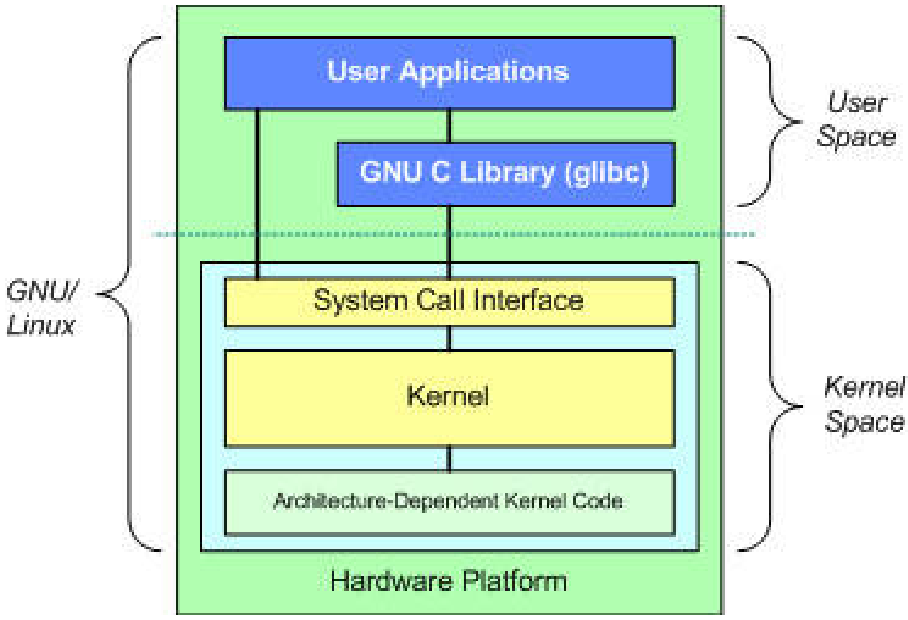
Introduction to Linux Kernel



- In linux system , several concurrent processes perform different tasks.
- Each process asks for system resources , be it computing power, memory , network connectivity or some other resources.
- The kernel is the big chunk of executable code in charge of handling all such requests.
- Although the distinction between the different kernel tasks isn't always clearly marked , the kernel's role can be split into the following parts:

Process Management
Memory Management
File System
Device Control
Networking

Process Management

- Kernel is in charge of creating and destroying processes.
- Communication among different processes.
- Scheduling the process.

Memory Management

- Computer's Memory is a major Resource and the policy used to deal with it is a critical one for system performnace.
- Kernel buils up a virtual addressing space on top of the limited available physical memory resources.

File System

- Linux is heavily based in the file system concept, lamost everything in unix can be treated as a file
- Kernel builds a structured file system on top of unstructured hardware, and the resulting file abstraction is heavily used throughout the whole system.

Device Driver

- Almost every system operation eventually maps to a physical device, with the exception of the processor , memory etc.
- The Kernel msut embedded in it a device driver for every peripheral on a system,from the hardrive to the keyboard.

Networking

- Networking must be managed by the operating system, because most network opeartions are not specific to a process; incoming packets are asynchronous events.

- Packets must be collected, identified , and dispacted before a process takes care of them. For example, windows media player playing songs, flash player videos etc.
- System is in charge of delivering data packets across program and network interface, and it must control the execution of programs according to their network activity.

- Additionally, all the routing and address resolution issues are implemented within kernel. For exxample, local host , intranet , domain name search based on hosts files, etc.

Introduction to Device Driver

- Device drivers take on a special role in the linux kernel.

- Device drivers completely hide the details of how the device works.

- User activites are performed by means of a set of standardized calls that are independent of the specified driver.

- Mapping those calls to device-specific operations that act on real hardware is then the role of the device driver.

- Drivers may be integrated directly into the kernel, or can be built separtely from the rest of the kernel and ‘plugged in’ at runtime when neeeded, i.e. loadable modules.

