Dynamically inserting & removing LINUX module to LINUX Kernel

System configuration

Ubuntu version : ubuntu-18.04
System architecture : x86 → Check with uname -m

Objective : The inspiration driving this assignment is to introduce Linux Kernel programming concerning creating a Linux module. Portion code executes to some extent (piece mode) where program headings can arrange the microprocessor to play out any movement. This allows any action to be executed.

Pre-setup

Introduce ubuntu 18.04 on a virtual machine and run the accompanying commands
        sudo apt update
        sudo apt install make binutils gcc

Downloading the standard code for a kernel module
         mkdir a3 cd a3 wget
        http://faculty.washington.edu/wlloyd/courses/tcss422/assignments/hello_modul e.tar.gz tar xzf hello_module.tar.gz
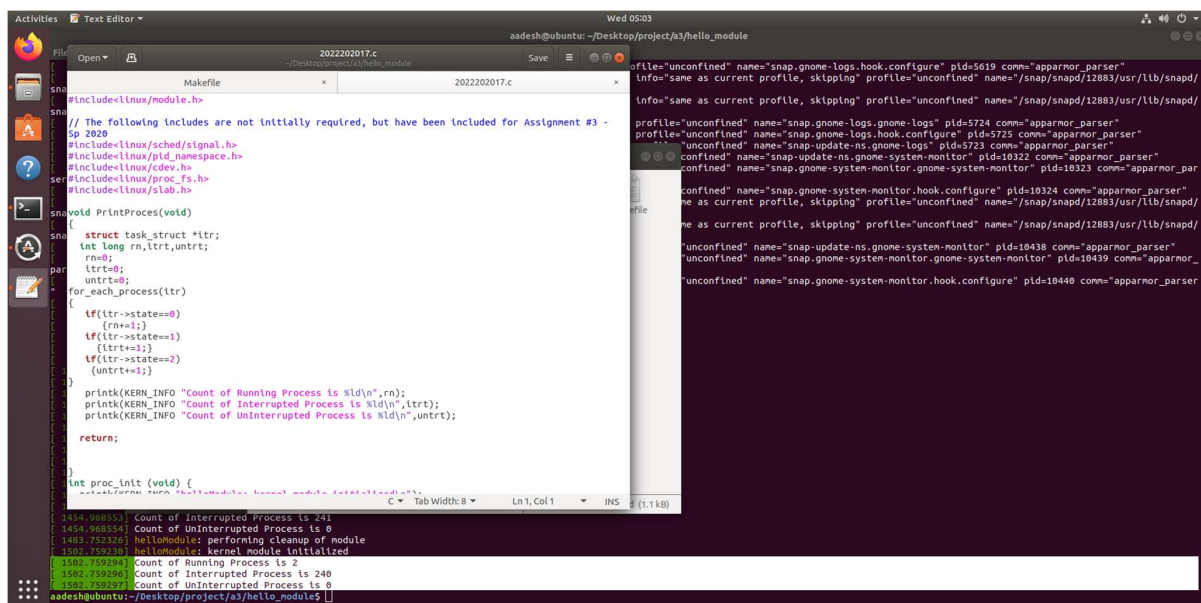
The **helloModule.c** contains the boiler plate code for the basic Kernel module.
The two functions that represent the crux while initializing & un-initializing kernel module are
1. **int proc_init(void) : called when the kernel is initialized**
2. **void proc_cleanup(void) : called when the kernel is removed**
Outputs for compiling, testing & test a kernel module

Source code for kernel module to find all, interrupted & uninterrupted processes.


