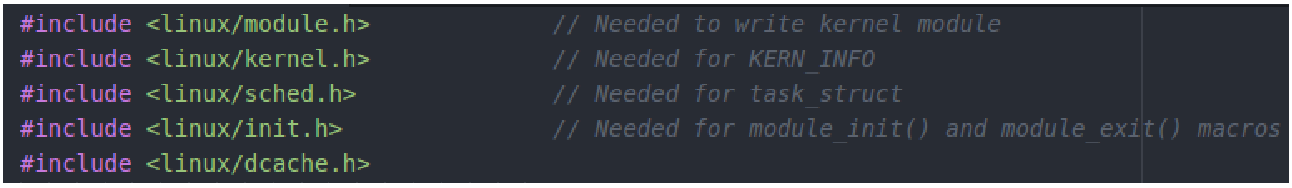
**Aim** – *To write a Linux kernel module that displays a list of all current running tasks in a Linux system. (Alternative command - “ps -el” on Linux Bash terminal).*

**Kernel Module** : *They are pieces of code that can be loaded and unloaded into the kernel upon demand. They extend the functionality of the kernel without rebooting the system.*

**Program Skeleton :**

```
#include “header files”

module_init() function definition {

        for_each_process(p)
                print task_details;

}
module_exit() function definition {

        print exit message;

}

module_init();
module_exit();

module_license()
module_author()
module_description()
```

Header files included :



```
#include <linux/module.h>           // Needed to write kernel module
#include <linux/kernel.h>           // Needed for KERN_INFO
#include <linux/sched.h>            // Needed for task_struct
#include <linux/init.h>             // Needed for module_init() and module_exit() macros
#include <linux/dcache.h>
```

**“linux/module.h”** is needed to write the code into module entry point.

**“kernel.h”** is required for writing optional prefix strings called **Loglevel** inside the **printk** function like *KERN_INFO*, *KERN_CRIT*, *KERN_ALERT*, for informational, critical conditions report and prompt for action must be taken immediately purposes.

**“sched.h”** has the macro defined for “**for_each_process**”, as well as the **struct – task_struct**, defined in it.

**“init.h”** has the definitions of the macros **module_init(f)** and **module_exit(fe)** as given below.

## INIT and CLEANUP functions

We can rename the init and cleanup functions of our kernel module. This is done with the **module_init()** and **module_exit()** macros. These macros are defined in **linux/init.h**.

We define our init and cleanup functions before calling the macros, as **start_module()** and **end_module()**, in order to avoid any compilation errors.

## Loading the Kernel module into the Linux kernel

When **module_init()** is invoked (upon running **sudo insmod ./*.ko**), the code written in the function **start_module()** gets executed, where using the macro “**for_each_process(struct task_struct *p)**” , we print each task’s name (executable name), its PID, and the state associated with it.

**for_each_process()** macro allows iteration over all current tasks in the system. We use printk function to print these contents in the kernel log buffer. We can specify optional ***Loglevel*** strings to provide special status to the content being printed. The default if not specified is *KERN_WARNING*. *KERN_INFO* is the one we are providing for printing welcome and exit messages at the time of module loading and unloading respectively.

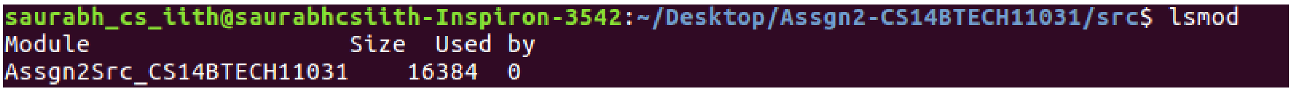
## Unloading the Kernel module from the Linux kernel

When the user wants to remove this loaded module into the kernel by running the command **sudo rmmod *modulename*** (which *unloads* the kernel module from the Linux kernel), the function called in the macro **module_exit()** will be executed, at module removal time. Therefore, to print an exit message, we define a function “**end_module()**”, which is invoked at the time of unloading the kernel module/invoking of the macro **module_exit()**.

The argument to both **start_module()** and **end_module()** functions is *void* here.