- Each driver is different; as a driver writer, you need to understand your specific device well. But most of the principles are basic techniques are the same for all drivers.

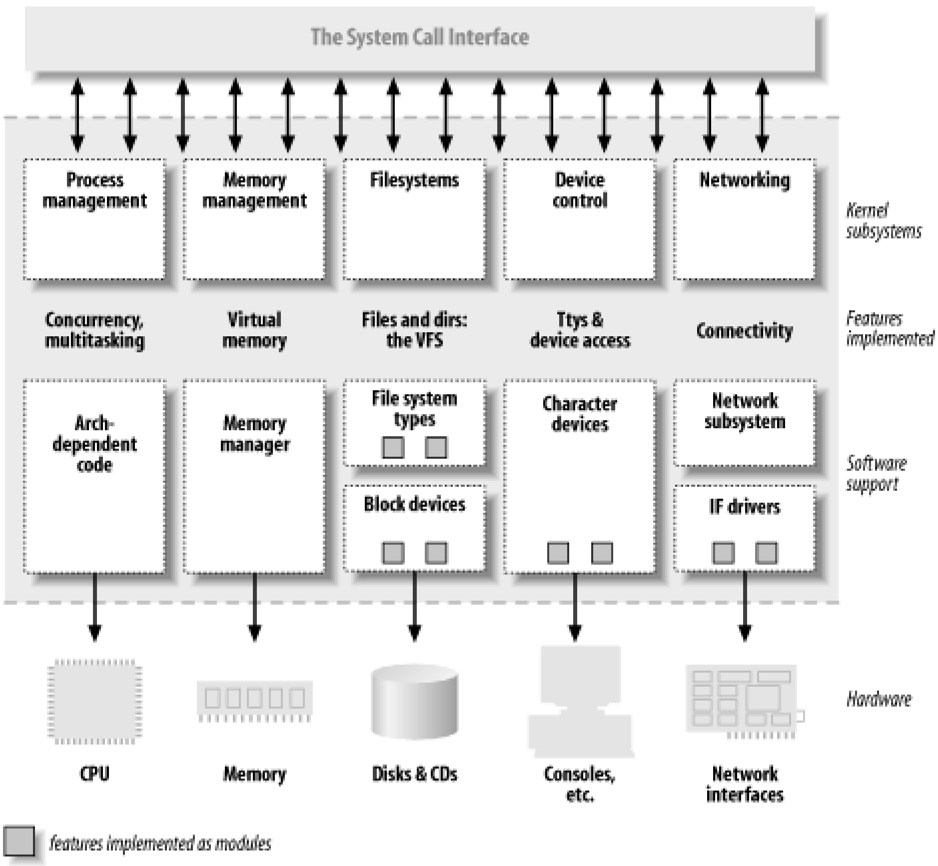
Classes of Devices and Modules

Linux way of looking at devices distinguishes between three fundamental device types :

- Character Devices
- Block Devices
- Network Devices

- Linux Drivers/modules usually implement one of these types.

- This division of modules into different types, or classes , is not a rigid one. It is a good programming pratice for scalability ans extendibility.



Character Drivers

- A character (char) device is one that can be accessed as a stream of bytes (like a file).

- A char driver is in charge of implementing this behaviour. For example , keyboard , mouse , camera , etc.

- The only relevant difference between a char device and a regular file is that you can always move back and forth in the regular file, whereas most char devices are just data channels , which you can only access sequentially.

Block Drivers

- Like char devices, block devices are accessed by file system nodes in the /dev directory.

- A block device is a device (e.g., a disk) that can host a file system. In most Unix systems , a block device can only handle I/O operations that transfer one or more whole blocks, which are usually 512 bytes (or a larger power of two) bytes in length.

- Block device drivers permit random access.

