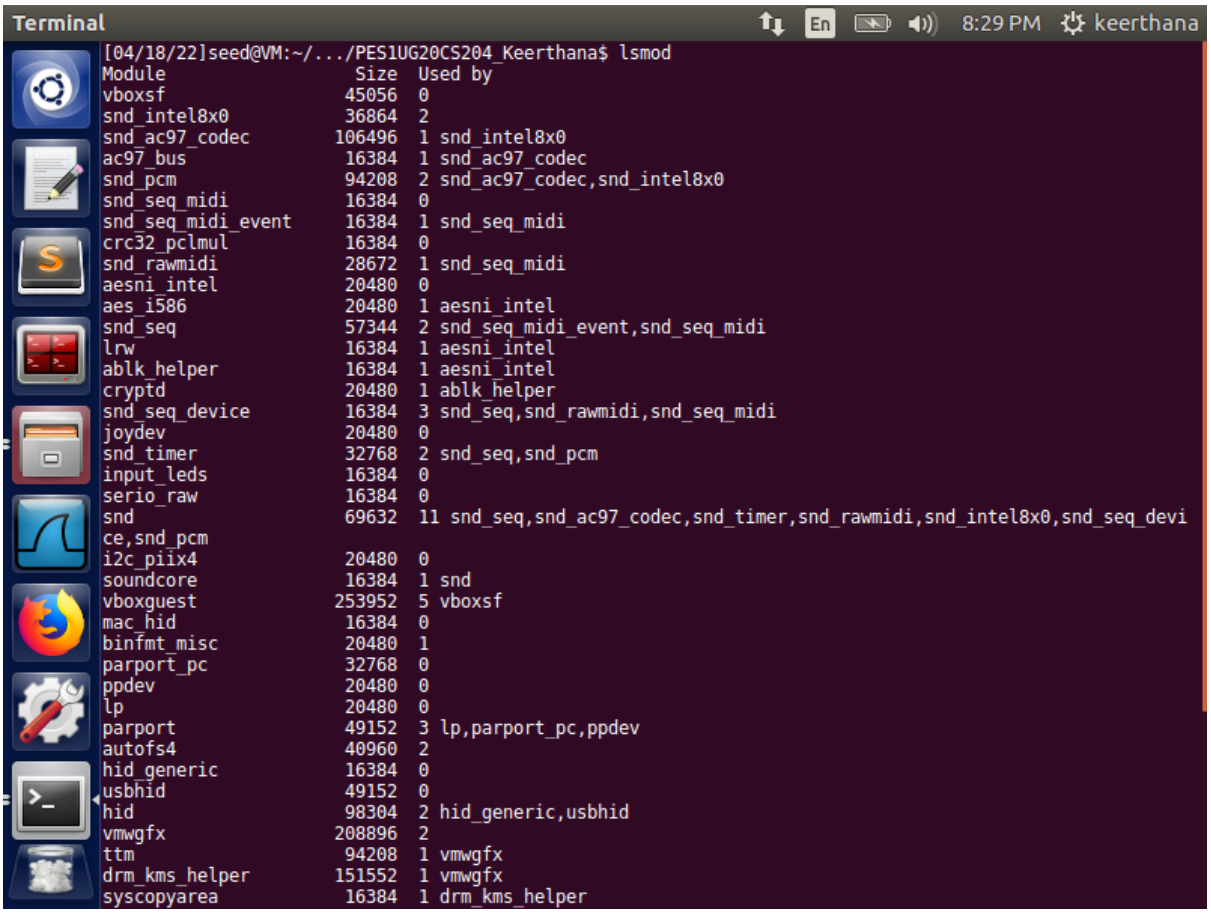
Keerthana Shivakumar
PES1UG20CS204
Operating Systems Project

Write a kernel module that lists all current tasks in a Linux system beginning from the init task. Refer to Chapter 2 in the text book for creating Linux kernel modules. Output the task name (known as executable name), state and process id of each task in a tree structure.

Step 1 – Creating Kernel Modules

Command lsmod can be used to list all the kernel modules that are currently loaded.



