

NAME: Soham Sarkar

SRN: PES2UG21CS532

Section: I

**Project Title: Linux kernel module that lists all the currently running processes in the system and their corresponding states.**

[Github link for my Linux kernel module](https://github.com/Sohoxic/Kernel-Module)

This is a simple Linux kernel module written in C programming language that lists all the currently running processes in the system and their corresponding states.

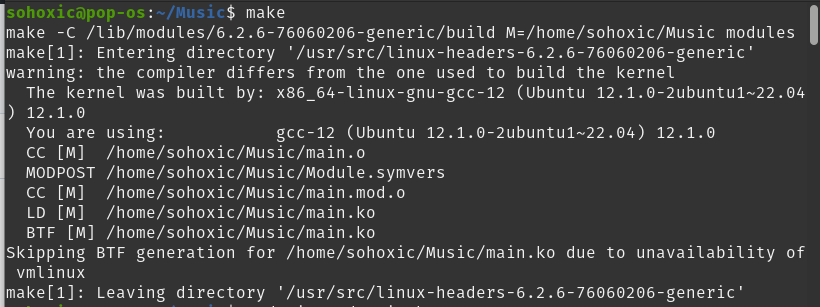
**How to run the kernel code?**

0. Execute the following command to install all the header files and dependencies required for the kernel module.

sudo apt-get install build-essential linux-headers-$(uname -r)

1. Execute the `make` command to create the `.ko` file along with other files.<br> [P.S: The makefile and the main.c file should be in the same directory(at the same directory level)]

make



2. Insert module into the kernel using `insmod`.

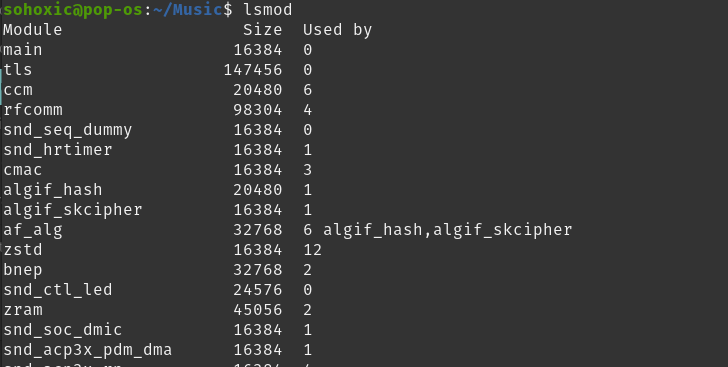
sudo insmod main.ko



3. To check if the kernel module is inserted into the kernel at runtime execute the `lsmod` command. This shows which loadable kernel modules are currently loaded(we can see the main module at the top.)