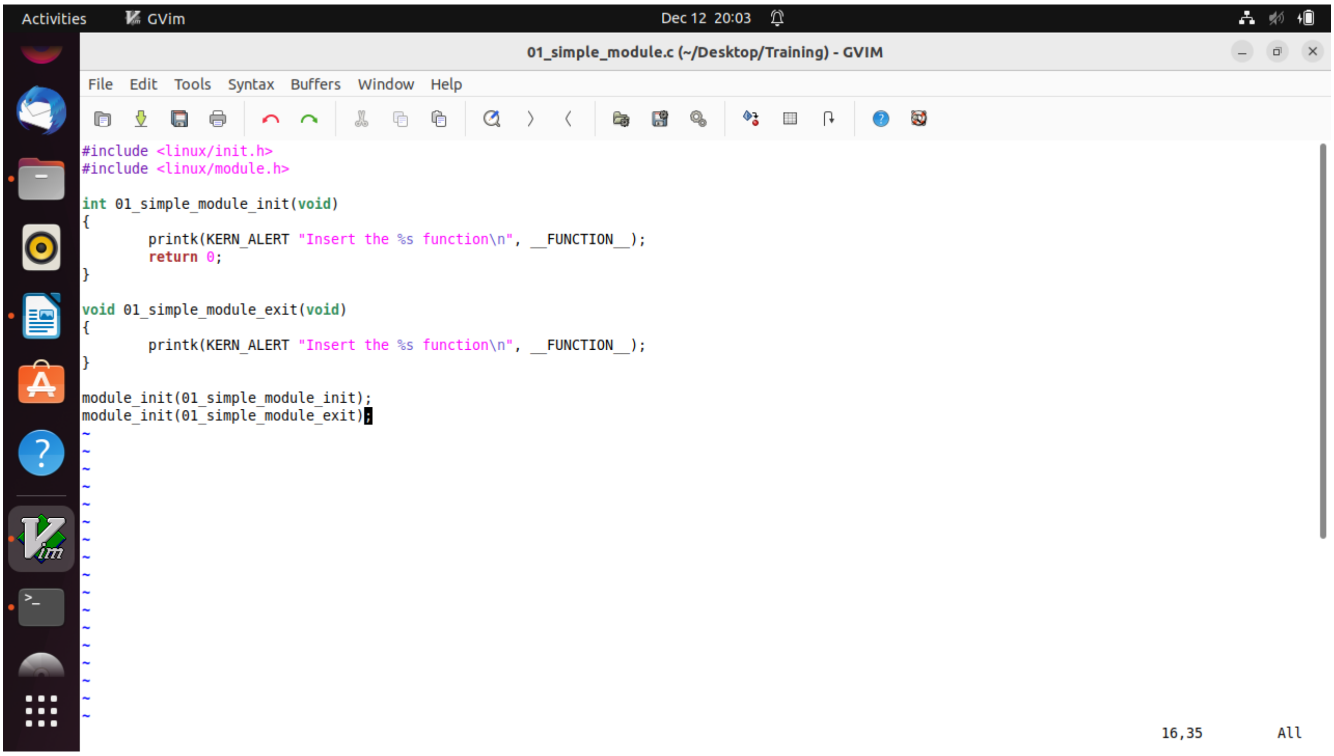


Fig 1. Sample LKM .c file

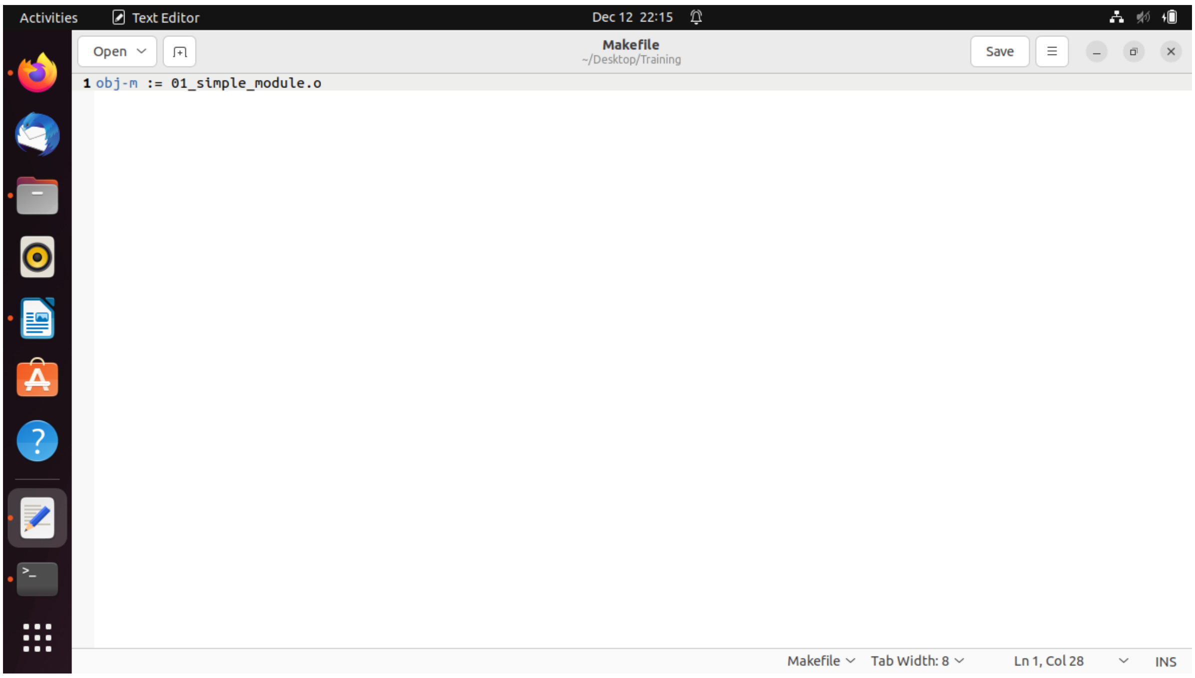


