Assignment-4

Roll No: 2022201063

Name: Krati Agrawal

Linux kernel module

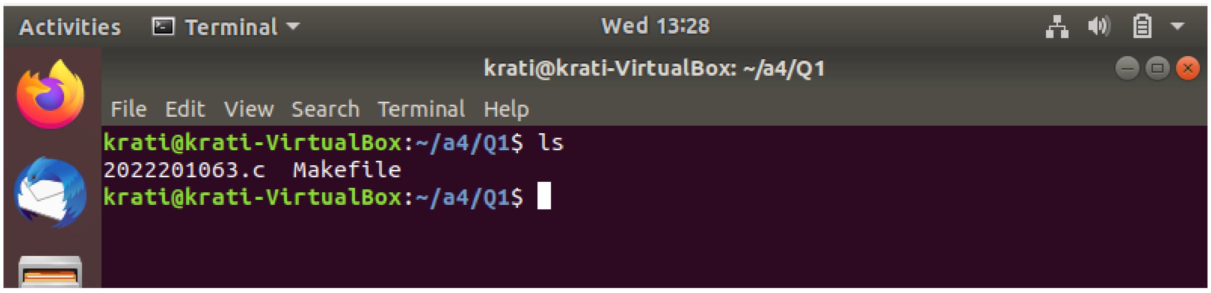
Kernel code gets execute in privileged mode where program instructions can command the CPU to perform any operation allowed

To do:

Kernel module to count no of running processes

These are the following steps to write, build, load, and remove the kernel module.

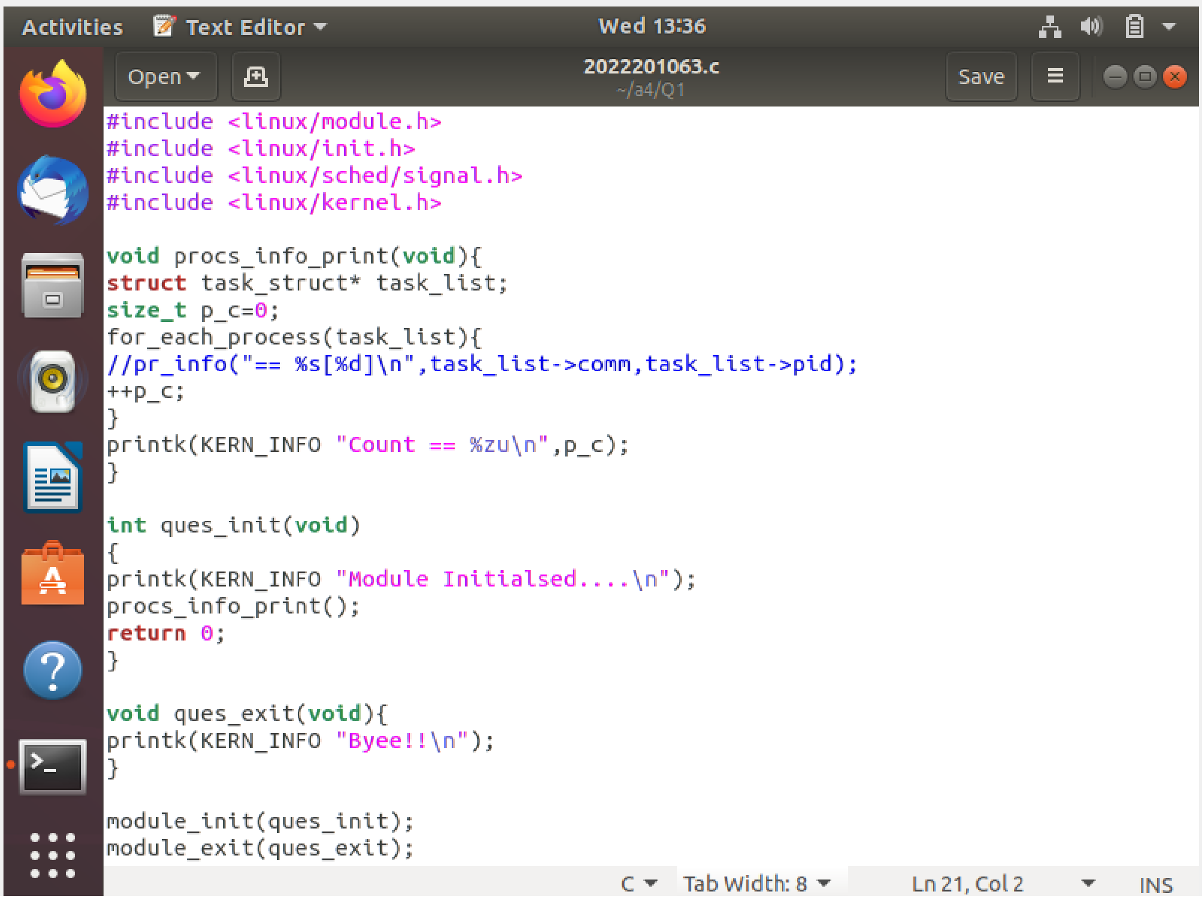
1. Step 1:
   - Download required dependencies /module
   - sudo apt install make binutils gcc
   - install flex and bison too
2. Step 2:
   - Make a directory, say Q1
   - Inside the directory make a .c file and Makefile, say 2022201063.c