Program for kernel – project3.c

//Code includes test cases created

#include<linux/init.h>

#include<linux/kernel.h>

```
#include<linux/module.h>
#include<linux/printk.h>
#include<linux/sched.h>
void dfs(struct task_struct *task){
        struct task_struct *task_next;
        struct list_head *list;
        list_for_each(list, &task->children) {
                task_next = list_entry(list, struct task_struct, sibling);
                if(task->pid==0)//child process
                printk(KERN_INFO "\t\tChild:- pid: %d | pname: %s | state: %ld\n",
task_next->pid, task_next->comm, task_next->state);
                else if(task->pid>0)//parent process
                printk(KERN_INFO "Parent:- pid: %d | pname: %s | state: %ld\n",
task_next->pid, task_next->comm, task_next->state);
                dfs(task_next);


                }
}
int tasks_lister_dfs_init(void)
{
  printk(KERN_INFO "Loading module...\n");
  dfs(&init_task);
  printk(KERN_INFO "Module loaded.\n");
  return 0;
}
void tasks_lister_dfs_exit(void)
{
```

```
  printk(KERN_INFO "Module removed.\n");
}
module_init(tasks_lister_dfs_init);
module_exit(tasks_lister_dfs_exit);
```

About the code

- linux/init.h – header file includes all things related to _init part.
- linux/module.h – header file where module_init (tells the kernel what is the entry point to our program) and module_exit (tells the kernel what is the exit point to our program) are present.
- linux/kernel.h – header file where all the workload happening features are listed.
- linux/sched.h – header file that contains scheduling parameters required for implementation of each supported scheduling policy.
- task_struct – declared under linux/sched.h.
- list_for_each – used to iterate over a list.
- list_entry – gets the struct for this entry. Arguments are the struct head pointer, type of struct it is embedded in and name of the head pointer within the struct.
- task_next->pid is to display the process ID of the task.
- task_next->comm is the command that triggered that event.
- task_next->state is used to indicate the process state. (In this case, all events are interruptible, since their states are displayed as 1)

```
- #define TASK_RUNNING              0
- #define TASK_INTERRUPTIBLE        1
- #define TASK_UNINTERRUPTIBLE      2
- #define TASK_ZOMBIE               4
- #define TASK_STOPPED              8
```

- The function tasks_lister_dfs_init is used to load a module and traverse the tree in DFS (depth first search).
- The function tasks_lister_dfs_exit is to remove the module.