Fig 2. Makefile for the corresponding LKM .c file

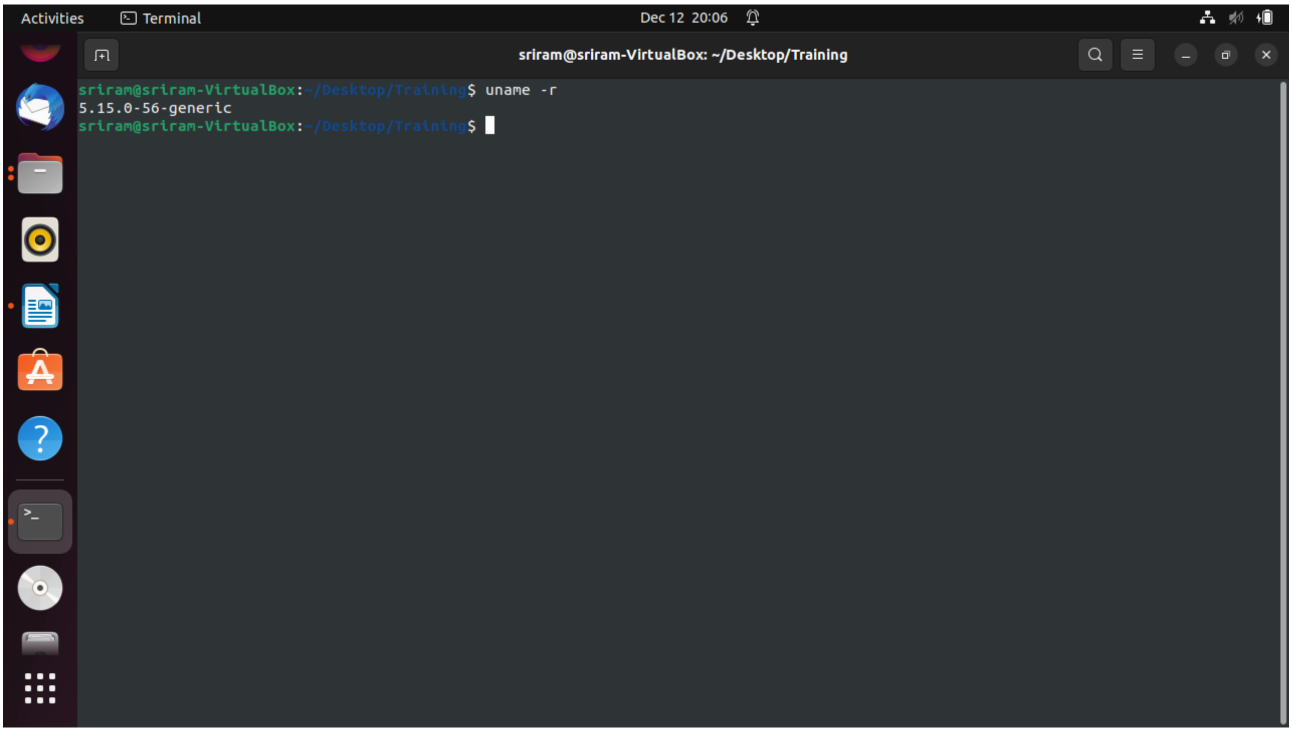


Fig 3. To display unix kernel version