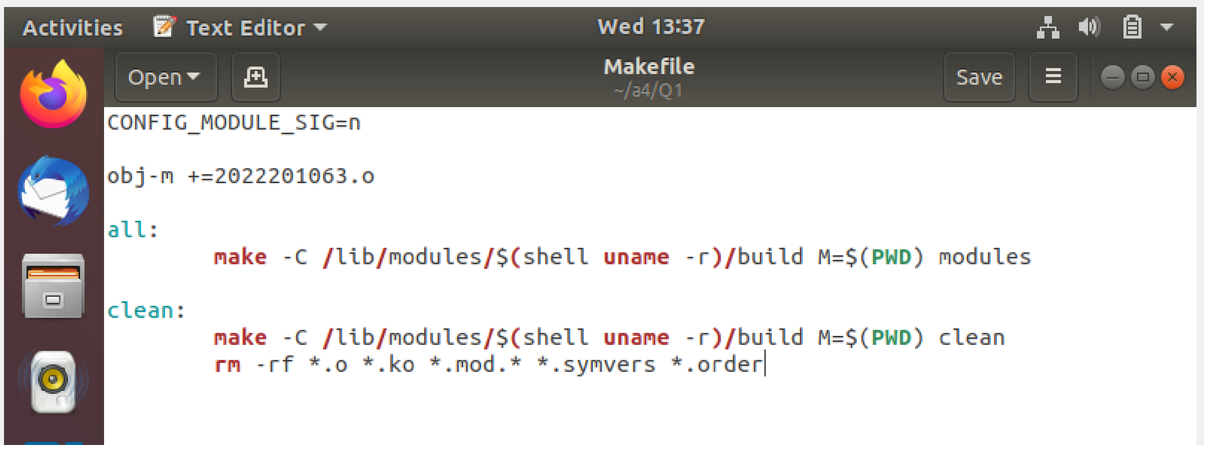
Activities
Terminal
Wed 13:28
krati@krati-VirtualBox: ~/a4/Q1
File Edit View Search Terminal Help

```
krati@krati-VirtualBox:~/a4/Q1$ ls
2022201063.c  Makefile
krati@krati-VirtualBox:~/a4/Q1$ 
```

3. Step 3:
   - In 2022201063.c write two functions that would be called on initialisation and removal of module.
   - Write one function which will use task_struct and for_each_process to count no of process
   - Call this function inside init function
   - Register init function on module initialisation
   - Register exit function on module exit
   - save the .c file
   - gedit Makefile
   - Mention commands /parameter to be used on compile

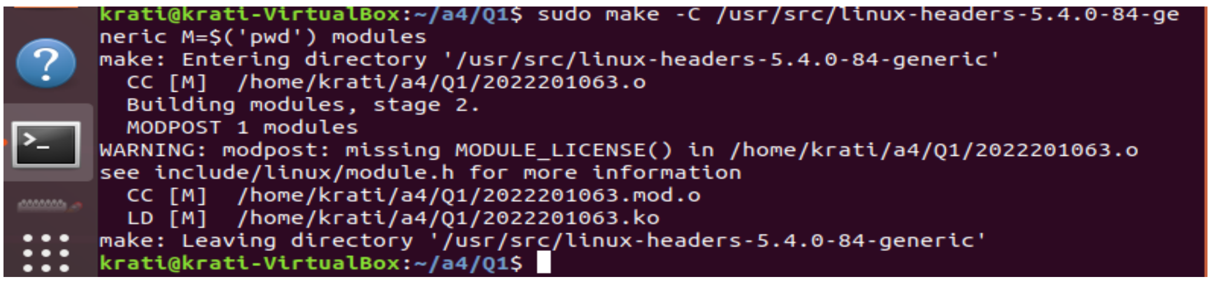


```c
#include <linux/module.h>
#include <linux/init.h>
#include <linux/sched/signal.h>
#include <linux/kernel.h>

void procs_info_print(void){
struct task_struct* task_list;
size_t p_c=0;
for_each_process(task_list){
//pr_info("== %s[%d]\n",task_list->comm,task_list->pid);
++p_c;
}
printk(KERN_INFO "Count == %zu\n",p_c);
}

int ques_init(void)
{
printk(KERN_INFO "Module Initialsed....\n");
procs_info_print();
return 0;
}

void ques_exit(void){
printk(KERN_INFO "Byee!!\n");
}

module_init(ques_init);
module_exit(ques_exit);
```