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

// The following includes are not initially required, but have been included for Assignment #3 - Sp 2020
#include<linux/sched/signal.h>
#include<linux/pid_namespace.h>
#include<linux/cdev.h>
#include<linux/proc_fs.h>
#include<linux/slab.h>

void PrintProces(void)
{
    struct task_struct *itr;
    int long rn,itrt,untrt;
    rn=0;
    itrt=0;
    untrt=0;
for_each_process(itr)
{
    if(itr->state==0)
        {rn+=1;}
    if(itr->state==1)
        {itrt+=1;}
    if(itr->state==2)
        {untrt+=1;}
}
    printk(KERN_INFO "Count of Running Process is %ld\n",rn);
    printk(KERN_INFO "Count of Interrupted Process is %ld\n",itrt);
    printk(KERN_INFO "Count of UnInterrupted Process is %ld\n",untrt);

    return;

}
int proc_init (void) {
    printk(KERN_INFO "helloModule: kernel module initialized\n");
```
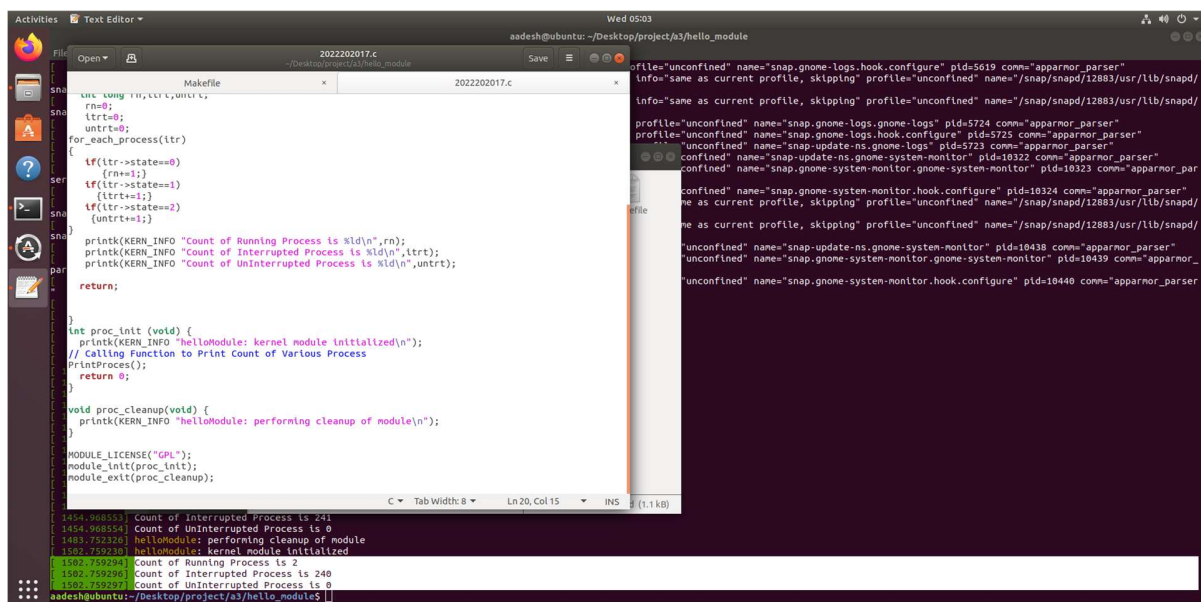


```c
    int long rn,itrt,untrt;
    rn=0;
    itrt=0;
    untrt=0;
for_each_process(itr)
{
    if(itr->state==0)
        {rn+=1;}
    if(itr->state==1)
        {itrt+=1;}
    if(itr->state==2)
        {untrt+=1;}

    printk(KERN_INFO "Count of Running Process is %ld\n",rn);
    printk(KERN_INFO "Count of Interrupted Process is %ld\n",itrt);
    printk(KERN_INFO "Count of UnInterrupted Process is %ld\n",untrt);

    return;

}
int proc_init (void) {
    printk(KERN_INFO "helloModule: kernel module initialized\n");
// Calling Function to Print Count of Various Process
PrintProces();
    return 0;
}

void proc_cleanup(void) {
    printk(KERN_INFO "helloModule: performing cleanup of module\n");
}

MODULE_LICENSE("GPL");
module_init(proc_init);
module_exit(proc_cleanup);
```

Printing the output after dynamically removing the kernel module