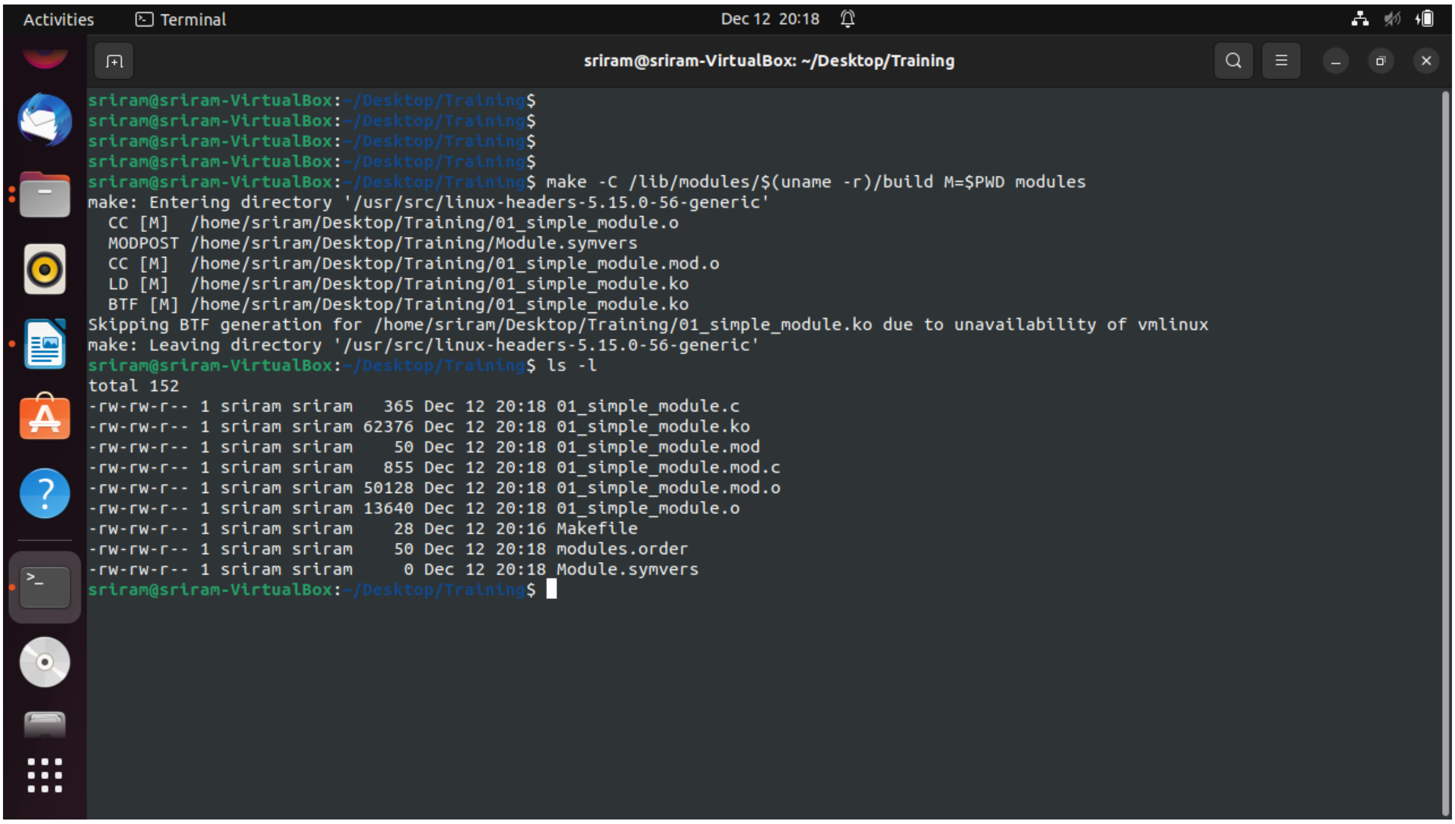


Fig 4. Linux Command to create kernel Object and display in current working directory