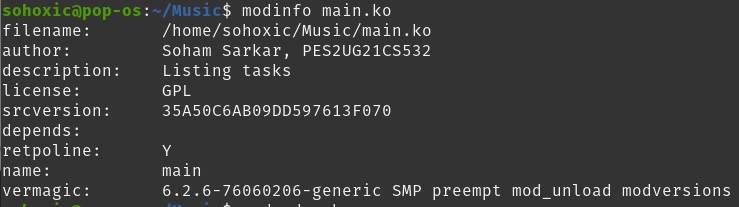
lsmod





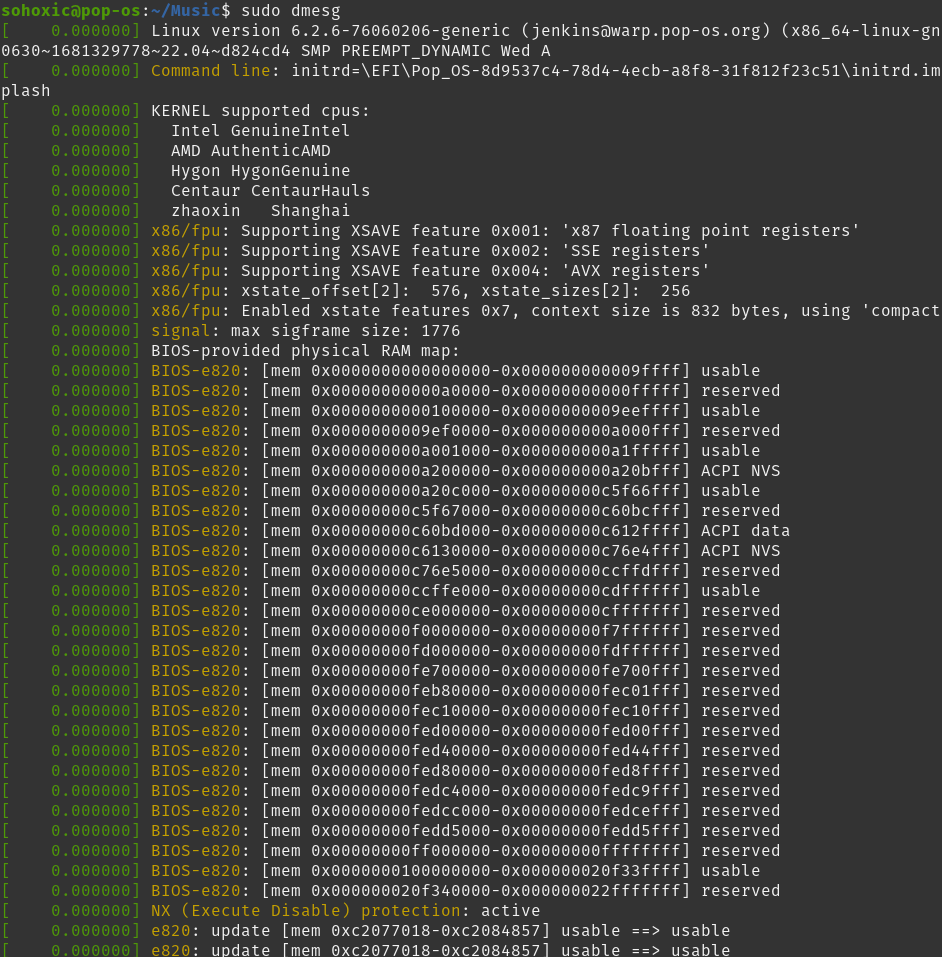
4. Execute `modinfo` command to display information about the kernel module.

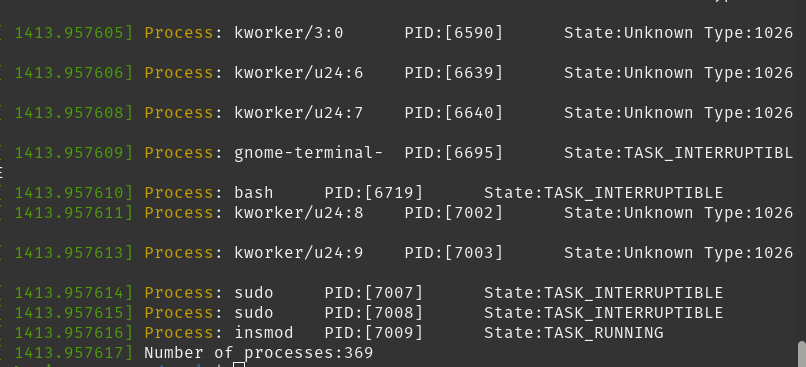
modinfo main.ko



5. `dmesg` is a useful command-line tool that provides a convenient way to access and analyze kernel messages, making it an important tool for system administration and troubleshooting in Linux/Unix systems.

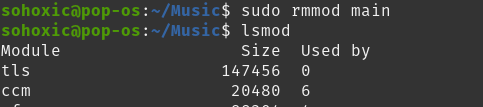
sudo dmesg





6. Now remove the kernel module from the kernel using `rmmod` and then using `lsmod` to view that the module is no longer part of the kernel. [P.S - Only check the first line after executing lsmod to see whether the kernel module name `main` is there or not.]

sudo rmmod main



**Code Explanation**

* The module first includes the necessary header files, which contain definitions of various data structures and functions used in the module.
* The get\_task\_state function takes a process state as an argument and returns a string representation of that state. It does this by using a switch statement to match the state with one of the pre-defined constants, and returns a string representation of that constant. If the state is not recognized, it generates an "Unknown Type" message and returns it in the buffer.
* The test\_tasks\_init function is the entry point for the module, which is called when the module is loaded. It declares a pointer to the task\_struct data structure, which represents a process in the Linux kernel. It then iterates over all processes in the system using the for\_each\_process macro, which iterates over a linked list of processes. For each process, it prints the process name, process ID, and process state by calling the pr\_info function. Finally, it prints the total number of processes found.
* The test\_tasks\_exit function is the exit point for the module, which is called when the module is unloaded. It simply prints a message indicating that the module is being unloaded.
* The module is licensed under the GPL, has a description and author information, and specifies the entry and exit points of the module using the module\_init and module\_exit macros.