Makefile – used for compiling the kernel module

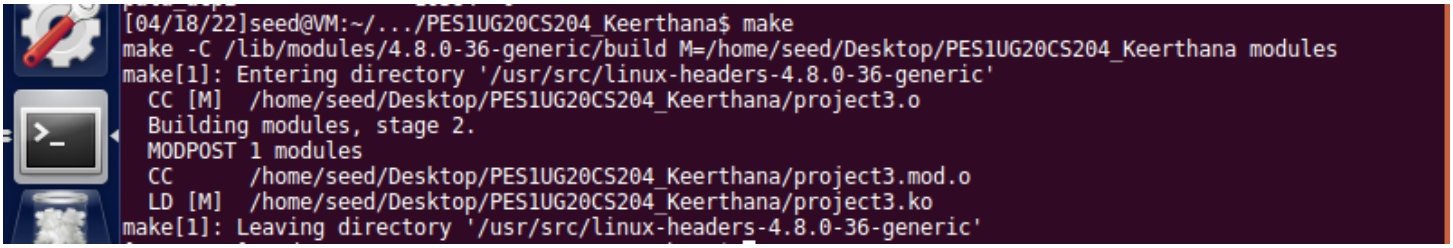
```
obj-m += project3.o
all:
```

make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

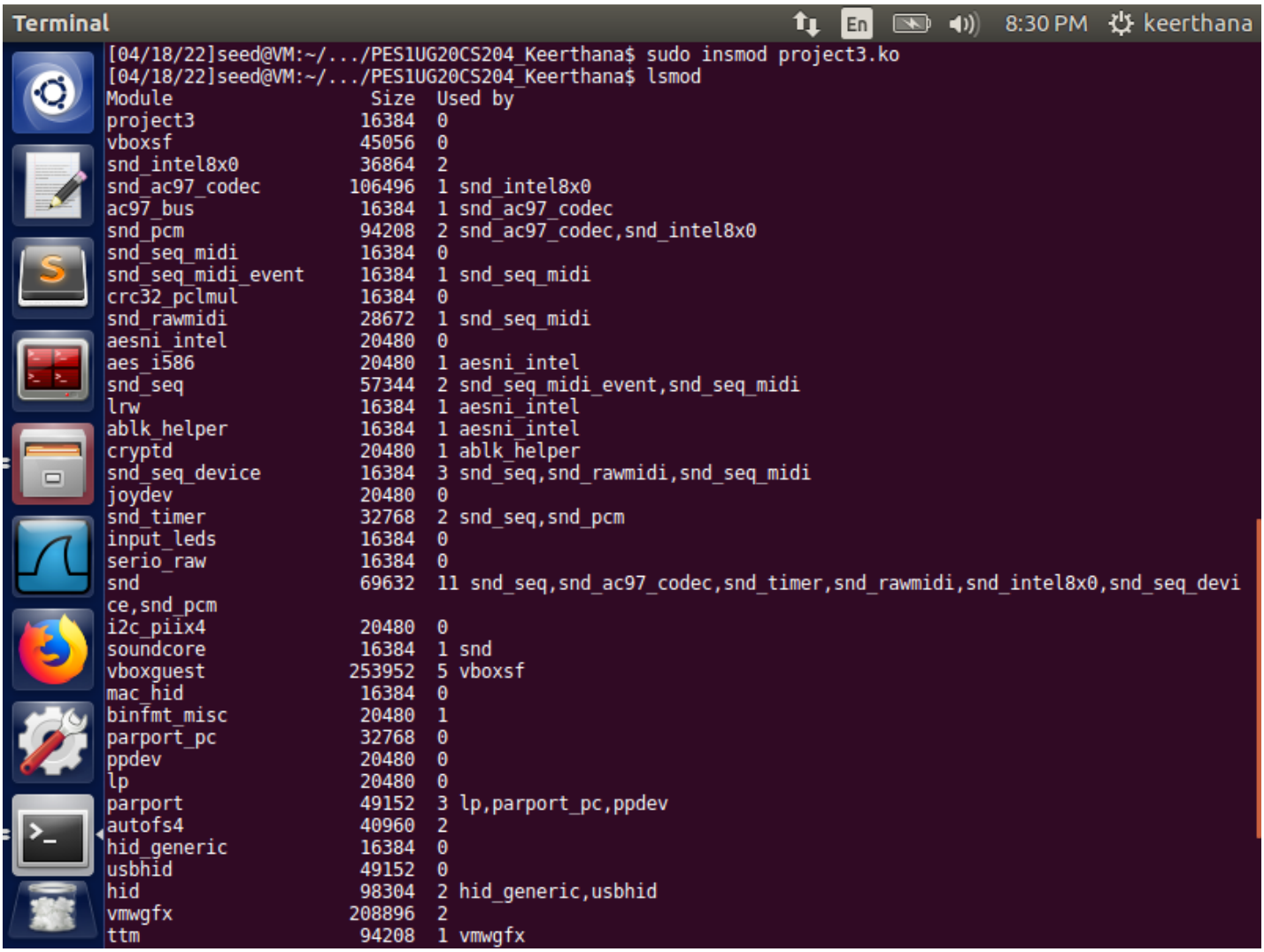
clean:

make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean

Step 2 - Compile the project3.c code using make command – files like project3.ko (compiled kernel module) , project3.mod.o (object file) and project3.o (object file) get created.



Step 3 - Load the kernel using the insmod command – sudo insmod project3.ko. It can be seen that the project3 module has been added.



Step 4 - To check the contents of the message in the kernel log buffer, command dmesg can be used. It will display all the contents of the module in the tree structure, traversed in the depth first search manner.