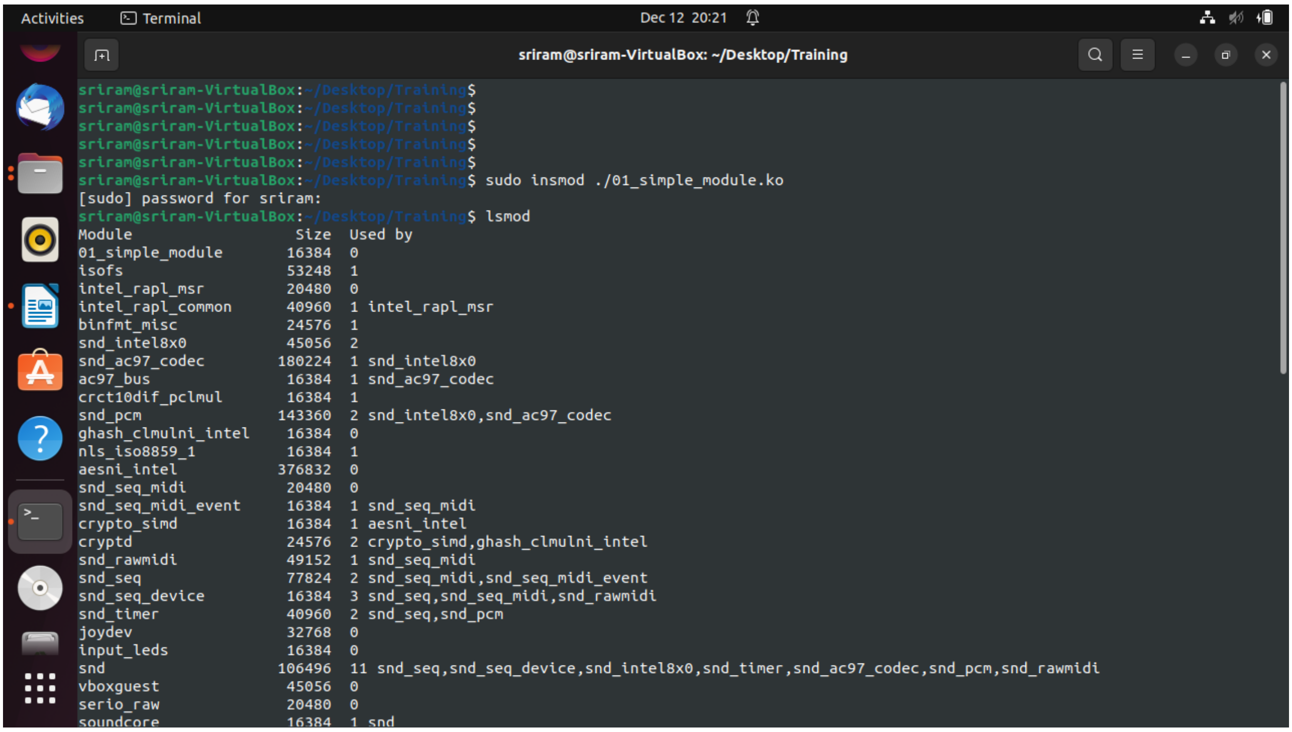


Fig 5. Command to display loaded modules in the kernel