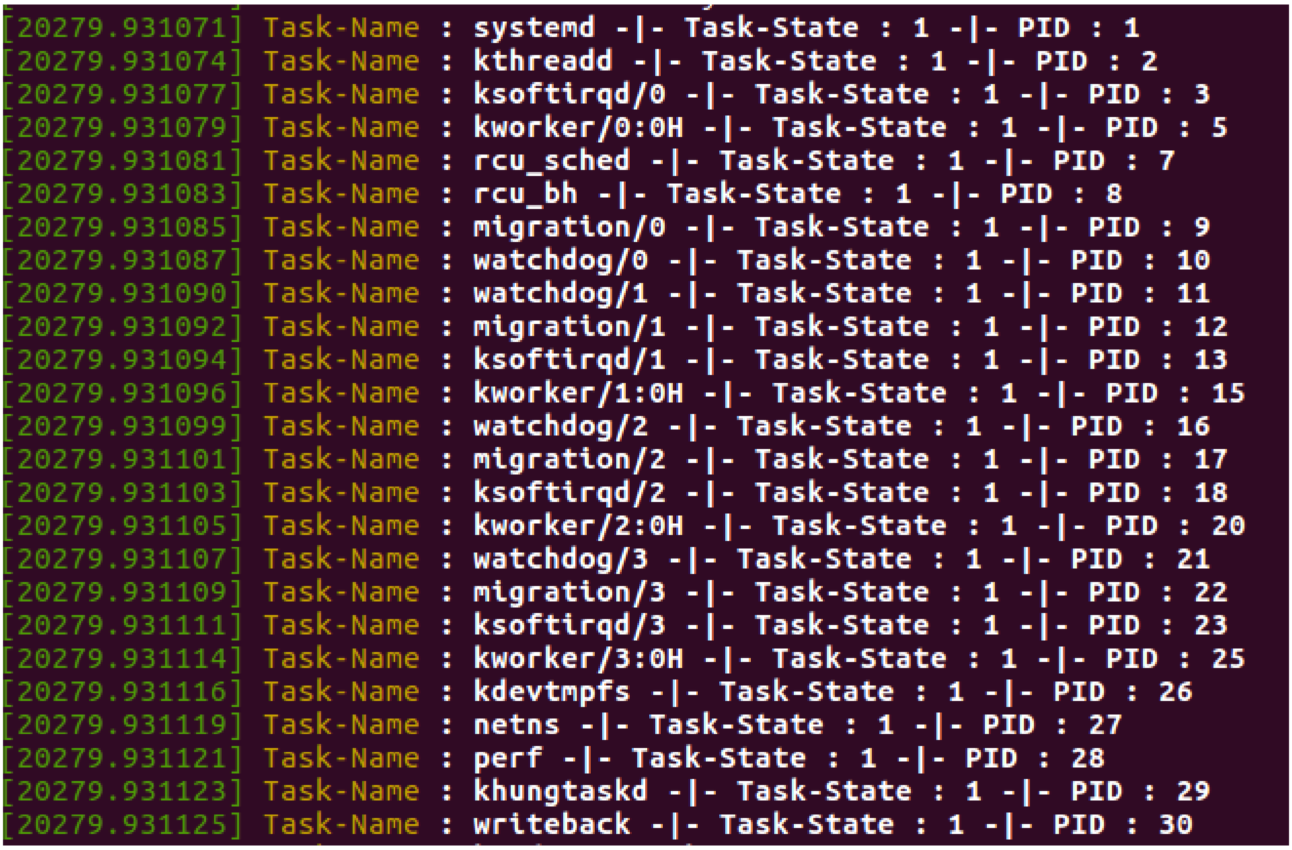
Once the code is executed, and the kernel module in loaded, the output of **printk** is inserted in the **kernel log buffer**. This output can be viewed by entering the **dmesg** command on bash terminal which shows the output of printk, and the name of the module inserted can be viewed by entering the **lsmod** command, which generally appears on the top of the list of the other kernel modules.

**lsmod Output** : Here, the name of the module is task_list -



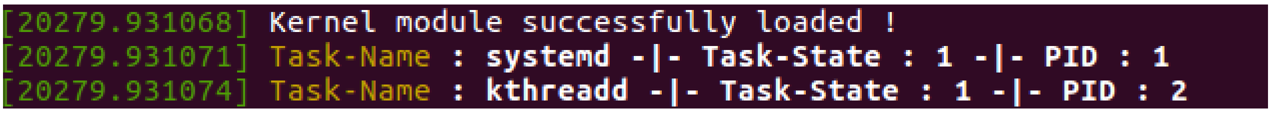
```
saurabh_cs_iith@saurabhcsiith-Inspiron-3542:~/Desktop/Assgn2-CS14BTECH11031/src$ lsmod
Module                  Size  Used by
Assgn2Src_CS14BTECH11031    16384  0
```