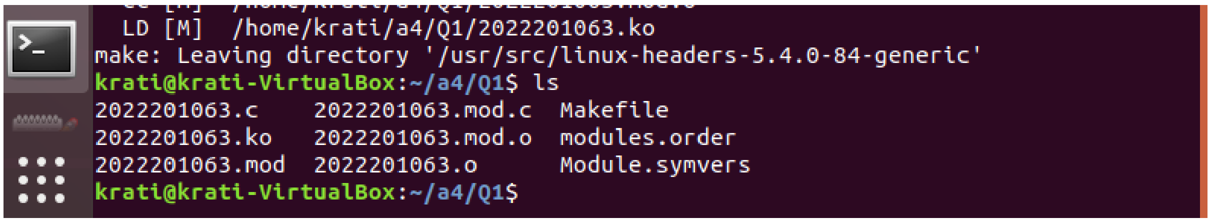
```makefile
CONFIG_MODULE_SIG=n

obj-m +=2022201063.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
	rm -rf *.o *.ko *.mod.* *.symvers *.order
```

4. Step 4
   - Use this command to compile



```
krati@krati-VirtualBox:~/a4/Q1$ sudo make -C /usr/src/linux-headers-5.4.0-84-ge
neric M=$('pwd') modules
make: Entering directory '/usr/src/linux-headers-5.4.0-84-generic'
  CC [M]  /home/krati/a4/Q1/2022201063.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/krati/a4/Q1/2022201063.o
see include/linux/module.h for more information
  CC [M]  /home/krati/a4/Q1/2022201063.mod.o
  LD [M]  /home/krati/a4/Q1/2022201063.ko
make: Leaving directory '/usr/src/linux-headers-5.4.0-84-generic'
krati@krati-VirtualBox:~/a4/Q1$
```

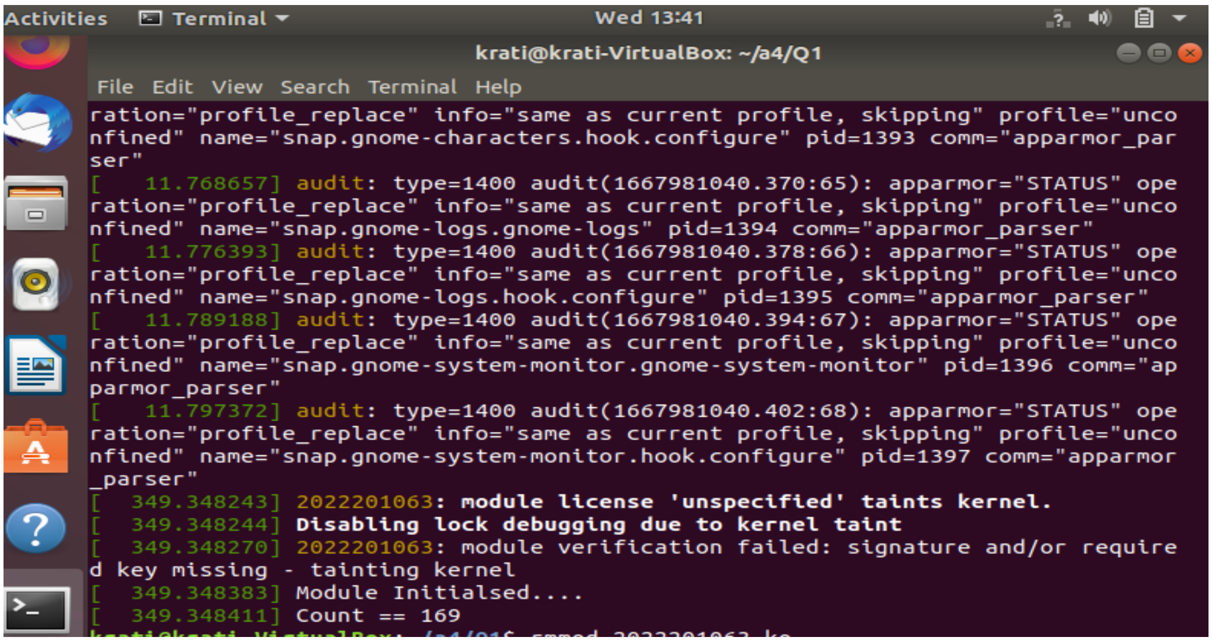
5. Step 5
   - Use **ls** to see the compiled code file, more files with extension **.ko, .mod, .symvers** etc. will get added to directory



```
  LD [M]  /home/krati/a4/Q1/2022201063.ko
make: Leaving directory '/usr/src/linux-headers-5.4.0-84-generic'
krati@krati-VirtualBox:~/a4/Q1$ ls
2022201063.c    2022201063.mod.c  Makefile
2022201063.ko   2022201063.mod.o  modules.order
2022201063.mod  2022201063.o      Module.symvers
krati@krati-VirtualBox:~/a4/Q1$
```