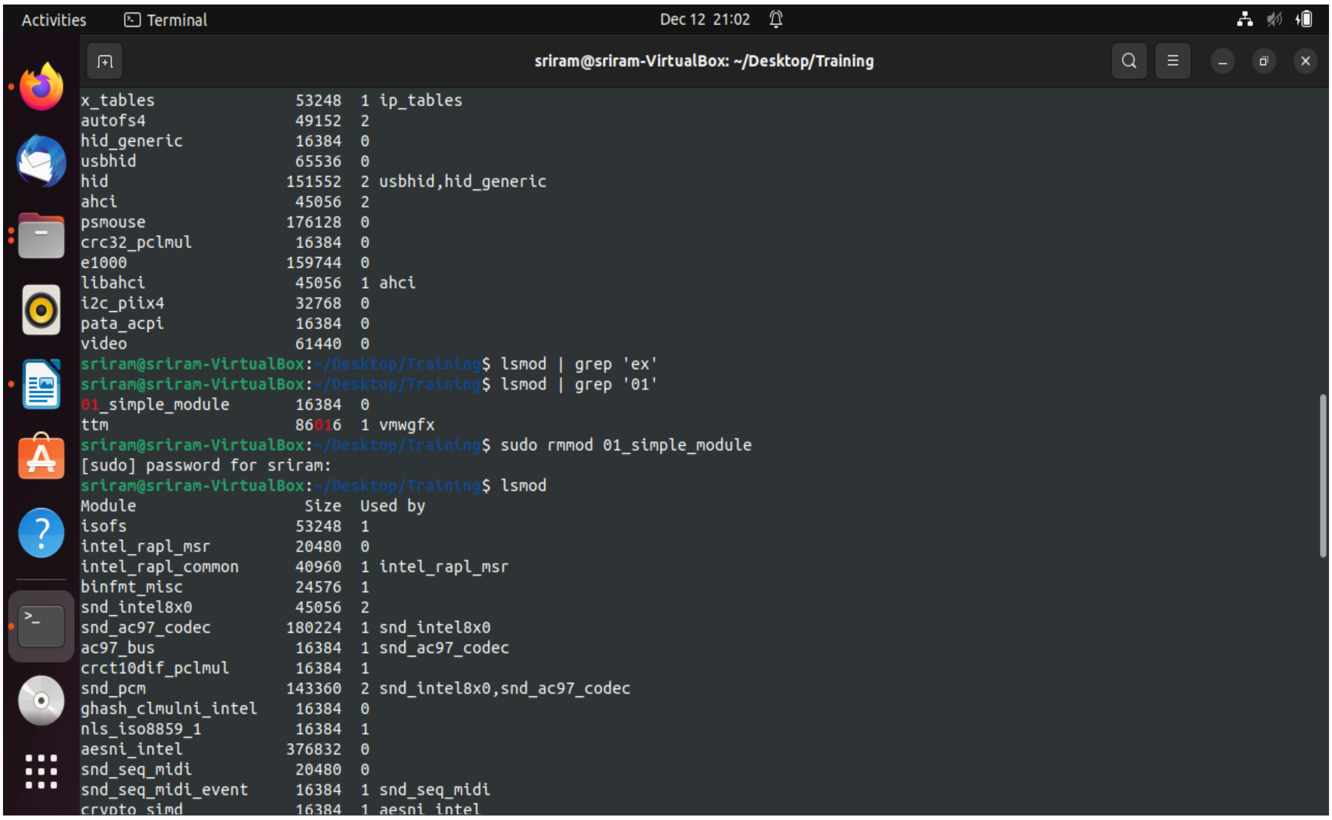


Fig 5a. Command to display loaded modules in the kernel