**dmesg Output** : This is a partial snapshot of some of the tasks, with their *name*, *state* and *PID* printed as shown below -



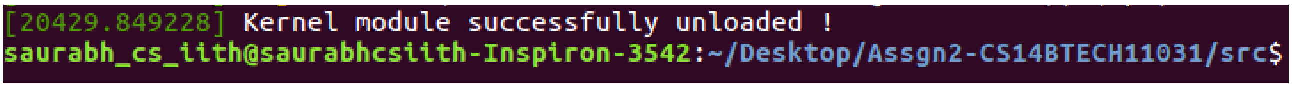
```
[20279.931071] Task-Name : systemd -|- Task-State : 1 -|- PID : 1
[20279.931074] Task-Name : kthreadd -|- Task-State : 1 -|- PID : 2
[20279.931077] Task-Name : ksoftirqd/0 -|- Task-State : 1 -|- PID : 3
[20279.931079] Task-Name : kworker/0:0H -|- Task-State : 1 -|- PID : 5
[20279.931081] Task-Name : rcu_sched -|- Task-State : 1 -|- PID : 7
[20279.931083] Task-Name : rcu_bh -|- Task-State : 1 -|- PID : 8
[20279.931085] Task-Name : migration/0 -|- Task-State : 1 -|- PID : 9
[20279.931087] Task-Name : watchdog/0 -|- Task-State : 1 -|- PID : 10
[20279.931090] Task-Name : watchdog/1 -|- Task-State : 1 -|- PID : 11
[20279.931092] Task-Name : migration/1 -|- Task-State : 1 -|- PID : 12
[20279.931094] Task-Name : ksoftirqd/1 -|- Task-State : 1 -|- PID : 13
[20279.931096] Task-Name : kworker/1:0H -|- Task-State : 1 -|- PID : 15
[20279.931099] Task-Name : watchdog/2 -|- Task-State : 1 -|- PID : 16
[20279.931101] Task-Name : migration/2 -|- Task-State : 1 -|- PID : 17
[20279.931103] Task-Name : ksoftirqd/2 -|- Task-State : 1 -|- PID : 18
[20279.931105] Task-Name : kworker/2:0H -|- Task-State : 1 -|- PID : 20
[20279.931107] Task-Name : watchdog/3 -|- Task-State : 1 -|- PID : 21
[20279.931109] Task-Name : migration/3 -|- Task-State : 1 -|- PID : 22
[20279.931111] Task-Name : ksoftirqd/3 -|- Task-State : 1 -|- PID : 23
[20279.931114] Task-Name : kworker/3:0H -|- Task-State : 1 -|- PID : 25
[20279.931116] Task-Name : kdevtmpfs -|- Task-State : 1 -|- PID : 26
[20279.931119] Task-Name : netns -|- Task-State : 1 -|- PID : 27
[20279.931121] Task-Name : perf -|- Task-State : 1 -|- PID : 28
[20279.931123] Task-Name : khungtaskd -|- Task-State : 1 -|- PID : 29
[20279.931125] Task-Name : writeback -|- Task-State : 1 -|- PID : 30
```

**sudo insmod** ***module_name*** **Output** : Message “*Kernel module successfully loaded !*” is printed once the kernel module is loaded.



```
[20279.931068] Kernel module successfully loaded !
[20279.931071] Task-Name : systemd -|- Task-State : 1 -|- PID : 1
[20279.931074] Task-Name : kthreadd -|- Task-State : 1 -|- PID : 2
```

**sudo rmmod** ***module_name*** **Output** : Message “*Kernel module successfully unloaded !*” is printed in the log buffer, once the module is successfully unloaded.



```
[20429.849228] Kernel module successfully unloaded !
saurabh_cs_iith@saurabhcsiith-Inspiron-3542:~/Desktop/Assgn2-CS14BTECH11031/src$
```

Note that **insmod** and **rmmod** commands require root privileges. The output of the **dmesg** command can be confirmed by running the **ps -el** command.

The user can compile and run the program by going into the “src” directory and running the **make** command. Then **insmod** to load the kernel module, **lsmod** to check the name of module, **dmesg** to scan the output given by *printk()*, and finally **rmmod** to remove the module from the Linux kernel.

Details regarding execution and loading/unloading is described in the readme file in detail.