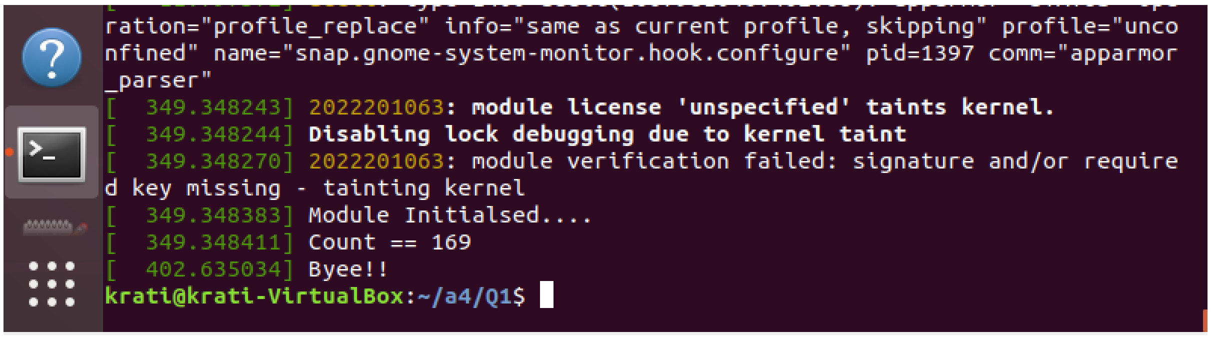
6. Step 6
   - sudo insmod 2022201063.ko (kernel object file)

7. Step 7
   - dmesg: it will print the initialisation function output


Activities
Terminal
Wed 13:41
krati@krati-VirtualBox: ~/a4/Q1
File Edit View Search Terminal Help

```
ration="profile_replace" info="same as current profile, skipping" profile="unco
nfined" name="snap.gnome-characters.hook.configure" pid=1393 comm="apparmor_par
ser"
[   11.768657] audit: type=1400 audit(1667981040.370:65): apparmor="STATUS" ope
ration="profile_replace" info="same as current profile, skipping" profile="unco
nfined" name="snap.gnome-logs.gnome-logs" pid=1394 comm="apparmor_parser"
[   11.776393] audit: type=1400 audit(1667981040.378:66): apparmor="STATUS" ope
ration="profile_replace" info="same as current profile, skipping" profile="unco
nfined" name="snap.gnome-logs.hook.configure" pid=1395 comm="apparmor_parser"
[   11.789188] audit: type=1400 audit(1667981040.394:67): apparmor="STATUS" ope
ration="profile_replace" info="same as current profile, skipping" profile="unco
nfined" name="snap.gnome-system-monitor.gnome-system-monitor" pid=1396 comm="ap
parmor_parser"
[   11.797372] audit: type=1400 audit(1667981040.402:68): apparmor="STATUS" ope
ration="profile_replace" info="same as current profile, skipping" profile="unco
nfined" name="snap.gnome-system-monitor.hook.configure" pid=1397 comm="apparmor
_parser"
[  349.348243] 2022201063: module license 'unspecified' taints kernel.
[  349.348244] Disabling lock debugging due to kernel taint
[  349.348270] 2022201063: module verification failed: signature and/or require
d key missing - tainting kernel
[  349.348383] Module Initialsed....
[  349.348411] Count == 169
```

8. Step 8
   - We can use sudo rmmod 2022201063.ko (to remove the module)
   - Do dmesg
   - It will print the exit function printk



```
ration="profile_replace" info="same as current profile, skipping" profile="unco
nfined" name="snap.gnome-system-monitor.hook.configure" pid=1397 comm="apparmor
_parser"
[  349.348243] 2022201063: module license 'unspecified' taints kernel.
[  349.348244] Disabling lock debugging due to kernel taint
[  349.348270] 2022201063: module verification failed: signature and/or require
d key missing - tainting kernel
[  349.348383] Module Initialsed....
[  349.348411] Count == 169
[  402.635034] Byee!!
krati@krati-VirtualBox:~/a4/Q1$
```

** Note:

Since the module/code get executed in kernel mode we can only use kernel functions like printk instead of printf